

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA E INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM ENGENHARIA COMPUTACIONAL

Maria Eduarda Ribeiro Facio

**Mapeamento de modelos de processo de negócio em artefatos de
especificação de software**

Juiz de Fora

2026

Ribeiro Facio, Maria Eduarda.

Mapeamento de modelos de processo de negócio em artefatos de especificação de software / Maria Eduarda Ribeiro Facio. - 2026.
123 f.

Orientador: Fabricio Martins Mendonça

Trabalho de Conclusão de Curso (Bacharelado) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia e Instituto de Ciências Exatas. Bacharelado em Engenharia Computacional, 2026.

1. BPMN. 2. UML. 3. especificação de software. 4. mapeamento de modelos. 5. inteligência artificial. I. Martins Mendonça, Fabricio, orient. II. Título.

Maria Eduarda Ribeiro Facio

**Mapeamento de modelos de processo de negócio em artefatos de
especificação de software**

Trabalho de Conclusão de Curso apresentado
ao corpo docente do curso de graduação em
Engenharia Computacional da Faculdade de
Engenharia e Instituto de Ciências Exatas
da Universidade Federal de Juiz de Fora
como requisito parcial à obtenção do grau de
Bacharel em Engenharia Computacional.

Orientador: Prof. Dr. Fabricio Martins Mendonça

Juiz de Fora

2026

Maria Eduarda Ribeiro Facio

**Mapeamento de modelos de processo de negócio em artefatos de
especificação de software**

Trabalho de Conclusão de Curso apresentado
ao corpo docente do curso de graduação em
Engenharia Computacional da Faculdade de
Engenharia e Instituto de Ciências Exatas
da Universidade Federal de Juiz de Fora
como requisito parcial à obtenção do grau de
Bacharel em Engenharia Computacional.

Aprovado em 16 de janeiro de 2026.

BANCA EXAMINADORA

Prof. Dr. Fabricio Martins Mendonça - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Gleiph Ghiotto Lima de Menezes
Universidade Federal de Juiz de Fora

Prof. Dr. Ruy Freitas Reis
Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Agradeço aos meus pais, Luzia e Eduardo, por todo o apoio ao longo da graduação. À minha mãe, em especial, pelo cuidado e pela força que certamente me fizeram chegar até aqui. Ao meu pai, por acreditar em mim, me incentivar e estar, à sua maneira, sempre por perto.

À minha irmã, Ana Laura, pela companhia e pelo carinho de sempre, pelas conversas e por torcer por mim mesmo nos dias mais difíceis. Obrigada por ser casa e abrigo.

Agradeço ao meu namorado, Enzo, por caminhar comigo nessa reta final com tanta paciência, parceria e amor, deixando tudo mais leve.

Às minhas melhores amigas Isabela, Beatriz, Eliza e Isabella, por todos os conselhos, risadas e por estarem sempre comigo quando eu precisei. Vocês fizeram mais diferença do que imaginam.

Um agradecimento especial ao meu orientador, Fabricio, pela orientação brilhante, pela atenção e pelo cuidado em cada detalhe. Obrigada por me guiar com tanta seriedade e por elevar o nível do meu trabalho do início ao fim.

Finalmente, agradeço aos professores Gleiph Ghiotto e Ruy Freitas, por aceitarem compor a banca e, principalmente, por terem sido tão importantes na minha formação. As aulas, o exemplo e o incentivo de vocês me ajudaram a construir o caminho que me trouxe até aqui e a tomar gosto pela profissão.

RESUMO

A elaboração de artefatos de especificação de software, especialmente diagramas UML, costuma ser reduzida ou negligenciada em contextos ágeis devido ao custo e ao tempo de produção manual, embora permaneça relevante para comunicação, alinhamento e documentação entre equipes. Em paralelo, modelos de processos de negócio em BPMN são amplamente utilizados nas organizações e carregam informações estruturadas (participantes, atividades e fluxos), que podem servir como ponto de partida para derivar especificações de software, conectando o que o negócio descreve ao que o sistema deve entregar. Neste trabalho, propõe-se uma metodologia para mapear modelos BPMN em artefatos de especificação de software com apoio de Inteligência Artificial (IA), aplicando regras consolidadas na literatura e adaptando-as para instruções claras em formato de prompts. A metodologia é estruturada em quatro etapas: seleção e análise da literatura-base; sistematização e adaptação das regras; estudo de caso com um BPMN distinto dos exemplos originais; e análise comparativa dos resultados. No estudo de caso, os prompts foram aplicados em ferramentas de IA (ChatGPT, Gemini e Claude) para gerar Diagrama de Casos de Uso e suas descrições textuais, Diagrama de Classes e Diagrama de Sequência, e os resultados foram comparados a uma aplicação manual das regras por meio de um checklist objetivo. Foram realizadas 50 execuções (10 por modelo), e, no resultado geral, obteve-se o ranking: ChatGPT Plus > Claude Free > Gemini Pro > ChatGPT Free > Gemini Free. Observou-se que as principais perdas não ocorreram em regras diretas, mas em regras que exigem rastreabilidade fina e coerência global entre participantes e fluxos. Como produto final, a pesquisa entrega recomendações de ferramentas e de estratégias de prompting para o mapeamento BPMN-UML, considerando eficiência, acurácia e agilidade, com foco em apoiar empresas e especialistas na adoção prática da abordagem.

Palavras-chave: BPMN. UML. especificação de software. mapeamento de modelos. inteligência artificial.

ABSTRACT

Software specification artifacts, especially UML diagrams, are often reduced or neglected in agile contexts due to the cost and time required for manual production, although they remain relevant for communication, alignment, and documentation across teams. In parallel, BPMN business process models are widely used in organizations and contain structured information (participants, activities, and flows) that can serve as a starting point to derive software specifications, bridging what the business describes and what the software must deliver. This work proposes a methodology to map BPMN models into software specification artifacts with the support of Artificial Intelligence (AI), applying mapping rules consolidated in the literature and adapting them into clear prompt-based instructions. The methodology is structured into four stages: selection and analysis of the baseline literature; systematization and adaptation of the rules; a case study using a BPMN model different from the original examples; and comparative analysis of the results. In the case study, prompts were applied to AI tools (ChatGPT, Gemini, and Claude) to generate a Use Case Diagram and its textual descriptions, a Class Diagram, and a Sequence Diagram, and the outputs were compared against a manual application of the rules using an objective checklist. A total of 50 runs (10 per model) were performed, and the overall ranking was: ChatGPT Plus > Claude Free > Gemini Pro > ChatGPT Free > Gemini Free. The main quality losses were not associated with straightforward rules, but with rules that require fine-grained traceability and global consistency across participants and flows. As a final outcome, the research provides recommendations on tools and prompting strategies for BPMN-to-UML mapping, considering efficiency, accuracy, and speed, with the goal of supporting companies and practitioners in the practical adoption of this approach.

Keywords: BPMN. UML. software specification. model mapping. artificial intelligence.

LISTA DE ILUSTRAÇÕES

1	Visão geral do fluxo da metodologia.	36
2	Visão geral do fluxo do estudo de caso.	52
3	Modelo BPMN do processo de agendamento de consultas.	54
4	Subprocesso BPMN para processamento de pagamento particular.	55
5	Diagrama de Casos de Uso de referência (execução manual das regras).	55
6	Diagrama de Classes de referência (execução manual das regras).	56
7	Diagrama de Sequência de referência (execução manual das regras).	57
8	DCU gerado na melhor execução do ChatGPT Plus.	68
9	DC gerado na melhor execução do ChatGPT Plus.	68
10	DS gerado na melhor execução do ChatGPT Plus.	69
11	DCU gerado na melhor execução do ChatGPT Free.	73
12	DC gerado na melhor execução do ChatGPT Free.	74
13	DS gerado na melhor execução do ChatGPT Free.	75
14	DCU gerado na melhor execução do Gemini Pro.	79
15	DC gerado na melhor execução do Gemini Pro.	79
16	DS gerado na melhor execução do Gemini Pro.	80
17	DC gerado na melhor execução do Gemini Free.	83
18	DCU gerado na melhor execução do Gemini Free.	84
19	DS gerado na melhor execução do Gemini Free.	85
20	DC gerado na melhor execução do Claude Free.	88
21	DCU gerado na melhor execução do Claude Free.	89
22	DS gerado na melhor execução do Claude Free.	90
23	Comparação das médias normalizadas (0–10) por artefato e por modelo de IA.	93

LISTA DE TABELAS

1	Principais elementos do BPMN 2.0.2.	21
2	Comparativo das abordagens da literatura de base.	34
3	Critérios de Seleção para a literatura de base.	39
4	Análise detalhada da seleção de artigos da RSL.	39
5	Regras mestras para o mapeamento BPMN-UML.	42
6	Ferramentas de IA utilizadas e modelos.	58
7	Checklist de perguntas utilizado na avaliação das regras de mapeamento. . . .	63
8	Pontuação do ChatGPT Plus por regra e por execução.	66
9	Execuções do ChatGPT Plus e respectivas configurações.	70
10	Pontuação do ChatGPT Free por regra e por execução.	71
11	Execuções do ChatGPT Free e respectivas configurações.	76
12	Pontuação do Gemini Pro por regra e por execução.	77
13	Execuções do Gemini Pro e respectivas configurações.	81
14	Pontuação do Gemini Free por regra e por execução.	81
15	Execuções do Gemini Free e respectivas configurações.	86
16	Pontuação do Claude Free por Regra e por Execução.	87
17	Execuções do Claude Free e respectivas configurações.	91
18	Comparação geral das médias por regra, diagrama e modelo de IA.	91
19	Estratégias de entrada recomendadas por modelo de IA.	95

LISTA DE ABREVIACÕES E SIGLAS

UML	<i>Unified Modeling Language</i>
BPM	<i>Business Process Management</i>
BPMN	<i>Business Process Model and Notation</i>
ABPMP	<i>Association of Business Process Management Professionals</i>
BPM CBOK	<i>Guide to the Business Process Management Body of Knowledge</i>
OMG	<i>Object Management Group</i>
MDE	<i>Model Driven Engineering</i>
MDA	<i>Model Driven Architecture</i>
IA	Inteligência Artificial
EPC	<i>Event-driven Process Chain</i>
7PMG	<i>Seven Process Modeling Guidelines</i>
LLM	<i>Large Language Model</i>
GPT	<i>Generative Pre-trained Transformer</i>
RSL	Revisão Sistemática da Literatura
DCU	Diagrama de Casos de Uso
DT	Descrição Textual
DC	Diagrama de Classes
DS	Diagrama de Sequência
QVT	<i>Query/View/Transformation</i>
CI	Critério de Inclusão

SUMÁRIO

1	INTRODUÇÃO	13
1.1	JUSTIFICATIVA	15
1.2	OBJETIVOS	16
1.3	METODOLOGIA DE PESQUISA	16
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	PROCESSO DE NEGÓCIO	18
2.1.1	Modelos de Processo de Negócio	18
2.1.2	<i>Business Process Management</i> (BPM)	19
2.1.3	<i>Business Process Model and Notation</i> (BPMN)	20
2.2	UML E METODOLOGIAS ÁGEIS	23
2.2.1	<i>Unified Modeling Language</i> (UML)	23
2.2.2	Metodologias Ágeis	24
2.2.3	O uso da UML no contexto ágil	24
2.3	SOFTWARES DE MODELAGEM	26
2.3.1	Draw.io	26
2.3.2	PlantUML	26
2.3.3	Mermaid	27
2.3.4	Bizagi	27
2.4	INTELIGÊNCIA ARTIFICIAL NA ESPECIFICAÇÃO DE SOFTWARE .	28
2.4.1	Inteligência Artificial Generativa	28
2.4.2	Aplicações na Engenharia de Software	28
2.4.3	Engenharia de <i>Prompts</i> para especificação de requisitos	29
3	TRABALHOS RELACIONADOS	31
3.1	ANÁLISE DA LITERATURA DE BASE	31
3.2	SÍNTESE E RELAÇÃO COM A PESQUISA	34

4	METODOLOGIA	36
4.1	CLASSIFICAÇÃO DA PESQUISA	36
4.2	SELEÇÃO DA LITERATURA DE BASE	37
4.2.1	Busca exploratória inicial	37
4.2.2	Seleção da literatura consolidada.....	38
4.2.3	Aplicação dos Critérios de Inclusão	39
4.3	SISTEMATIZAÇÃO E ADAPTAÇÃO DAS REGRAS PARA IA	41
4.3.1	Sistematização das Regras de Mapeamento.....	42
4.3.2	Adaptação das Regras para <i>Prompts</i>	44
4.4	ESTUDO DE CASO	51
4.4.1	Seleção e preparação do BPMN.....	52
4.4.2	Aplicação manual das regras	55
4.4.3	Execução do prompt e geração dos diagramas	58
5	ANÁLISE DOS RESULTADOS	61
5.1	SISTEMA DE MÉTRICAS	61
5.2	CHATGPT PLUS	66
5.2.1	Visão geral e estabilidade	66
5.2.2	Desempenho por artefato	67
5.2.3	Principais dificuldades por regra	67
5.2.4	Melhor execução e exemplos visuais.....	67
5.2.5	Influência do formato de entrada.....	70
5.3	CHATGPT FREE	71
5.3.1	Visão geral e estabilidade	72
5.3.2	Desempenho por artefato	72
5.3.3	Principais dificuldades por regra	72
5.3.4	Melhor execução e exemplos visuais.....	72
5.3.5	Influência do formato de entrada.....	76
5.4	GEMINI PRO	77
5.4.1	Visão geral e estabilidade	78

5.4.2	Desempenho por artefato	78
5.4.3	Principais dificuldades por regra	78
5.4.4	Melhor execução e exemplos visuais.....	79
5.4.5	Influência do formato de entrada.....	80
5.5	GEMINI FREE	81
5.5.1	Visão geral e estabilidade	82
5.5.2	Desempenho por artefato	83
5.5.3	Principais dificuldades por regra	83
5.5.4	Melhor execução e exemplos visuais.....	83
5.5.5	Influência do formato de entrada.....	86
5.6	CLAUDE FREE	87
5.6.1	Visão geral e estabilidade	87
5.6.2	Desempenho por artefato	88
5.6.3	Principais dificuldades por regra	88
5.6.4	Melhor execução e exemplos visuais.....	88
5.6.5	Influência do formato de entrada.....	90
5.7	COMPARAÇÃO GLOBAL	91
5.7.1	Comparação por artefato.....	93
5.7.2	Comparação por regra	93
5.8	IMPACTO DO FORMATO DE ENTRADA NAS EXECUÇÕES	94
6	CONCLUSÃO	96
APÊNDICES		101
A	ANEXO A – Descrição Textual dos Casos de Uso (Geração Manual)	102
B	ANEXO B – Descrição Textual dos Casos de Uso (ChatGPT Plus)	107
C	ANEXO C – Descrição Textual dos Casos de Uso (ChatGPT Free)	110

D	ANEXO D – Descrição Textual dos Casos de Uso (Gemini Pro) . 114
E	ANEXO E – Descrição Textual dos Casos de Uso (Gemini Free) . 117
F	ANEXO F – Descrição Textual dos Casos de Uso (Claude Free) . 120

1 INTRODUÇÃO

O desenvolvimento de sistemas de software envolve diversas etapas, que vão desde a compreensão dos processos de negócio do cliente até a implementação das funcionalidades. Dentro desse contexto, a elaboração do documento de especificação de requisitos de software é uma etapa essencial, mesmo no atual cenário de metodologias ágeis, pois o processo de documentação detalha as necessidades do sistema, atuando como um guia claro para desenvolvedores e demais *stakeholders*.

A modelagem de requisitos é uma prática indispensável para a engenharia de software contemporânea, pois, como defendem Pressman e Maxim (2021), o modelo de análise funciona como uma ponte para o projeto do software e, ao mesmo tempo, oferece os meios para que tanto o desenvolvedor quanto o cliente possam avaliar a qualidade do produto final. Para isso, o uso de uma linguagem gráfica padronizada contribui para uma comunicação eficaz entre os envolvidos, reduzindo ambiguidades e facilitando o alinhamento de expectativas. Foi criada, então, a *Unified Modeling Language* (UML), que, conforme destacam Bernhardt *et al.* (2023, p.1), a linguagem “emerge como uma ferramenta valiosa e padronizada para especificar, visualizar, construir e documentar sistemas de software”. Segundo John (2025, p.1, tradução nossa), o papel da UML é fundamental para “facilitar a comunicação entre as partes interessadas, aprimorar o *design* do sistema e melhorar a qualidade da documentação”¹. A importância dessa prática é reforçada por Fowler (2014, p.25), para quem o principal motivo de existirem linguagens como a UML é que “as linguagens de programação não estão em um nível de abstração suficientemente alto para facilitar as discussões sobre projeto”.

No entanto, a modelagem manual de artefatos UML pode ser complexa e onerosa, exigindo tempo, ferramentas adequadas e domínio técnico. Em metodologias ágeis, que valorizam entregas rápidas, essa prática acaba sendo negligenciada em muitas equipes. Conforme aponta um estudo de Souza *et al.* (2024, p.1), “os processos ágeis argumentam sobre o uso de uma documentação mínima, utilizando histórias de usuários

¹Texto original: “*facilitating communication among stakeholders, enhancing system design, and improving documentation quality.*”

e desestimulando o uso de documentos de especificação longos e complexos, sendo a sua preferência a comunicação face-a-face”. Embora os métodos ágeis não sejam contrários à documentação, essa ênfase em artefatos mais leves pode resultar em falhas na comunicação e em retrabalho durante o desenvolvimento, como advertem Pressman e Maxim (2021).

Em paralelo, os modelos de processo de negócio, especialmente os representados em *Business Process Model and Notation* (BPMN), são amplamente utilizados pelas empresas para descrever seus fluxos operacionais, atuando como uma “ponte padronizada para a lacuna entre o desenho do processo de negócio e a sua implementação”² (Object Management Group (OMG), 2014, p.1, tradução nossa). Esses modelos contêm informações valiosas que podem servir como ponto de partida para a geração de artefatos de especificação de software, alinhando-se a práticas de *Model Driven Engineering* (MDE), nas quais “o modelo de software é considerado o núcleo das atividades de desenvolvimento”³ (Shaikh *et al.*, 2021, p.1, tradução nossa). Além disso, “a validação do modelo garante que ele esteja livre de erros”⁴ (Shaikh *et al.*, 2021, p.10, tradução nossa), o que aumenta a confiabilidade dos artefatos gerados e reduz problemas que poderiam ser transferidos para etapas posteriores do desenvolvimento.

No contexto de implementação e automação, uma categoria de ferramentas comumente associada ao BPM é a *Business Process Management Suite* (BPMS), um mecanismo que reúne componentes essenciais para suportar todo o ciclo de vida do gerenciamento de processos, incluindo modelagem, execução, monitoramento e melhoria contínua (Association of Business Process Management Professionals (ABPMP), 2013). Em termos práticos, elas permitem que modelos desenhados em BPMN sejam convertidos em fluxos executáveis de automação, conectando tarefas humanas e sistemas de informação. Um exemplo de solução corporativa desse tipo é o Sydle One, uma plataforma de BPMS desenvolvida pela empresa brasileira Sydle, que integra modelagem de processos, execução e automação em um único ambiente (SYDLE, 2025). Apesar de demonstrarem a viabilidade de usar modelos de processo como base para implementação, tais soluções

²Texto original: “*standardized bridge for the gap between the business process design and process implementation*”.

³Texto original: “*the software model is considered the nucleus of software development activities*”.

⁴Texto original: “*Model validation ensures that the model is error-free*”.

tendem a ter custo elevado e não são, em geral, de domínio público, estando restritas a grandes organizações e inacessíveis a muitos profissionais e equipes menores.

A literatura da área descreve algumas metodologias e regras para realizar o mapeamento de modelos BPMN em artefatos de especificação de software, em especial diagramas UML (Santander *et al.*, 2022; Chagas *et al.*, 2024). Entretanto, a maioria dessas abordagens ainda é manual ou semiautomática, exigindo conhecimento técnico especializado e tempo considerável para execução. Nesse contexto, os avanços recentes das ferramentas de Inteligência Artificial (IA) abrem caminho para a evolução desses trabalhos por meio da automação desse processo, oferecendo maior rapidez e menor custo operacional. Essa abordagem pode contribuir significativamente para a produtividade, padronização e acessibilidade da modelagem em projetos de software, aproveitando o potencial da IA de apoiar tarefas tradicionalmente trabalhosas.

1.1 JUSTIFICATIVA

Dado que os modelos BPMN capturam os fluxos de trabalho e as regras de negócio, eles funcionam como uma fonte rica de requisitos para o sistema de software. A engenharia de software, por sua vez, utiliza a UML para especificar como esses requisitos serão implementados. Estabelecer uma ponte direta entre esses dois padrões é, portanto, uma prática que favorece a rastreabilidade e a consistência entre o que o negócio exige (BPMN) e o que o software especifica (UML). Essa abordagem também é favorecida por reutilizar o esforço de modelagem de negócio e por reduzir ambiguidades ou falhas de interpretação, que podem ocorrer quando artefatos UML são criados manualmente e de forma independente. Entretanto, a aplicação sistemática de regras de mapeamento ainda tende a depender de execução manual ou semiautomática, o que demanda tempo e conhecimento especializado.

Esse trabalho se justifica, primeiro, pela necessidade de consistência entre processos de negócio e especificação de software, que pode ser alcançada pelo mapeamento BPMN-UML; e, segundo, pela oportunidade de evoluir o estado da arte, investigando o uso de ferramentas de Inteligência Artificial para automatizar a aplicação de regras de

mapeamento já propostas e validadas na literatura. Ao invés de criar novas regras do zero, a pesquisa se concentra em adaptar e aplicar o conhecimento existente por meio da IA, avaliando se tais técnicas podem ser automatizadas sem perda de qualidade. Com isso, busca-se desenvolver um método acessível que possa democratizar essa prática, tornando-a mais viável e integrada ao cotidiano de equipes de diferentes portes.

1.2 OBJETIVOS

Considerando o desafio de otimizar a criação de artefatos de especificação, este trabalho tem como **objetivo geral** propor uma metodologia de mapeamento de modelos de processos de negócio em artefatos de especificação de software, utilizando Inteligência Artificial para aplicar regras e métodos já consolidados na literatura.

Os objetivos específicos incluem:

1. Consolidar, a partir da literatura, um **conjunto de regras mestras** de mapeamento de BPMN para UML, organizadas por artefato alvo.
2. Desenvolver um **conjunto de *prompts* e diretrizes de execução** que operacionalize as regras mestras em ferramentas de IA, especificando o formato de entrada e as estratégias de envio utilizadas no estudo.
3. Produzir um **artefato de referência** para comparação, obtido pela aplicação manual das regras mestras ao modelo BPMN do estudo de caso.
4. Fornecer um **sistema de métricas** para verificar aderência e consistência dos resultados, com potencial de reuso em novos estudos e em cenários práticos.
5. Realizar uma **comparação quantitativa de desempenho** entre diferentes ferramentas e modelos de IA, reportando resultados por regra e por artefato no estudo de caso.

1.3 METODOLOGIA DE PESQUISA

O processo metodológico empregado foi estruturado em quatro etapas principais, que descrevem a seleção da literatura de base, a sistematização das regras, a condução do estudo de caso e a avaliação dos resultados.

Na primeira etapa foi realizada a seleção e análise da literatura de base. Em vez de conduzir uma nova revisão da literatura, optou-se estrategicamente por partir de um

estudo consolidado: o artigo “Obtendo Casos de Uso a Partir de Modelos de Processos de Negócio: Uma Revisão Sistemática da Literatura” (Santander *et al.*, 2022). A partir dos 22 artigos analisados nessa revisão, foi feito um processo de filtragem, aplicando critérios de inclusão e exclusão para selecionar apenas os trabalhos que utilizam BPMN e que apresentam regras de mapeamento passíveis de automação por IA.

A segunda etapa consistiu na sistematização e adaptação das regras extraídas dos artigos selecionados, procedimento que é descrito em detalhes na Seção 4.3. As metodologias e diretrizes propostas na literatura foram analisadas, sintetizadas e reestruturadas para servirem como instruções claras e eficazes para ferramentas de Inteligência Artificial.

Na terceira etapa, foi conduzido um estudo de caso prático. Para isso, utilizou-se um modelo de processo de negócio em BPMN diferente daqueles empregados nos artigos originais, a fim de reduzir vieses e favorecer uma avaliação mais imparcial da metodologia. Esse modelo foi usado como *input* para as ferramentas de IA (ChatGPT e Gemini). Os *prompts* desenvolvidos na etapa anterior foram aplicados para gerar os artefatos de especificação de software, como diagramas de caso de uso, diagramas de classes e descrições textuais.

Por fim, na quarta etapa, foi realizada a análise dos resultados. Os artefatos gerados pela IA foram comparados com os resultados esperados, com base na aplicação manual das regras da literatura. Essa análise comparativa permitiu avaliar a qualidade de saída da IA em termos de aderência às diretrizes e à consistência com os resultados esperados.

Diante do cenário apresentado, a pesquisa se propõe a responder às seguintes questões:

- **RQ1:** É possível gerar regras de mapeamento úteis para que as ferramentas e modelos de IA entendam e transformem modelos BPMN em artefatos de especificação de software, incluindo diagramas UML?
- **RQ2:** Como diferentes ferramentas e modelos de IA se comportam ao receber regras de mapeamento BPMN–UML, no que se refere ao entendimento de conceitos da área, tempo de execução e qualidade dos artefatos gerados?
- **RQ3:** Qual é o grau de similaridade e a consistência dos artefatos gerados pelas ferramentas e modelos de IA em relação aos mesmos artefatos gerados manualmente a partir das regras elaboradas?

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos teóricos que servem como embasamento para esta pesquisa, estruturando-se em três pilares principais. Inicialmente, é detalhado o domínio da Modelagem de Processos de Negócio, com foco na notação BPMN, que constitui o artefato de entrada para a metodologia proposta. Em seguida, são abordados os conceitos da UML no contexto das Metodologias Ágeis, contextualizando o debate sobre o papel da documentação em projetos modernos. Por fim, a última seção explora o campo da Inteligência Artificial Generativa e as técnicas de Engenharia de *Prompts*, que são as ferramentas centrais utilizadas para a automação neste trabalho.

2.1 PROCESSO DE NEGÓCIO

Para que um sistema de software seja desenvolvido, é fundamental que, antes das automações, o próprio negócio seja compreendido, para que, depois, o sistema reflita as operações e as regras da organização. A modelagem de processos surge como a principal ferramenta para essa compreensão. Conforme defendem Dumas *et al.* (2018), um processo de negócio é um conjunto de atividades logicamente relacionadas que agregam valor a uma organização e a seus clientes. A modelagem, portanto, é a representação gráfica desse fluxo de trabalho, tornando-o explícito e compreensível.

2.1.1 Modelos de Processo de Negócio

Um processo de negócio descreve “**como** o trabalho é realizado dentro de uma organização, em contraste com o foco de um produto, que enfatiza **o que** é feito”¹ (Davenport, 1993, p.5, tradução nossa). Pode ser definido como uma associação entre eventos, atividades e decisões que, envolvendo múltiplos atores e recursos, levam a um resultado que agrega valor a um cliente (Dumas *et al.*, 2018). Quase todas as operações de uma empresa podem ser descritas como um processo de negócio.

No entanto, processos em ambientes corporativos raramente são simples, podendo

¹Texto original: “*how work is done within an organization, in contrast to a product focus’s emphasis on what*”.

envolver diversas áreas, sobrepor-se e conter diversas regras complexas. Diante disso, o modelo de processo de negócio emerge como uma ferramenta valiosa para a gestão dessa complexidade. Weske (2019) define um modelo de processo de negócio como uma representação de um processo, geralmente gráfica, que descreve as atividades, seus executores, a ordem lógica das tarefas e os fluxos de dados envolvidos.

O objetivo principal da modelagem não é apenas documentar, mas também analisar, otimizar e, eventualmente, automatizar (Weske, 2019). Um modelo claro serve como uma ótima ferramenta de comunicação, alinhando a visão de analistas de negócio, gestores e equipes de TI. Ele permite que a organização identifique gargalos, visualize oportunidades de melhoria e crie uma base sólida para a automação, garantindo que o software a ser desenvolvido esteja alinhado às necessidades operacionais.

2.1.2 *Business Process Management* (BPM)

Fazer o desenho de um processo, isto é, a modelagem dele, não é uma atividade isolada, mas sim uma das etapas centrais de uma filosofia muito mais ampla, conhecida como *Business Process Management* (BPM) ou Gerenciamento de Processos de Negócio. O BPM é uma abordagem sistemática e disciplinada para identificar, desenhar, executar, monitorar e melhorar processos de negócio, visando alcançar os objetivos estratégicos da organização (Dumas *et al.*, 2018; Weske, 2019).

Mais do que apenas um conjunto de ferramentas, o BPM é um ciclo de vida contínuo. A associação de profissionais da área, *Association of Business Process Management Professionals* (ABPMP), em seu Guia de Corpo Comum de Conhecimento para Gerenciamento de Processos de Negócio, ou *Guide to the Business Process Management Body of Knowledge* (BPM CBOK), estrutura esse ciclo em fases que tipicamente incluem: planejamento, análise, desenho, implementação, monitoramento e refinamento. Porém, independentemente dos nomes ou do número de fases adotado, a lógica deve refletir o princípio fundamental de melhoria iterativa do ciclo PDCA (*Plan, Do, Check, Act*), também conhecido como ciclo de Deming (Association of Business Process Management Professionals (ABPMP), 2013, p.52-53).

Esta pesquisa se situa exatamente na transição entre as fases de desenho e implementação, citadas pela ABPMP. O BPM defende que os modelos de processo, isto é, a saída da fase de desenho, não devem ser apenas documentos estáticos, mas sim artefatos vivos, que servem como base para a implementação do software que irá automatizar ou apoiar a execução desse processo. Nesse contexto, a escolha de uma notação padronizada se torna crucial, o que justifica o foco na BPMN.

2.1.3 *Business Process Model and Notation* (BPMN)

Historicamente, diversas notações foram utilizadas para modelar processos, como fluxogramas tradicionais e *Event-driven Process Chains* (EPCs) (Weske, 2019; Dumas *et al.*, 2018). No entanto, essas notações frequentemente careciam de uma semântica formal ou eram complexas demais para gestores de negócio, ou simples demais para equipes de TI. Para unificar a representação de processos, o *Object Management Group* (OMG), o mesmo grupo que padroniza a UML, publicou a especificação do *Business Process Model and Notation* (BPMN).

Conforme definido pelo próprio Object Management Group (OMG) (2014, p.1, tradução nossa), “o objetivo principal do BPMN é fornecer uma notação que seja facilmente compreensível por todos os usuários de negócios, desde os analistas que criam os rascunhos iniciais dos processos até os desenvolvedores técnicos responsáveis por implementar a tecnologia que os executará”². A notação visa, portanto, ser a “ponte padronizada para a lacuna entre o desenho do processo de negócio e a sua implementação”³, conforme citado na introdução deste trabalho.

Para alcançar essa dualidade de ser simples para negócios e, ao mesmo tempo, rico para os desenvolvedores, o BPMN 2.0.2 se estrutura em um conjunto de elementos gráficos, separados em cinco categorias principais: Objetos de Fluxo (*Flow Objects*), Dados (*Data*), Objetos de Conexão (*Connecting Objects*), Raias (*Swimlanes*) e Artefatos

²Texto original: “*The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes*”.

³Texto original: “*standardized bridge for the gap between the business process design and process implementation*”.

(*Artifacts*) (Object Management Group (OMG), 2014). A Tabela 1 detalha os elementos que compõem cada uma dessas categorias, apresentando suas respectivas descrições e notações gráficas.

Tabela 1: Principais elementos do BPMN 2.0.2.

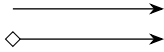
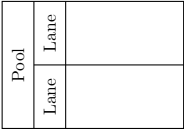
Categoria	Elemento	Descrição	Notação
Objetos de Fluxo	Evento	É algo que “acontece” durante o curso de um processo. Afeta o fluxo do modelo e geralmente tem uma causa (gatilho / <i>trigger</i>) ou um impacto (resultado). Representado por círculos com centros abertos para permitir que marcadores internos diferenciem diferentes gatilhos ou resultados. Existem 3 tipos: Início , Intermediário ou Fim .	
	Atividade	Representa as tarefas ou o trabalho que precisa ser executado no processo. Existem 2 tipos: Subprocesso e Tarefa .	
	<i>Gateway</i>	Utilizado para lidar com convergências e divergências condicionais de fluxo. Determina a ramificação, bifurcação, fusão e junção de caminhos. Marcadores internos indicam o tipo de controle de comportamento (Exclusivo , Inclusivo ou Paralelo).	
Objetos de Conexão	Fluxo de Sequência	Mostra a ordem em que as Atividades serão executadas. Pode ter uma expressão de condição, que é avaliada em tempo de execução, para determinar se o fluxo de Sequência será usado ou não (nesse caso, haverá um losango no início do conector).	
	Fluxo de Mensagem	Mostra o fluxo de Mensagens entre dois participantes que estão preparados para enviá-las e recebê-las.	
	Associação	Usada para vincular informações e artefatos com elementos gráficos BPMN. Anotações de Texto e outros artefatos podem ser associados aos elementos gráficos. Uma ponta de seta na Associação indica uma direção de fluxo, quando apropriado.	

Tabela 1: Principais elementos do BPMN 2.0.2.

Categoria	Elemento	Descrição	Notação
Raias	<i>Pool</i>	Representação gráfica de um participante em uma colaboração. Atua como um contêiner gráfico para particionar um conjunto de Atividades de outros <i>Pools</i> . Usado para organizar e categorizar Atividades.	
	<i>Lane</i>	É uma subpartição dentro de um processo, às vezes dentro de um <i>Pool</i> . Pode ser vertical ou horizontal.	
Dados	Objeto de Dados	Fornece informações sobre o que as Atividades exigem para serem executadas e/ou o que elas produzem. Podem representar um objeto singular ou uma coleção de objetos. Entrada e Saída de Dados fornecem as mesmas informações para Processos.	Data Object  Data Object (Collection)  Data Input Data Output  
Artefatos	Grupo	Agrupamento de elementos gráficos que estão dentro da mesma categoria. Não afeta os Fluxos de Sequência dentro dele. O nome da categoria aparece no diagrama como o rótulo do grupo. Pode ser usado para fins de documentação ou análise.	
	Anotação de Texto	Mecanismo para que um modelador forneça informações textuais adicionais ao leitor do diagrama BPMN.	

Fonte: Adaptado de Object Management Group (OMG) (2014, p.26-38)

Contudo, a simples utilização desses elementos não garante um bom modelo. A literatura da área de BPM demonstra que a forma como o modelo é desenhado impacta diretamente sua compreensibilidade e utilidade (Dumas *et al.*, 2018). Para guiar a criação de modelos de alta qualidade, Mendling *et al.* (2010) propõem um conjunto de sete diretrizes, conhecidas como *Seven Process Modeling Guidelines* (7PMG), como “usar o mínimo possível de elementos no modelo”⁴ e “estruturar o máximo possível os fluxos do modelo”⁵, da esquerda para a direita e de cima para baixo, por exemplo.

⁴Texto original: “*Use as few elements in the model as possible*”.

⁵Texto original: “*Model as structured as possible*”.

A riqueza semântica do BPMN, quando aliada a essas boas práticas de modelagem, permite criar artefatos que carregam informações estruturadas sobre regras de negócio, responsabilidades e interações, sendo essenciais para a especificação de requisitos de software. É essa riqueza de modelos bem feitos que esta pesquisa busca explorar.

2.2 UML E METODOLOGIAS ÁGEIS

Após a etapa de compreensão dos processos de negócio, a engenharia de software tradicionalmente se volta para a especificação dos requisitos de software. Nesse contexto, a UML se consolidou como uma linguagem padrão para a documentação de artefatos. Contudo, a ascensão das metodologias ágeis trouxe uma nova filosofia de desenvolvimento, que questiona a necessidade de documentação extensa e formal, gerando um debate sobre o papel da modelagem em projetos modernos.

2.2.1 *Unified Modeling Language* (UML)

A *Unified Modeling Language* (UML) é uma linguagem gráfica padronizada para visualizar, especificar, construir e documentar os artefatos de um sistema de software (Bernhardt *et al.*, 2023). Conforme destacado no Capítulo 1 deste trabalho, o principal objetivo da UML é elevar o nível de abstração da discussão de um projeto para além do código-fonte, facilitando a comunicação entre as partes interessadas de um projeto (John, 2025).

Para que desenvolvedores e analistas possam modelar o sistema sob diferentes perspectivas, a UML fornece um conjunto de diagramas, cada um com um foco específico. Dentre eles, dois são particularmente relevantes para a especificação de requisitos funcionais e estruturais, foco deste trabalho:

- **Diagrama de Casos de Uso:** Descreve as funcionalidades do sistema do ponto de vista do usuário. Ele representa as interações entre os “atores” (usuários ou outros sistemas) e os “casos de uso” (as funcionalidades que o sistema oferece), sem se aprofundar em detalhes de implementação. Segundo Booch *et al.* (1996), seu

propósito é capturar os requisitos funcionais e apresentar uma visão externa do que o sistema deve fazer.

- **Diagrama de Classes:** Representa a estrutura estática de um sistema orientado a objetos. Ele detalha as classes, seus atributos, métodos e os relacionamentos de associação, herança e dependência que existem entre elas. Para Fowler (2014), este é um dos principais diagramas da documentação técnica, pois serve como base para o projeto detalhado e para a posterior implementação do software.

2.2.2 Metodologias Ágeis

As metodologias ágeis surgiram como uma resposta aos modelos de desenvolvimento tradicionais, como o modelo Cascata (*Waterfall*), que eram criticados por sua rigidez e lentidão (Pressman e Maxim, 2021). Em vez de um processo sequencial e com fases longas, a abordagem ágil preza por ciclos de desenvolvimento curtos e iterativos, com entregas frequentes de software funcional (Beck *et al.*, 2001).

A filosofia ágil foi consolidada em 2001 no “Manifesto para o Desenvolvimento Ágil de Software”. Nele, Beck *et al.* (2001) estabelecem quatro valores fundamentais, dentre os quais se destaca a valorização de “software em funcionamento mais que documentação abrangente”. Isso não significa a ausência de documentação, mas uma clara preferência por artefatos que agreguem valor imediato ao processo de desenvolvimento.

Conforme aponta o estudo de Souza *et al.* (2024), essa filosofia estimula o uso de documentação mínima, como histórias de usuários, e favorece a comunicação direta em detrimento de documentos de especificação complexos. O objetivo, como defendem os métodos ágeis, é manter o foco na entrega de valor ao cliente e ter a flexibilidade para responder a mudanças de requisitos ao longo do projeto (Pressman e Maxim, 2021; Beck *et al.*, 2001).

2.2.3 O uso da UML no contexto ágil

O aparente conflito entre a necessidade de modelagem e a filosofia ágil é um dos principais debates na engenharia de software contemporânea. De um lado, a UML oferece clareza e

uma base sólida para o *design* e, do outro, os métodos ágeis advertem sobre o risco de a diagramação detalhada atrasar a entrega de código funcional.

A complexidade envolvida na modelagem de sistemas com UML e a consequente baixa adesão em ambientes ágeis podem gerar lacunas na documentação de especificação de requisitos de software. Em muitos casos, a preferência por artefatos mais leves e pela comunicação direta reduz a produção de modelos detalhados, o que pode dificultar o alinhamento e a rastreabilidade ao longo do desenvolvimento. Ainda assim, mesmo em contextos ágeis, a UML mantém relevância como linguagem padronizada para apoiar a comunicação e o alinhamento entre as partes interessadas (Bernhardt *et al.*, 2023; John, 2025).

Um ponto central para esse debate é a posição do cientista Martin Fowler, que não apenas é autor de uma obra de referência sobre a linguagem (UML Essencial, 2014), mas também é um dos signatários do Manifesto Ágil (2001). O próprio autor soluciona essa aparente dicotomia, pois esclarece sua preferência: “como um dos autores do *Manifesto for Agile Software Development*, sou fã ardoroso das estratégias ágeis” (Fowler, 2014, p.51); e, ao mesmo tempo, defende o valor da modelagem, afirmando: “Quase sempre utilizo a UML para fazer esboços” (Fowler, 2014, p.29). Isso demonstra que o valor da UML não é questionado pelo movimento ágil, mas sim a forma como ela é aplicada.

Para formalizar essa aplicação ágil da modelagem, Fowler (2014, p. 26) propõe que a UML seja utilizada em dois modos principais. O primeiro é justamente a UML como esboço, em que os diagramas são usados de forma colaborativa e informal para explorar ideias e facilitar a comunicação. Esses esboços são rápidos, descartáveis e perfeitamente alinhados ao espírito ágil. O segundo modo é a UML como projeto detalhado. Nesse caso, os diagramas são artefatos complexos e precisos, que servem como especificação formal para a geração de código. Embora poderoso, este modo é frequentemente visto como lento e burocrático, sendo o principal alvo de críticas em ambientes ágeis.

Percebe-se, portanto, que, apesar da ênfase ágil em artefatos leves, a modelagem continua sendo uma ferramenta fundamental para a comunicação e o planejamento do software. O desafio, então, está em obter os benefícios da clareza e da padronização da

UML sem incorrer no alto custo e na rigidez do modo de projeto detalhado. É exatamente nesta lacuna que a presente pesquisa se insere, ao propor um método que automatiza a geração de artefatos de especificação, buscando unir a agilidade do processo com a robustez da modelagem.

2.3 SOFTWARES DE MODELAGEM

Para viabilizar a construção, edição e documentação dos modelos utilizados no estudo, foram selecionadas ferramentas com objetivos complementares: (i) uma ferramenta focada em *BPMN* para criação e exportação do processo (incluindo XML), e (ii) ferramentas para criação/edição de diagramas e artefatos de especificação (principalmente *UML*) de forma reprodutível e fácil de manter.

2.3.1 Draw.io

O Draw.io (também conhecido como diagrams.net) é um editor de diagramas de propósito geral, com foco em criação manual de diagramas (por exemplo, fluxogramas, organogramas, UML, ER e BPMN), oferecendo bibliotecas de formas e modelos prontos (diagrams.net (draw.io), 2025a,b). No contexto acadêmico, esse tipo de ferramenta é especialmente útil quando se precisa de ajustes finos de layout, padronização visual entre figuras e exportação rápida para inclusão no documento. Assim, o Draw.io é indicado como apoio para edições pontuais, refinamentos visuais e diagramação de figuras que exigem controle manual mais direto.

2.3.2 PlantUML

O PlantUML é uma ferramenta de geração de diagramas a partir de descrições textuais, permitindo criar diferentes tipos de diagramas UML de maneira reprodutível e com boa integração a controle de versão (PlantUML, 2025; PlantUML contributors, 2025). Essa abordagem “diagrama como código” é coerente com a necessidade de manter consistência e rastreabilidade, pois pequenas alterações no modelo podem ser registradas como mudanças no texto-fonte do diagrama, reduzindo retrabalho e facilitando revisões. Além disso, por

ser textual, o PlantUML favorece padronização de nomes e estruturas, alinhando-se ao uso de UML como linguagem de modelagem de referência em especificação de software (Fowler, 2014).

2.3.3 Mermaid

O Mermaid é uma ferramenta baseada em JavaScript que renderiza diagramas a partir de definições textuais inspiradas em Markdown (Mermaid contributors, 2025). Na prática, ele é muito utilizado para documentação técnica em repositórios e ambientes que suportam Markdown, pois permite manter diagramas próximos do texto explicativo e atualizá-los de forma incremental. Plataformas como o GitHub oferecem suporte à renderização de diagramas Mermaid em documentos Markdown, o que reforça seu uso como recurso de documentação viva (GitHub Docs, 2025). No escopo deste trabalho, o Mermaid é adequado como alternativa leve para diagramas em documentação e registros auxiliares, especialmente quando o fluxo de escrita e revisão ocorre junto a artefatos textuais.

2.3.4 Bizagi

O Bizagi Modeler é uma ferramenta voltada à modelagem e documentação de processos em BPMN, com suporte à criação de diagramas e colaboração, sendo amplamente utilizada como ferramenta de modelagem de processos (Bizagi, 2025b,a). Considerando que BPMN é um padrão mantido pela OMG e projetado para representar processos de negócio de forma compreensível para diferentes perfis de stakeholders (analistas, gestores e desenvolvedores), o uso do Bizagi favorece a produção do modelo BPMN com boa legibilidade e posterior exportação do conteúdo. Dessa forma, o Bizagi foi empregado como ferramenta principal para elaboração do BPMN do estudo de caso e para obtenção do modelo em formato exportável, que serve de base para a geração e avaliação dos artefatos UML.

2.4 INTELIGÊNCIA ARTIFICIAL NA ESPECIFICAÇÃO DE SOFTWARE

Conforme discutido, a criação manual de artefatos de especificação de software, como os diagramas UML, representa um desafio, especialmente em contextos que valorizam a agilidade. Diante desse cenário, os recentes avanços em Inteligência Artificial (IA), em particular na área de IA Generativa, apresentam uma oportunidade promissora para automatizar e otimizar tarefas tradicionalmente intelectuais e repetitivas.

2.4.1 Inteligência Artificial Generativa

A Inteligência Artificial Generativa é um campo da IA focado em criar modelos capazes de gerar conteúdo novo e original, como textos, imagens e códigos. Embora o objetivo deste trabalho não envolva se aprofundar nos complexos fundamentos teóricos dessa área, é essencial focar em sua aplicação prática no contexto da Engenharia de Software. As ferramentas centrais para essa aplicação são os Modelos de Linguagem de Grande Porte, ou *Large Language Models* (LLMs), como os da família *Generative Pre-trained Transformer* (GPT) e o Gemini (Zhao *et al.*, 2023).

Esses modelos são redes neurais profundas treinadas com volumes massivos de dados textuais e de código, o que lhes confere uma capacidade avançada de compreender, resumir, traduzir e gerar texto de forma coerente e relevante. A arquitetura *Transformer*, base da maioria dos LLMs modernos, permite que eles processem e relacionem palavras em longos trechos de texto, capturando nuances e dependências complexas da linguagem humana e de programação (Zhao *et al.*, 2023).

2.4.2 Aplicações na Engenharia de Software

A capacidade dos LLMs de processar e gerar informação estruturada encontrou um campo fértil na Engenharia de Software. As pesquisas e aplicações práticas já demonstram o potencial dessas ferramentas em diversas etapas do ciclo de vida do desenvolvimento de software.

Um levantamento recente sobre o tema cataloga as principais aplicações, que incluem: geração de código a partir de descrições em linguagem natural, tradução de código entre

diferentes linguagens de programação, depuração e sugestão de correções para *bugs*, e geração automática de documentação técnica (Zhang *et al.*, 2023). Essas aplicações reduzem o esforço manual dos desenvolvedores e permitem que eles se concentrem em tarefas de maior complexidade, como a arquitetura e o design do sistema.

2.4.3 Engenharia de *Prompts* para especificação de requisitos

A eficácia de um LLM para realizar uma tarefa específica não depende apenas do modelo em si, mas, principalmente, da qualidade da instrução que ele recebe. A prática de projetar e refinar essas instruções é conhecida como Engenharia de *Prompts*. Um *prompt* é a entrada textual fornecida ao modelo, que contém a descrição da tarefa, o contexto, as restrições e o formato de saída desejado (White *et al.*, 2023).

No contexto da especificação de requisitos, a engenharia de *prompts* se torna a ferramenta-chave para traduzir o conhecimento humano em instruções que a máquina possa executar. As regras de mapeamento entre modelos de negócio e artefatos de software, já consolidadas na literatura, podem ser sistematizadas e adaptadas para compor um conjunto de *prompts* bem estruturados.

A eficácia de um *prompt*, no entanto, vai além de uma simples pergunta. Para tarefas complexas, como a transformação de modelos, a engenharia de *prompts* emprega diversas técnicas para refinar a instrução e guiar o modelo a um resultado mais preciso. A literatura da área já cataloga essas técnicas como “padrões de *prompt*” (*prompt patterns*), que servem como receitas reutilizáveis para otimizar a comunicação com o LLM (White *et al.*, 2023). Dentre as técnicas mais comuns, destacam-se:

- **Atribuição de Persona:** Instruir o modelo a atuar como um especialista (por exemplo, “Aja como um engenheiro de requisitos sênior”) para que ele adote o jargão, o raciocínio e o formato de saída esperados daquele papel.
- **Aprendizado com Poucos Exemplos (*Few-shot Prompting*):** Fornecer no próprio *prompt* um ou mais exemplos completos de uma entrada e da saída correspondente desejada. Isso ensina ao modelo, por demonstração, qual é o padrão esperado para a resposta.

- **Estruturação da Saída:** Definir explicitamente o formato que a resposta deve seguir, como JSON, Markdown, ou uma lista numerada, para facilitar o processamento automático do resultado.

A aplicação dessas técnicas é fundamental para traduzir as regras de mapeamento, extraídas da literatura, em um conjunto de instruções robustas. Isso garante que o LLM não apenas compreenda a tarefa, mas a execute com maior consistência e aderência ao método desejado, tornando a automação do processo mais confiável.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma análise aprofundada da literatura de base selecionada para esta pesquisa. Conforme será detalhado no Capítulo 4, esta seleção é o resultado da primeira etapa do processo metodológico, onde um conjunto de critérios de inclusão foi aplicado sobre a Revisão Sistemática da Literatura (RSL) de Santander *et al.* (2022). A análise a seguir não busca replicar a RSL, mas sim extrair e sintetizar o conteúdo essencial dos 5 artigos selecionados: as regras, heurísticas e metodologias propostas por eles. O objetivo é estabelecer o conjunto de conhecimento validado pela literatura que servirá como matéria-prima para a segunda etapa da metodologia, identificando, ao final, a lacuna de automação que esta pesquisa se propõe a preencher.

3.1 ANÁLISE DA LITERATURA DE BASE

Ao longo desta seção, os artigos da literatura de base são referenciados por IDs numéricos (ID 1, ID 4, ID 15, ID 18 e ID 21). Esses identificadores seguem a numeração dos estudos primários apresentada na Revisão Sistemática da Literatura de Santander *et al.* (2022). Por esse motivo, os IDs podem ser não consecutivos. Essa escolha mantém rastreabilidade direta entre a análise conduzida neste trabalho e a RSL adotada como referência.

O trabalho de Rodríguez *et al.* (2010) (**ID 1**) propõe uma abordagem alinhada à *Model Driven Architecture* (MDA) para gerar múltiplos artefatos de especificação, incluindo o **Diagrama de Casos de Uso** (DCU) e o **Diagrama de Classes** (DC). O diferencial da proposta é a definição de todas as regras de mapeamento utilizando a linguagem formal *Query/View/Transformation* (QVT), que é um padrão estabelecido pelo OMG, o mesmo grupo responsável pela UML e pelo BPMN, projetado especificamente para definir como um modelo-fonte pode ser transformado em um modelo-alvo (por exemplo, de BPMN para UML). A metodologia é executada em estágios: primeiramente, regras QVT são aplicadas para converter o modelo BPMN de entrada em um **Diagrama de Atividades** UML, que serve como um modelo intermediário. Nessa etapa, elementos como *Pools* e *Lanes* do BPMN são traduzidos para **ActivityPartitions** no Diagrama de Atividades, enquanto as Atividades são transformadas em **Ações**. Em seguida, um segundo conjunto

de regras QVT é aplicado sobre esse diagrama intermediário para gerar os modelos finais de especificação: as *ActivityPartitions* são convertidas em **Atores** no DCU e em **Classes** no DC, enquanto as Ações são mapeadas para **Casos de Uso** e **Operações** respectivamente.

Cruz *et al.* (2014) (ID 4) apresentam uma abordagem sistemática para derivar tanto o **Diagrama de Casos de Uso** quanto suas **especificações textuais** detalhadas. A proposta se destaca por ir além das regras diretas de mapeamento de diagrama (como *Pools* e *Lanes* para **Atores** ou Atividades para **Casos de Uso**) e por incorporar a extração da descrição textual dos casos. Para isso, os autores propõem tabelas de mapeamento e *templates* que utilizam outros elementos do BPMN para preencher os campos de cada Caso de Uso, como *Gateways* para definir **Pré-condições**, Eventos para identificar **Gatilhos**, e elementos de dados (Objetos de Dados, Associações e Mensagens) para compor o **Cenário**.

O trabalho de Khelif *et al.* (2018) (ID 15) também se baseia formalmente na MDA, abordagem que propõe a transformação de modelos. O ponto de partida é o CIM (*Computation Independent Model*, ou Modelo Independente de Computação), que descreve o “quê” será feito, sem se preocupar em “como” a computação será feita. O objetivo é gerar um PIM (*Platform Independent Model*, ou Modelo Independente de Plataforma), que descreve o “como” lógico do sistema (a arquitetura, os componentes), mas ainda sem depender de uma tecnologia específica. Para realizar essa transformação, os autores propõem a metodologia DESTINY (*a moDel-driven procESs-aware requiremenTs engineerINg methodologY*), que gera um modelo de análise UML (PIM) completo a partir de um modelo BPMN (CIM). A metodologia gera três artefatos: **Diagrama de Casos de Uso**, **Diagrama de Classes** e **Diagrama de Sequência**. Para os Casos de Uso, as regras mapeiam *Pools* para **Atores externos** e *Lanes* para **Atores internos**. Diferente de outras propostas, as Atividades não são mapeadas individualmente, mas sim agrupadas com base em seu objetivo funcional, formando um único **Caso de Uso**. No Diagrama de Classes, os Objetos de Dados do BPMN são mapeados para **Classes** UML, e suas associações são convertidas em relações entre essas classes. Por fim, no Diagrama de Sequência, as *Lanes* tornam-se **Linhas de Vida**, e os Fluxos de Sequência

entre Atividades de diferentes *Lanes* são convertidos em **Mensagens** trocadas entre essas linhas.

Expandindo o trabalho de Cruz *et al.* (2014), o trabalho de Cruz e da Cruz (2018) (ID 18) introduz um conjunto de heurísticas para derivar múltiplos artefatos: **Diagrama de Casos de Uso**, **Diagrama de Classes** e **Diagrama de Sequência**. Para o DCU, a abordagem define regras claras, como o mapeamento de *Pools* e *Lanes* para **Atores**. Uma regra central é o agrupamento de Tarefas consecutivas pertencentes à mesma *Lane* para formar um único **Caso de Uso**. O artigo se destaca por definir regras explícitas para os relacionamentos *«include»* e *«extend»*. Na proposta, sub-processos reutilizáveis são mapeados para *include* e *Gateways* que indicam caminhos alternativos são mapeados para *extend*. A metodologia também define regras para o Diagrama de Classes, mapeando Objetos de Dados são mapeados para **Classes** e *Lanes* (papéis internos) para **Classes UML**, enquanto as Tarefas dessas *Lanes* tornam-se **Operações** (métodos). No Diagrama de Sequência, as *Lanes* são representadas como **Linhas de Vida**, e os Fluxos de Mensagem e de Sequência entre *Lanes* distintas são mapeados como **Mensagens** trocadas entre essas Linhas de Vida.

Por fim, Bouzidi *et al.* (2017) (ID 21) apresentam uma abordagem de transformação de modelos baseada em MDA, utilizando a linguagem ATL (*ATLAS Transformation Language*), que é uma linguagem de transformação de modelos inspirada no padrão QVT e implementada sobre o *Eclipse Modeling Framework* (EMF). A proposta integra diferentes *plugins* para gerenciar os artefatos: o *EMF Eclipse BPMN2 Modeler* (para o BPMN), o *UML Designer* (para os **Diagramas de Casos de Uso**) e o *Acceleo* (para gerar as **descrições textuais**). Além das regras que mapeiam *Pools* e *Lanes* para **Atores** e sequências de Tarefas para **Casos de Uso**, o trabalho apresenta um diferencial notável: o tratamento da hierarquia de raias. Em vez de utilizar herança entre atores, a abordagem mapeia a estrutura hierárquica das raias do BPMN para uma hierarquia de **Pacotes** (*Packages*) UML que contêm os Atores correspondentes. O método também define regras para a derivação dos relacionamentos *«include»* e *«extend»* a partir da análise dos *Gateways* do processo, e gera automaticamente descrições textuais para os Casos de

Uso.

Para oferecer uma visão comparativa consolidada, a Tabela 2 resume as abordagens, os artefatos UML gerados e as principais contribuições de cada trabalho.

Tabela 2: Comparativo das abordagens da literatura de base.

ID (Autor)	Abordagem Principal	Artefatos UML Gerados				Principal Contribuição
		DCU	Texto	DC	DS	
ID 1 (Rodríguez <i>et al.</i>)	MDA com regras formais em QVT e uso de modelo intermediário (Diagrama de Atividades).	✓	×	✓	×	Define um processo de transformação formal em múltiplos estágios.
ID 4 (Cruz <i>et al.</i>)	Abordagem sistemática com <i>templates</i> e tabelas de mapeamento, com foco na descrição textual.	✓	✓	×	×	Foco na geração detalhada da especificação textual do Caso de Uso (Pré-condições, Gatilhos, Cenário).
ID 15 (Khelif <i>et al.</i>)	MDA (CIM p/ PIM) com a metodologia DESTINY.	✓	×	✓	✓	Geração de múltiplos diagramas de análise UML a partir de um único modelo BPMN.
ID 18 (Cruz e da Cruz)	Abordagem heurística (evolução do ID 4) para gerar múltiplos diagramas.	✓	×	✓	✓	Define regras explícitas para relacionamentos <i>«include»</i> e <i>«extend»</i> .
ID 21 (Bouzidi <i>et al.</i>)	MDA com linguagem ATL no <i>framework</i> Eclipse (EMF).	✓	✓	×	×	Propõe o mapeamento da hierarquia de raias para Pacotes UML, evitando herança entre atores.

3.2 SÍNTESE E RELAÇÃO COM A PESQUISA

A análise conjunta dos cinco trabalhos selecionados revela um conjunto robusto de regras e heurísticas para a transformação de modelos, demonstrando a viabilidade de gerar artefatos UML a partir de modelos BPMN. Observa-se uma forte convergência quanto à forma como os elementos de processo informam os elementos de requisitos. Regras como o mapeamento de *Pools* e *Lanes* para **Atores**, o agrupamento de Atividades ou Tarefas para formar **Casos de Uso**, e a utilização de *Gateways* para definir relacionamentos

como «*include*» e «*extend*» são validadas por múltiplas abordagens. Da mesma forma, a derivação de **Diagramas de Classe** a partir de Objetos de Dados e a geração de **Diagramas de Sequência** a partir do fluxo entre *Lanes* são práticas consolidadas na literatura de base.

Contudo, a principal lacuna identificada em todos esses trabalhos é a dependência de execução manual ou de alta complexidade técnica para a automação. Conforme demonstrado na Tabela 3, todos os cinco artigos analisados foram classificados como “Passíveis de Automação”, mas nenhum oferece uma solução automatizada acessível. As soluções propostas, embora robustas, dependem de linguagens de transformação de modelos (como QVT e ATL) que exigem conhecimento técnico especializado e o uso de ambientes de desenvolvimento complexos (como o Eclipse EMF). Outras abordagens, como a de Cruz e da Cruz (2018), reconhecem a necessidade de intervenção do analista para resolver ambiguidades, o que limita a automação completa e contrasta com a demanda atual por agilidade.

É precisamente nesta lacuna que a presente pesquisa se insere. Este trabalho não busca criar novas regras de mapeamento, mas sim evoluir o estado da arte ao aplicar **Inteligência Artificial Generativa** para automatizar a aplicação das regras já validadas pela literatura analisada neste capítulo. O objetivo é verificar se a IA, guiada por princípios de **Engenharia de *Prompts***, conforme detalhado no Capítulo 2, pode atuar como um motor de transformação acessível e de baixo custo, conectando as contribuições teóricas da literatura às necessidades práticas de agilidade no desenvolvimento de software.

4 METODOLOGIA

Este capítulo apresenta em detalhes o percurso metodológico adotado para o desenvolvimento desta pesquisa, que foi brevemente introduzido no Capítulo 1. As seções a seguir descrevem a classificação desta pesquisa e também as quatro etapas principais que foram executadas: (1) a seleção e análise da literatura de base, (2) a sistematização e adaptação das regras de mapeamento para a Inteligência Artificial, (3) a condução do estudo de caso prático e, por fim, (4) a avaliação dos resultados. A Figura 1 apresenta uma visão geral do fluxo da metodologia.



Figura 1: Visão geral do fluxo da metodologia.

4.1 CLASSIFICAÇÃO DA PESQUISA

Na presente seção, é feita a caracterização desta pesquisa quanto à sua tipologia. Para tanto, foram consultadas referências clássicas da literatura em metodologia científica (Lakatos e Marconi, 2017; Gil, 1991; Creswell, 2003), que serviram de base para definir a classificação adotada.

De acordo com os critérios propostos por Lakatos e Marconi (2017), uma pesquisa pode ser analisada segundo seus objetivos, natureza, procedimentos técnicos e forma de abordagem. Assim, esta investigação é classificada como exploratória, aplicada, bibliográfica e qualitativa, conforme descrito a seguir.

Quanto aos *objetivos*, a pesquisa é classificada como **exploratória**, pois busca proporcionar maior familiaridade com o problema e facilitar sua delimitação, investigando um tema ainda em consolidação na literatura: o mapeamento entre modelos BPMN e UML. Essa abordagem permitiu identificar e compreender as principais propostas existentes e os fundamentos que as sustentam.

Em relação à *natureza*, caracteriza-se como uma pesquisa **aplicada**, uma vez que visa gerar uma metodologia de mapeamento entre modelos BPMN e UML, contribuindo de forma prática para a Engenharia de Requisitos e a Engenharia de Software. Embora fundamentada em trabalhos anteriores, a proposta apresenta elementos de originalidade ao integrar conceitos e técnicas extraídos da literatura em uma solução própria.

Quanto aos *procedimentos técnicos*, trata-se de uma pesquisa **bibliográfica**, desenvolvida com base em materiais científicos já publicados que serviram de suporte para o embasamento teórico e para a análise comparativa das abordagens identificadas.

Por fim, quanto à *forma de abordagem* do problema, adota-se uma abordagem **qualitativa**, pois o estudo é conduzido de maneira descritiva e interpretativa, buscando compreender as diferentes abordagens sobre o tema e extrair significados e relações a partir da análise conceitual da literatura, sem recorrer a medições numéricas ou análises estatísticas.

Dessa forma, a classificação da pesquisa fundamenta as decisões metodológicas adotadas e orienta as etapas subsequentes de análise e desenvolvimento do artefato proposto.

4.2 SELEÇÃO DA LITERATURA DE BASE

A primeira etapa da metodologia consistiu na seleção e análise da literatura de base. Conforme definido após a pesquisa inicial, o foco da pesquisa está no mapeamento de BPMN para artefatos de especificação. Para otimizar o esforço de pesquisa e evitar a realização de uma nova Revisão Sistemática da Literatura (RSL) do zero, optou-se estrategicamente por partir de um estudo consolidado e atual (recente).

4.2.1 Busca exploratória inicial

Seguindo a classificação exploratória e bibliográfica, foi conduzida uma busca ampla para responder à questão: “Quais modelos ou diagramas são usados atualmente na especificação de requisitos de software?”. Para isso, foi utilizada a seguinte *string* de busca na plataforma Google Scholar:

```
("especificação de requisitos" OR "engenharia de requisitos")
AND ("modelo" OR "modelos" OR "workflows" OR "fluxo" OR
"fluxos" OR "diagrama" OR "diagramas")
```

A busca retornou 153 resultados. Para focar a análise exploratória em trabalhos mais recentes, aplicou-se um filtro temporal desde 2021, o que reduziu o conjunto a 54 artigos, cujos resumos foram analisados, selecionando os 20 artigos considerados mais relevantes para o tema.

A análise preliminar desses artigos indicou uma tendência recorrente: a geração de artefatos de especificação de software a partir de modelos de processo de negócio, com destaque para o uso de BPMN. Esse resultado foi fundamental para delimitar e confirmar o escopo desta pesquisa.

4.2.2 Seleção da literatura consolidada

O artigo de Santander *et al.* (2022), intitulado “Obtendo Casos de Uso a Partir de Modelos de Processos de Negócio: Uma Revisão Sistemática da Literatura”, foi escolhido como ponto de partida. Essa decisão se justifica pois o referido artigo já havia realizado um rigoroso processo de busca em cinco bases de dados, analisando 6.267 estudos para, ao final, selecionar 22 trabalhos como os mais relevantes sobre o tema.

Contudo, o escopo da revisão de Santander *et al.* (2022) era mais amplo do que o desta pesquisa, incluindo trabalhos que não utilizavam BPMN ou que não eram focados em regras de transformação. O objetivo desta pesquisa, por outro lado, é especificamente a automação de regras de mapeamento de BPMN para UML via IA. Portanto, foi necessário realizar um segundo processo de filtragem sobre os 22 artigos selecionados pelos autores. Os Critérios de Inclusão (CI) definidos para esta etapa, bem como suas justificativas, são detalhados na Tabela 3.

Tabela 3: Critérios de Seleção para a literatura de base.

ID	Critério	Justificativa
CI 1	O artigo não apresenta uma solução automatizada.	O objetivo desta pesquisa é propor um novo método de automação, via IA, para regras que ainda são manuais. Portanto, apenas trabalhos que os autores classificaram como “Não Automatizado” foram incluídos na seleção.
CI 2	O artigo utiliza BPMN como notação de entrada.	Garante que os trabalhos selecionados estejam alinhados ao escopo do objetivo geral deste trabalho. Conforme aponta a própria RSL de base, o BPMN é o padrão internacional e a notação mais utilizada para modelar processos, correspondendo a 50% dos trabalhos analisados. Portanto, apenas aqueles que utilizam BPMN foram selecionados.
CI 3	As regras propostas são passíveis de automação.	Para evitar a subjetividade desta pesquisa, foi adotada a classificação da própria RSL de base. Os autores já haviam analisado quais trabalhos, embora manuais, apresentavam características que contribuem positivamente para a automatização do processo. Apenas estes foram selecionados.
CI 4	O artigo propõe regras de mapeamento claras.	A metodologia deste TCC depende da extração de regras que possam ser traduzidas em <i>prompts</i> . Sendo assim, apenas os artigos que apresentavam heurísticas ou regras concretas selecionados, por fornecerem a matéria-prima essencial para a segunda etapa.

4.2.3 Aplicação dos Critérios de Inclusão

A aplicação detalhada desses critérios em cada um dos 22 trabalhos da RSL de Santander *et al.* (2022) é apresentada na Tabela 4. Os trabalhos foram analisados com base nos critérios de inclusão (CI) definidos, e um comentário foi adicionado para justificar a exclusão daqueles que não atendiam ao escopo desta pesquisa.

Tabela 4: Análise detalhada da seleção de artigos da RSL.

ID	Título	CI 1	CI 2	CI 3	CI 4
1	<i>Semi-formal transformation of secure business process into analysis class and use case models: An MDAapproach</i>	✓	✓	✓	✓

Tabela 4: Análise detalhada da seleção de artigos da RSL.

ID	Título	CI 1	CI 2	CI 3	CI 4
2	<i>A description of semi-automatic creation of requirements specification from business activities</i>	✓	×	-	-
3	<i>Deriving use case from business process models developed using Norm Analysis</i>	✓	×	-	-
4	<i>From Business Process Models to Use Case Models: A Systematic Approach</i>	✓	✓	✓	✓
5	<i>Transformation of Coloured Petri Nets to UML 2 Diagrams</i>	✓	×	-	-
6	<i>Transformation of decisional models into UML: application to GRAI grids</i>	✓	×	-	-
7	<i>Transforming an enterprise model into a use case model in business process systems</i>	×	-	-	-
8	<i>Deriving use cases from business process model</i>	✓	×	-	-
9	<i>A business-oriented approach to Requirements Elicitation</i>	✓	×	-	-
10	<i>Improving Requirements Analysis through Business Process Modelling: A Participative Approach</i>	✓	✓	×	-
11	<i>An algorithm to derive use cases from business processes</i>	✓	×	-	-
12	<i>A framework for business model driven development</i>	✓	✓	×	-
13	<i>Coordination Analysis: A Method for Deriving Use Cases from Process Dependencies</i>	✓	×	-	-
14	<i>Deriving Use Cases from Business Processes: A Goal-Oriented Transformational Approach</i>	×	-	-	-
15	<i>From a BPMN Model to an Aligned UML Analysis Model</i>	✓	✓	✓	✓
16	<i>Application of Heuristics in Business Process Models to Support Software Requirements Specification</i>	×	-	-	-
17	<i>Specification of software requirements with support of business process ontologies</i>	×	-	-	-
18	<i>Deriving Integrated Software Design Models from BPMN Business Process Models</i>	✓	✓	✓	✓
19	<i>Business Process Model Driven Automatic Software Requirements Generation</i>	×	-	-	-
20	<i>Knowledge-Based UML Use Case Model Transformation Algorithm</i>	✓	×	-	-
21	<i>Deriving use case models from BPMN models</i>	✓	✓	✓	✓

22	<i>Deriving a Goal-based Use Case Model from Business Goals and Process Model</i>	✓	×	-	-
----	---	---	---	---	---

Legenda: ✓ = O artigo passou nesta etapa de verificação; × = O artigo falhou nesta etapa de verificação (levando à exclusão); - = Não aplicável.

A Tabela 4 demonstra visualmente o processo de filtragem. Dos 22 artigos da RSL, 17 foram excluídos por não atenderem a um ou mais dos critérios de inclusão definidos na Tabela 3. A exclusão ocorreu da seguinte forma:

- **Falha no CI 1:** O primeiro grupo, composto por 5 artigos (IDs 7, 14, 16, 17 e 19), foi excluído por já apresentar uma solução automatizada. Conforme a análise de Santander *et al.* (2022), estes trabalhos já possuíam ferramentas próprias de automação, como a EMUCase (ID 7), IRIS-RE (ID 14), SRPD (ID 16), o ecossistema PM2ONTO (ID 17) e ferramentas do *Eclipse Modeling Framework* (ID 19).
- **Falha no CI 2:** O maior grupo, com 10 artigos (IDs 2, 3, 5, 6, 8, 9, 11, 13, 20 e 22), foi excluído por não utilizar o BPMN como notação de entrada, utilizando outros modelos como Diagramas de Atividades ou Análise de Normas.
- **Falha no CI 3:** Por fim, 2 artigos (IDs 10 e 12) foram excluídos por não serem passíveis de automação. Segundo os autores da RSL, suas regras dependiam excessivamente de artefatos externos ou não apresentavam um passo a passo claro, inviabilizando a automação.

Com isso, a seleção final foi composta pelos 5 trabalhos a seguir, que atenderam a todos os critérios de inclusão:

- **ID 1:** *Semi-formal transformation of secure business process into analysis class and use case models: An MDA approach* (Rodríguez *et al.*, 2010).
- **ID 4:** *From Business Process Models to Use Case Models: A Systematic Approach* (Cruz *et al.*, 2014).
- **ID 15:** *From a BPMN Model to an Aligned UML Analysis Model* (Khlif *et al.*, 2018).
- **ID 18:** *Deriving Integrated Software Design Models from BPMN Business Process Models* (Cruz e da Cruz, 2018).
- **ID 21:** *Deriving use case models from BPMN models* (Bouzidi *et al.*, 2017).

Estes artigos, que foram detalhados no Capítulo 3, constituem a literatura base que serviu como entrada para a segunda etapa da metodologia.

4.3 SISTEMATIZAÇÃO E ADAPTAÇÃO DAS REGRAS PARA IA

A segunda etapa da metodologia consiste no processo de analisar, sintetizar e adaptar as regras de mapeamento, extraídas da literatura de base, detalhada no Capítulo 3, para

um formato que possa ser compreendido e executado por uma Inteligência Artificial Generativa. Este processo foi dividido em duas fases: sistematização das regras e adaptação para *prompts*.

4.3.1 Sistematização das Regras de Mapeamento

O primeiro passo é a sistematização. As regras e heurísticas dos cinco artigos selecionados foram analisadas e comparadas, buscando identificar pontos de convergência e divergência. Conforme visto no Capítulo 3, embora as abordagens tenham particularidades, como o uso de modelos intermediários ou foco em texto, há um forte consenso sobre o mapeamento dos elementos centrais.

A partir dessa análise, foi criado um “conjunto de regras mestras”, que consolida o conhecimento validado pela literatura de base, mas não se limita a uma reprodução literal das regras extraídas. Foram realizadas adaptações e ajustes de forma a reduzir ambiguidades, padronizar nomenclaturas e tornar as instruções mais executáveis no contexto desta pesquisa. Além disso, regras complementares foram propostas pela própria autora, a partir da verificação de lacunas e necessidades identificadas durante testes preliminares com diferentes ferramentas de IA.

A Tabela 5 apresenta este conjunto, agrupado pelo artefato UML de destino, e indica quais dos artigos de base dão suporte a cada regra.

Tabela 5: Regras mestras para o mapeamento BPMN-UML.

Artefato Alvo (UML)	ID	Regra de Mapeamento Sistematizada	Fontes (IDs da RSL)
Diagrama de Casos de Uso	R1	Criar Atores a partir de <i>Lanes</i> e, quando não houver <i>Lanes</i> , a partir da <i>Pool</i> ; ignorar <i>Pools</i> vazias (sem elementos funcionais).	1, 4, 15, 18, 21
	R2	Criar Casos de Uso a partir de <i>Tasks</i> ; permitir agrupamento de <i>Tasks</i> sequenciais com objetivo único; na dúvida, manter 1:1.	1, 4, 15, 18, 21
	R3	Mapear <i>Call Activity</i> como Caso de Uso e gerar relacionamentos << <i>include</i> >> para os Casos de Uso das atividades internas.	18

Tabela 5: Regras mestras para o mapeamento BPMN-UML.

Descrição Textual	R4	Para <i>Gateways</i> com caminhos alternativos, gerar << <i>extend</i> >> da primeira <i>Task</i> de cada caminho para o Caso de Uso anterior ao <i>Gateway</i> .	18, 21
	R5	Usar condições de fluxo (<i>Gateways</i> e <i>Intermediate Link Events</i>) para definir Pré-condições .	4
	R6	Usar <i>End Events</i> para definir Pós-condições dos Casos de Uso de origem.	4
	R7	Usar <i>Start Events</i> e <i>Intermediate Events</i> para identificar Gatilhos .	4
	R8	Usar <i>Data Objects</i> , <i>Associations</i> e <i>Text Annotations</i> para descrever o Cenário ; incorporar anotações obrigatoriamente e fazê-las prevalecer sobre suposições.	4
Diagrama de Classes	R9	Criar Classes a partir de <i>Pools</i> e <i>Lanes</i> ; ignorar <i>Pools</i> vazias; não criar classes a partir de nomes de <i>Tasks</i> .	18
	R10	Quando uma <i>Pool</i> contiver <i>Lanes</i> , relacionar a Classe da <i>Pool</i> às Classes das <i>Lanes</i> por agregação (partes opcionais) ou composição (partes obrigatórias).	elaborado pela autora
	R11	Mapear cada <i>Task</i> como operação pública na Classe da <i>Lane</i> (ou da <i>Pool</i> , se não houver <i>Lanes</i>).	1, 15, 18
	R12	Mapear <i>Data Objects</i> como Classes (com ajuste de nomes) e criar associações nomeadas por verbos com Classes que manipulam/transmitem os dados.	1, 15, 18
	R13	Incluir atributos mínimos nas Classes (com nome, tipo e visibilidade privada), permitindo inferência de atributos como única criação além do XML.	elaborado pela autora
Diagrama de Sequência	R14	Criar Linhas de Vida a partir de <i>Lanes</i> e, quando não houver <i>Lanes</i> , a partir da <i>Pool</i> ; ignorar <i>Pools</i> vazias.	15, 18
	R15	Criar Mensagens para os fluxos de saída de <i>Tasks</i> , <i>Start Events</i> e <i>Catching Events</i> (exceto <i>linkEventDefinition</i>), considerando <i>sequenceFlow</i> , <i>messageFlow</i> e ligações por <i>Link Events</i> ; usar autochamada quando origem = destino.	elaborado pela autora (com base em 15, 18)

Tabela 5: Regras mestras para o mapeamento BPMN-UML.

R16	Representar <i>Gateways</i> alternativos com fragmento alt , com operandos e guardas [] conforme as condições do BPMN, agrupando as mensagens de cada caminho.	elaborado pela autora
-----	--	-----------------------

4.3.2 Adaptação das Regras para *Prompts*

O segundo passo desta etapa consistiu em adaptar o conjunto de “regras mestras”, apresentado na Tabela 5, para um *prompt* capaz de orientar a execução precisa por um modelo de linguagem de grande porte (LLM). O desafio desse processo foi traduzir regras semânticas, originalmente formuladas para interpretação humana, em instruções procedimentais claras, inequívocas e auditáveis, adequadas à execução automatizada por uma Inteligência Artificial Generativa.

Para aumentar a fidelidade da transformação e reduzir variações entre execuções, o *prompt* final foi construído com base em técnicas de Engenharia de *Prompts* discutidas no Capítulo 2, incorporando não apenas as regras de mapeamento, mas também diretrizes rígidas de comportamento, validação e formatação. O processo de construção do *prompt* seguiu as etapas a seguir:

1. **Atribuição de Persona e Prioridade de Regras:** O *prompt* inicia instruindo o modelo a atuar como um engenheiro de requisitos sênior, especialista em BPMN e UML. Em seguida, define explicitamente que a prioridade máxima é a adesão estrita ao conjunto de regras fornecido, que deve prevalecer sobre boas práticas genéricas, convenções de mercado ou *design patterns*. Essa diretriz foi incluída para reduzir “ajustes criativos” do modelo e forçar a execução conforme a especificação do experimento.
2. **Definição da Entrada e Fonte Oficial:** O *prompt* especifica que o modelo BPMN será fornecido em XML BPMN 2.0, podendo ser enviado como texto no chat ou como arquivo. Também prevê o envio opcional de uma imagem do diagrama como apoio visual. Entretanto, o XML é definido como a fonte oficial e deve prevalecer em

qualquer decisão de interpretação, sendo a imagem utilizada apenas para desempate quando o XML permitir múltiplas leituras estruturais.

3. **Diretrizes de Conduta e Controle de Ruído:** Para aumentar a reprodutibilidade, o *prompt* impõe restrições comportamentais: leitura completa de todos os arquivos enviados, proibição de comentários e explicações fora do escopo, e obrigação de perguntar antes de gerar artefatos sempre que houver ambiguidade estrutural no XML. Essas restrições foram importantes para evitar respostas que misturam justificativas, inferências não verificáveis ou decisões implícitas.
4. **Tradução das Regras Mestras em Instruções Operacionais:** O núcleo do *prompt* apresenta o conjunto completo de 16 regras de mapeamento, organizadas por artefato UML de destino (DCU, descrição textual, DC e DS). Cada regra foi escrita em linguagem diretiva, orientada a ações executáveis (por exemplo, “Para cada Lane, crie um Ator”), com critérios de decisão explícitos quando necessário (por exemplo, regra de agrupamento de *Tasks* com padrão 1:1 na dúvida). Essa conversão buscou eliminar interpretações abertas e transformar as regras em passos de execução.
5. **Processo de Execução Obrigatório em Etapas (1–3):** Além das regras, o *prompt* foi estruturado com um processo obrigatório dividido em três etapas para controlar o raciocínio e reduzir erros em cascata:
 - **Etapas 1 – Mapeamento Estrutural do XML:** antes de gerar qualquer artefato, o modelo deve produzir um inventário explícito de Pools, Lanes, elementos internos (Tasks, Events, Gateways), conexões e sucessores imediatos, incluindo o tipo de fluxo e a Lane de destino. Também deve registrar decisões formais de agrupamento de *Tasks*, analisar Gateways (identificando *Task Base* e *Tasks de Extensão*), listar subprocessos e mapear Data Objects e suas associações.
 - **Etapas 2 – Validação (QA) do Mapeamento:** o modelo deve atuar como revisor técnico sênior e validar rigorosamente a Etapa 1 contra o XML, por

meio de um checklist. Caso exista qualquer erro ou omissão, deve apontar explicitamente a divergência e fornecer o dado corrigido; caso contrário, deve declarar a validação como concluída.

- **Etapla 3 – Geração dos Artefatos Finais:** somente após a organização e validação do mapeamento estrutural, o modelo deve aplicar exclusivamente as 16 regras e gerar os artefatos finais.

6. **Estruturação e Padronização da Saída:** O *prompt* instrui o modelo a organizar a resposta em formato *Markdown*, com seções fixas por etapa e por artefato. Os diagramas devem ser fornecidos em código **PlantUML**, delimitados entre `@startuml` e `@enduml`. A descrição textual dos casos de uso deve ser gerada em *snippet* de código *Markdown*, no formato apropriado para um arquivo `.md`, contendo obrigatoriamente os campos: Nome do Caso de Uso, Atores, Pré-condições, Pós-condições, Gatilhos e Cenário.

7. **Restrições de Vocabulário na Descrição Textual:** Para evitar que a saída misture notação de processo com notação de requisitos, o *prompt* proíbe explicitamente o uso de termos BPMN na descrição textual (por exemplo, *gateway*, *event*, *timer*, *message flow* e *sequence flow*). Assim, as condições e comportamentos devem ser descritos apenas em linguagem natural típica de Casos de Uso UML, interpretando os elementos do processo como condições, ações e fluxos funcionais, sem citar a notação de origem.

Esse formato foi escolhido por aumentar a legibilidade e, simultaneamente, impor estrutura suficiente para comparação sistemática entre execuções. Em especial, a separação em etapas (mapeamento, validação e geração) foi adotada para reduzir inconsistências na interpretação do XML e garantir que a aplicação das regras ocorra apenas após a consolidação e checagem do modelo estrutural. A aplicação dessas técnicas resultou no *prompt* de mapeamento base, apresentado a seguir, que foi o principal instrumento utilizado na execução do estudo de caso.

Prompt de Mapeamento BPMN-UML

Aja como um engenheiro de requisitos sênior, especialista em modelagem de processos BPMN e especificação de software UML. Sua prioridade máxima é a adesão estrita às regras de mapeamento que serão fornecidas abaixo, mesmo que elas pareçam contradizer práticas comuns de mercado ou padrões de “design patterns” tradicionais. O conjunto de regras é a lei absoluta desse projeto.

Você receberá, por meio de um arquivo .zip ou em formato de texto colado aqui no chat, um modelo de processo de negócio BPMN como entrada, em formato XML BPMN 2.0. Também pode ser enviada, apenas como apoio visual, uma imagem do diagrama, que deve ser usada somente para esclarecer dúvidas estruturais quando o XML permitir múltiplas interpretações, sem jamais substituir a estrutura declarada no XML, que deve ser a fonte oficial para todos os mapeamentos.

Não tenha preguiça de abrir todos os documentos que serão enviados e fazer a leitura completa deles. Não adicione comentários, explicações ou observações fora da aplicação estrita das regras. Se houver qualquer dúvida ou ambiguidade estrutural no XML, pergunte antes de gerar os artefatos.

[enviar XML no chat ou em arquivo + imagens]

Analise rigorosamente o modelo BPMN e o conjunto de 16 regras de mapeamento a seguir:

Para o Diagrama de Casos de Uso (DCU):

- **R1:** Para cada Lane presente no BPMN, crie um Ator correspondente. Quando uma Pool NÃO possuir Lanes internas, crie o Ator diretamente a partir dessa Pool. Ignore as Pools que não contenham Lanes, Tasks, Events, Gateways ou qualquer elemento funcional do processo, ela NÃO deve ser mapeada para nenhum elemento UML.
- **R2:** Para cada Task presente no BPMN, crie um único Caso de Uso e defina como Ator aquele correspondente à Pool ou à Lane em que essa Task está localizada no BPMN. Quando duas Tasks sequenciais (conectadas por um Sequence Flow ou um Message Flow) compartilharem um único objetivo funcional (por exemplo, “Enviar dados” e “Receber dados”, ou “Solicitar dados” e “Fornecer dados”), você deverá juntar essas duas Tasks em um único Caso de Uso e definir como Atores aqueles correspondentes à Pool ou à Lane em que essas Tasks estão localizadas no BPMN. Entretanto, em caso de qualquer dúvida se as Tasks compartilham o mesmo objetivo, ou se o agrupamento tornar o nome do Caso de Uso vago, OBEDEÇA O PADRÃO 1:1 (crie um Caso de Uso separado para cada Task).
- **R3:** Para cada Subprocesso Reutilizável (Call Activity), crie um único Caso de Uso que o represente. Em seguida, para cada Task interna desse Subprocesso (ou para cada grupo de Tasks internas mapeado como Caso de Uso pela R2), crie relacionamentos <<include>> partindo do Caso de Uso do Subprocesso e chegando aos Casos de Uso internos.
- **R4:** Para cada Gateway que represente um fluxo alternativo, cada primeira Task de cada caminho de saída deve ser modelada como um Caso de Uso que estende (<<extend>>) o Caso de Uso que antecede o Gateway.

Para a Descrição Textual do Caso de Uso:

- **R5:** Use as condições de fluxo (Gateways e Intermediate Link Events) para auxiliar na definição das Pré-condições de cada Caso de Uso que representam as Tasks geradas por esses fluxos.
- **R6:** Use os End Events para definir as Pós-Condições de cada Caso de Uso que representam as Tasks de origem.
- **R7:** Use os Start Events e os Intermediate Events para identificar os Gatilhos de cada Caso de Uso.
- **R8:** Use os Data Objects, Associations e Text Annotations para descrever o Cenário de cada Caso de Uso. Toda Text Annotation associada a uma Task, Event ou Flow deve ser obrigatoriamente incorporada ao texto do Caso de Uso correspondente. As Text Annotations devem prevalecer sobre qualquer suposição, generalização ou dedução não presente no XML. Sempre que existir uma Text Annotation esclarecendo limitações do processo, essa anotação deve ser refletida explicitamente na descrição textual.

Para o Diagrama de Classes (DC):

- **R9:** Para cada Pool e para cada Lane presente no BPMN, crie exatamente uma Classe UML correspondente, usando o nome da Pool ou da Lane como nome da Classe. Não crie classes adicionais baseadas no nome de Tasks, funções ou descrições internas. Ignore as Pools que não contenham Lanes, Tasks, Events, Gateways ou qualquer elemento funcional do processo, ela NÃO deve ser mapeada para nenhum elemento UML.
- **R10:** Quando uma Pool possuir Lanes internas, a Classe correspondente à Pool deve se relacionar com cada Classe que representa suas Lanes por meio de um relacionamento de Agregação (quando as Lanes representarem partes opcionais) ou de Composição (quando representarem partes obrigatórias e inseparáveis).
- **R11:** Para cada Task presente no BPMN, crie uma Operação na Classe correspondente à Lane em que essa Task está localizada. Se a Pool não possuir Lanes internas, a Operação deve ser criada diretamente na Classe dessa Pool. As Operações devem ter modo de proteção público.
- **R12:** Para cada Data Object do BPMN, crie uma Classe UML correspondente. Se o nome do Data Object começar com “Dados do(a)”, remova esse prefixo e renomeie a Classe UML usando apenas o complemento, ajustando para um nome coerente. Só crie a Classe UML se ainda não existir uma Classe originada de Pool ou Lane com esse mesmo nome (ou equivalente). Crie um relacionamento de Associação entre essa Classe de Dados e todas as Classes originadas de Pools ou Lanes que criam, fornecem, consomem, consultam, manipulam, registram ou transmitem esse dado, e atribua a cada Associação um nome expresso em verbo na terceira pessoa do singular, como “envia”, “fornece”, “recebe”, “analisa”, “manipula”, “registra”, ou outro termo adequado ao papel exercido pela Classe em relação ao dado.
- **R13:** Inclua diretamente dentro de cada Classe UML os atributos que forem necessários para representar de forma mínima e coerente as informações manipuladas por essa Classe no processo. Esses atributos podem ser inferidos pela IA, mesmo que não estejam presentes no

XML do BPMN, sendo este o único ponto do mapeamento em que a criação de elementos não declarados no XML é permitida. Os atributos devem possuir nomes claros, tipos definidos e modo de proteção privado.

Para o Diagrama de Sequência (DS):

- **R14:** Para cada Lane no BPMN, crie uma Linha de Vida correspondente no Diagrama de Sequência. Quando uma Pool não possuir Lanes internas, crie a Linha de Vida diretamente a partir dessa Pool. Ignore as Pools que não contenham Lanes, Tasks, Events, Gateways ou qualquer elemento funcional do processo, ela NÃO deve ser mapeada para nenhum elemento UML.

- **R15:** Para cada Task, Start Event e Catching Event (exceto os de linkEventDefinition, pois eles funcionam como um Tipo de Fluxo) presente no BPMN, identifique todos os seus fluxos de saída (via sequenceFlow, messageFlow ou [intermediateThrowEvent + intermediateCatchEvent com linkEventDefinition e que tenham os mesmos nomes]). Identifique a Lane/Pool de origem e a Lane/Pool do elemento de destino (Task, Gateway, Event ou qualquer outro elemento do BPMN). Em seguida, crie uma Mensagem no Diagrama de Sequência saindo da Linha de Vida de origem para a Linha de Vida de destino. Nomeie a mensagem com o nome do elemento de origem. Se a Lane ou Pool de origem for igual a de destino, crie uma Mensagem de autochamada, na qual a própria Linha de Vida é emissora e receptora.

- **R16:** Para cada Gateway que represente um fluxo alternativo, inclua um Fragmento de Interação do tipo alt que represente essa lógica condicional. Para cada fluxo de saída do Gateway crie um Operando (bloco) dentro do fragmento alt, que possua uma condição de guarda indicada entre colchetes [], exatamente como definida no fluxo condicional do BPMN. Dentro de cada Operando, agrupe todas as Mensagens pertencentes ao caminho específico daquele Operando.

Processo de Execução Obrigatório (Chain of Thought): Para garantir a precisão, você deve seguir estas etapas sequencialmente antes de gerar os artefatos finais:

Etapas 1 - Mapeamento Estrutural do XML (Mental Scratchpad): Antes de gerar qualquer diagrama, faça uma leitura completa do(s) arquivo(s) XML e liste explicitamente os seguintes dados:

1. **Inventário de Elementos e Conexões:** Liste cada Pool e suas Lanes internas. Dentro de cada Lane (ou Pool sem Lanes internas), liste ordenadamente todas as Tasks, Events e Gateways. Para cada elemento, identifique explicitamente o Elemento Sucessor Imediato (via sequenceFlow, messageFlow ou [intermediateThrowEvent + intermediateCatchEvent com linkEventDefinition e que tenham os mesmos nomes]), e em qual Lane esse sucessor está localizado. Formato Obrigatório: [Elemento Atual] (Tipo) (Lane A) –(Tipo de Fluxo)–> [Elemento Sucessor] (Tipo) (Lane B).
2. **Análise de Agrupamento:** Para cada par de Tasks sequenciais, aplique o teste de “objetivo funcional único”. Declare explicitamente sua decisão para cada caso: “DECISÃO: AGRUPAMENTO: [Task A + Task B] -> [Novo Nome Unificado]” ou “DECISÃO: MANTER SEPARADO (Critério de Diferenciação)”. Lembre-se: na dúvida, mantenha separado.

3. **Análise de Gateways (Predecessores e Sucessores):** Para cada Gateway, rastreie o fluxo de entrada “para trás” até encontrar a última Task executada antes do Gateway (Task Base). Para cada fluxo de saída, rastreie “para frente” até encontrar a primeira Task válida (Tasks de Extensão). Verifique Sequence Flows, Message Flows e Fluxos via Intermediate Link Events ([intermediateThrowEvent + intermediateCatchEvent com linkEventDefinition e que tenham os mesmos nomes]). Formato de Saída: [Task Base] –(Gateway X: Caminho A)–> [Task de Extensão].
4. **Estrutura de Subprocessos:** Identifique todos os Subprocessos e liste o nome do Subprocesso e, caso esteja disponível, as Tasks Internas contidas nele.
5. **Mapeamento de Dados:** Liste os Data Objects e quais Tasks explicitamente se conectam a eles via Associations.

Formato da Saída: Organize a resposta em formato Markdown. Crie uma seção de Nível 2 (##) contendo o resultado da Etapa 1, sem gerar os artefatos UML ainda.

Etapa 2 - Atue agora como um Revisor Técnico Sênior (QA): Valide rigorosamente os dados mapeados na Etapa 1 antes de gerar qualquer diagrama.

Seu objetivo é garantir que o “Mental Scratchpad” (Etapa 1) esteja 100% fiel ao XML BPMN original. Erros aqui causarão falhas em cascata nos diagramas.

Execute o seguinte Checklist de Validação cruzando os dados da Etapa 1 com o XML:

1. **Validação de Estrutura:** Todas as Pools e Lanes presentes no XML foram listadas? As Pools “vazias” (sem elementos funcionais) foram corretamente identificadas para serem ignoradas?
2. **Validação de Inventário:** Todas as Tasks, Events e Gateways presentes no XML foram listadas? Os “Elementos Sucessores” foram listados corretamente? A “Lane de Destino” identificada para cada sucessor bate com a Lane declarada no XML para aquele elemento alvo?
3. **Validação de Agrupamento:** Revise os candidatos a agrupamento. A decisão de agrupar ou separar seguiu um critério estrito? Se houver ambiguidade, a decisão foi “Manter Separado”?
4. **Validação de Gateways:** O rastreamento está bidirecional? Para trás: Identificou a Task Base correta? Para frente: Seguiu o fluxo (via sequenceFlow, messageFlow e [intermediateThrowEvent + intermediateCatchEvent com linkEventDefinition e que tenham os mesmos nomes]) até encontrar a primeira Task de Extensão real?
5. **Validação de Subprocessos:** Se houver Subprocessos, as atividades internas foram corretamente listadas para o <<include>>?
6. **Validação de Dados:** Os Data Objects e suas Associações com as Tasks foram mapeados corretamente?

SAÍDA OBRIGATÓRIA:

- Se encontrar QUALQUER erro ou omissão: Liste explicitamente: “ERRO ENCONTRADO no item X: [Explicação da divergência com o XML]”. Em seguida, forneça o dado corrigido.
- Se estiver TUDO CORRETO: Responda apenas: “Validação Concluída com Sucesso. A Etapa 1 está fiel ao XML. Pronto para a Etapa 3.”

Aguardando sua validação.

Etapa 3 - “Geração dos Artefatos Finais”: Agora que os dados estão organizados e validados, aplique EXCLUSIVAMENTE o conjunto das 16 regras de mapeamento, baseando-se nos dados gerados pela Etapa 1, validados e corrigidos na Etapa 2, e gere os seguintes artefatos de software: Diagrama de Casos de Uso, Descrição Textual do Caso de Uso (Nome do Caso de Uso, Atores, Pré-condições, Pós-condições, Gatilhos e Cenário), Diagrama de Classes e Diagrama de Sequência. Deve ser feita uma descrição textual para cada um dos Casos de Uso criados no DCU.

Vocabulário da Descrição Textual: Não utilize termos próprios da notação BPMN, tais como Gateway, Event, Timer, Intermediate Event, Message Flow, Sequence Flow ou qualquer outro elemento técnico do BPMN. A descrição deve ser escrita exclusivamente com linguagem natural típica da Engenharia de Requisitos e dos Casos de Uso UML, interpretando esses elementos como condições, ações ou comportamentos funcionais, sem mencionar a notação de origem.

Formato da Saída Final: Organize a resposta final em formato Markdown. Apresente os artefatos finais em seções de Nível 2 (##) para cada artefato (DCU, Texto, DC, DS). A Descrição Textual deve ser fornecida em Snippet de código Markdown, exatamente no formato adequado para um arquivo .md, mantendo a formatação consistente e estruturada. Cada diagrama deve ser representado em código PlantUML delimitado entre @startuml e @enduml. Utilize as propriedades e estratégias do PlantUML para gerar diagramas claros: minimize cruzamentos de linhas, mantenha alinhamento horizontal/vertical sempre que possível e evite reordenação automática de elementos, defina a ordem dos elementos sempre que possível.

4.4 ESTUDO DE CASO

A Figura 2 apresenta uma visão geral do fluxo do estudo de caso, desde a modelagem e exportação do BPMN até a execução manual, a execução nas ferramentas de IA, a geração dos artefatos e a avaliação objetiva por checklist.

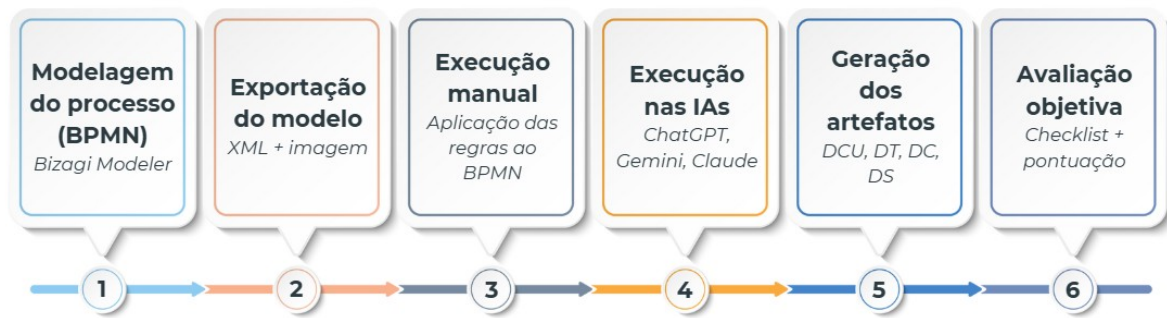


Figura 2: Visão geral do fluxo do estudo de caso.

4.4.1 Seleção e preparação do BPMN

Antes da aplicação das regras de mapeamento, foi criado um modelo BPMN de entrada, que não fosse reaproveitado de exemplos clássicos da literatura utilizada como base. Essa diretriz foi adotada para reduzir vieses, evitar que o estudo se limitasse a cenários já explorados e permitir uma avaliação mais imparcial da metodologia proposta.

O processo escolhido foi o *agendamento de consultas* em uma clínica médica. A escolha foi feita por se tratar de um cenário genérico, comum e de fácil compreensão, sem depender de particularidades de um domínio específico. Buscou-se um modelo relativamente simples em termos de quantidade de elementos, mas, ao mesmo tempo, com variedade suficiente para exercitar os casos previstos no conjunto de regras, de forma que cada regra pudesse ser validada em pelo menos um trecho do fluxo.

A modelagem foi construída no *Bizagi Modeler*, organizando o processo principal em dois participantes (*Pools*): a **Clínica Médica** e a **Operadora do Plano de Saúde**. Na clínica, as responsabilidades foram separadas em raias (*Lanes*) para **Paciente**, **Atendente** e **Médico**, permitindo representar com clareza quem executa cada ação. A visão geral do processo está apresentada na Figura 3.

No fluxo principal, o paciente inicia a solicitação enviando seus dados pessoais e preferências de horário, e o atendente agenda a consulta a partir dessas informações. Em seguida, o atendimento é condicionado a uma confirmação enviada ao paciente com antecedência de 24h ao horário agendado. Nesse processo não é considerado o caso em que o paciente não responde à mensagem de confirmação. Então, quando o paciente responde

não confirmando o horário, a consulta é cancelada e o processo é encerrado. Quando há a confirmação, na hora da consulta, o atendente inicia o pré-atendimento e verifica se o paciente possui plano de saúde. Se o paciente não possuir plano, o processo segue para pagamento particular. Se possuir, são solicitados os dados do plano e é feita uma interação com a Operadora para análise de cobertura, de modo que o modelo representa explicitamente a comunicação entre participantes e o retorno do resultado. Se a cobertura for aprovada, o atendimento é liberado e o médico realiza a consulta. Se a cobertura não for aprovada, o fluxo direciona novamente para pagamento particular.

Para detalhar o pagamento particular sem sobrecarregar o diagrama principal, essa parte foi representada por um subprocesso específico, mostrado na Figura 4. Nesse subprocesso, o atendente solicita o pagamento ao paciente, aguarda o envio do comprovante e, após o recebimento, registra o pagamento no sistema, encerrando o processamento. Essa separação ajuda a manter o processo principal mais legível e, ao mesmo tempo, preserva um trecho do modelo dedicado a etapas que se repetem dentro do processo, o que é relevante para a validação das regras.

Por fim, o modelo foi preparado para os experimentos por meio da exportação no formato BPMN 2.0 (XML), garantindo que a entrada utilizada fosse a estrutura formal do processo e não apenas sua representação visual.

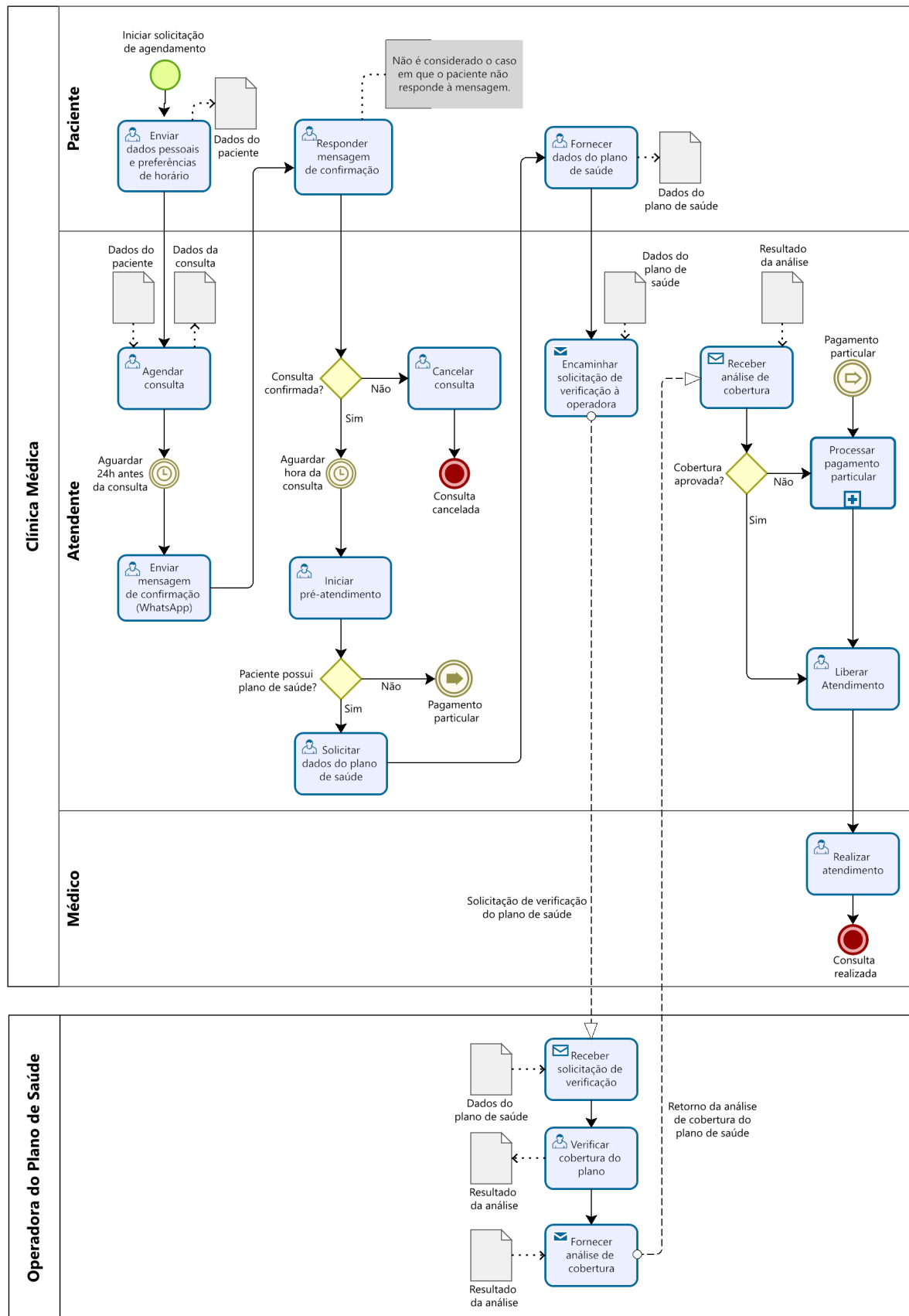


Figura 3: Modelo BPMN do processo de agendamento de consultas.

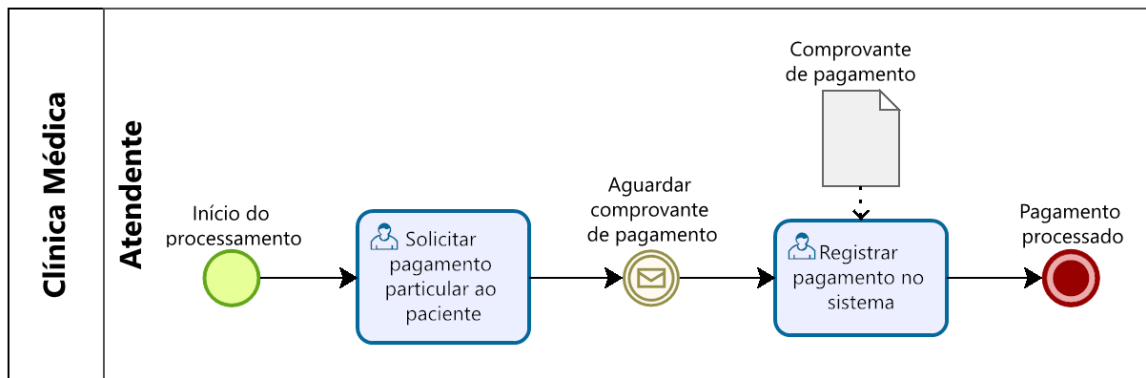


Figura 4: Subprocesso BPMN para processamento de pagamento particular.

4.4.2 Aplicação manual das regras

Com o BPMN definido e preparado, aplicou-se manualmente o conjunto de regras sistematizadas na Tabela 5 para produzir os artefatos UML de referência do estudo de caso. Esse resultado manual serviu como linha base para comparar as saídas dos modelos de IA, permitindo avaliar, regra a regra, a aderência aos mapeamentos propostos.

A aplicação manual seguiu estritamente as regras R1–R16. Primeiramente, foram levantados e classificados os elementos do BPMN (Pools, Lanes, Atividades, Eventos, Gateways e Fluxos) e definidos seus respectivos mapeamentos para os artefatos de especificação. Em seguida, foram elaborados os códigos em PlantUML e gerados os Diagramas de Casos de Uso (Figura 5), de Classes (Figura 6) e de Sequência (Figura 7).

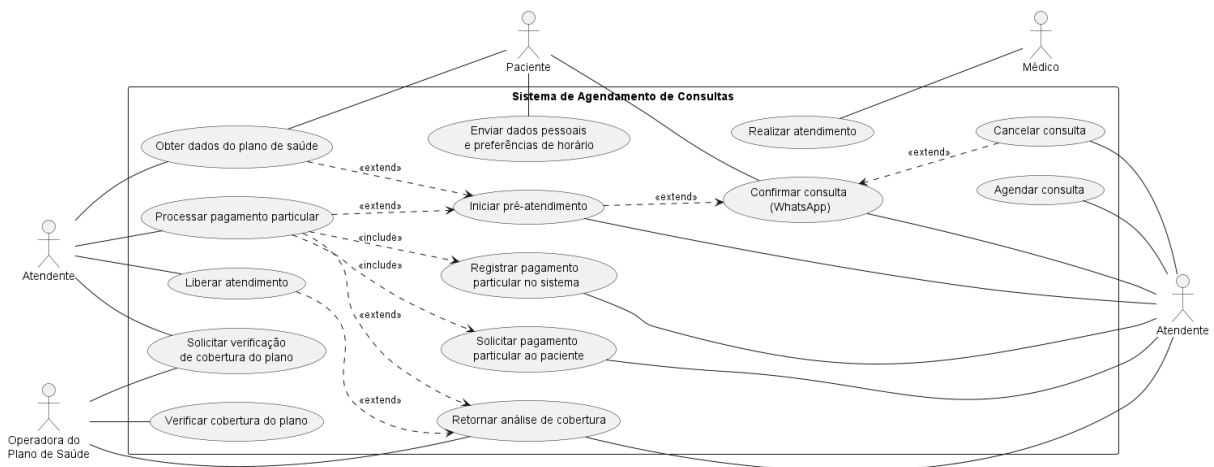


Figura 5: Diagrama de Casos de Uso de referência (execução manual das regras).

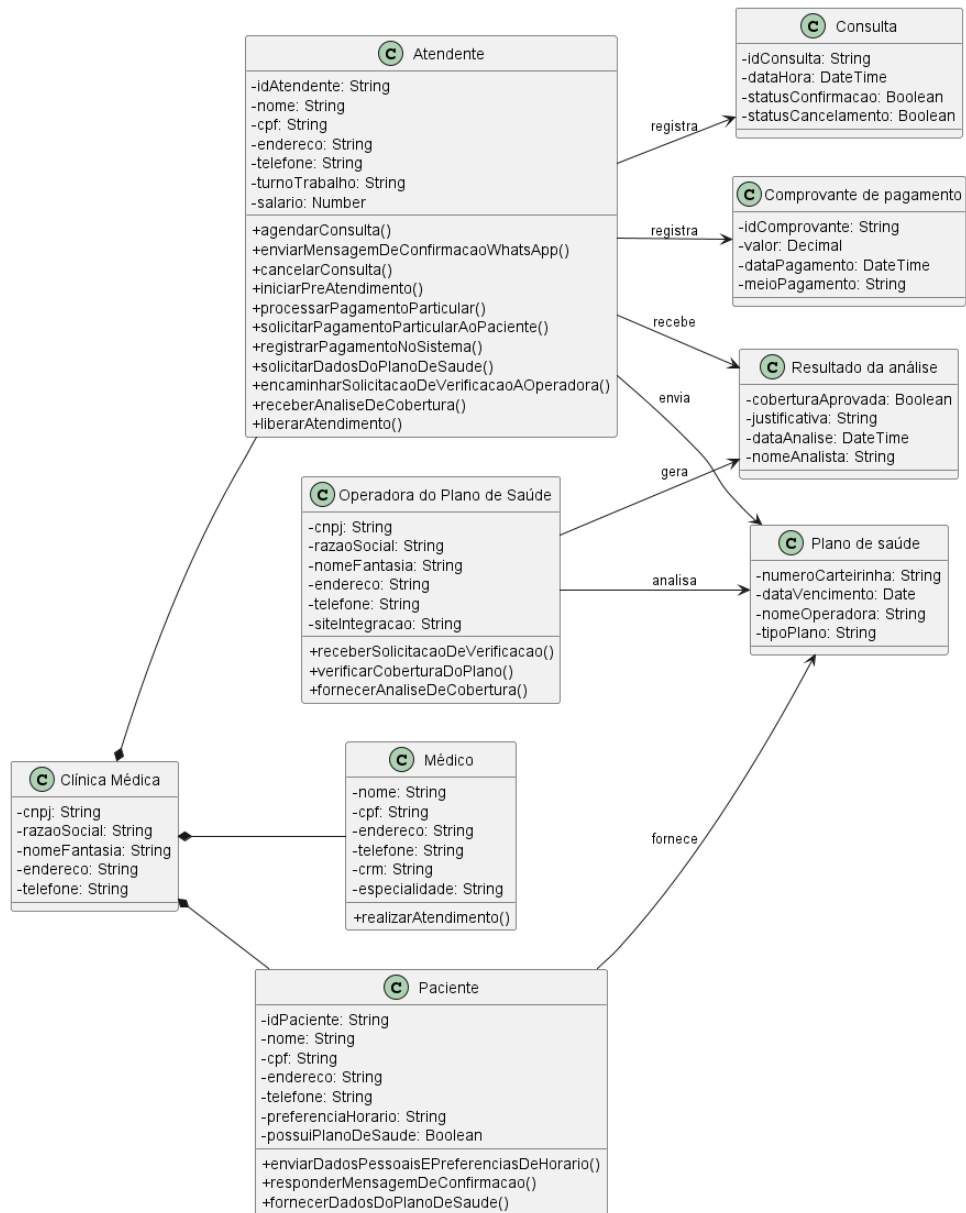


Figura 6: Diagrama de Classes de referência (execução manual das regras).

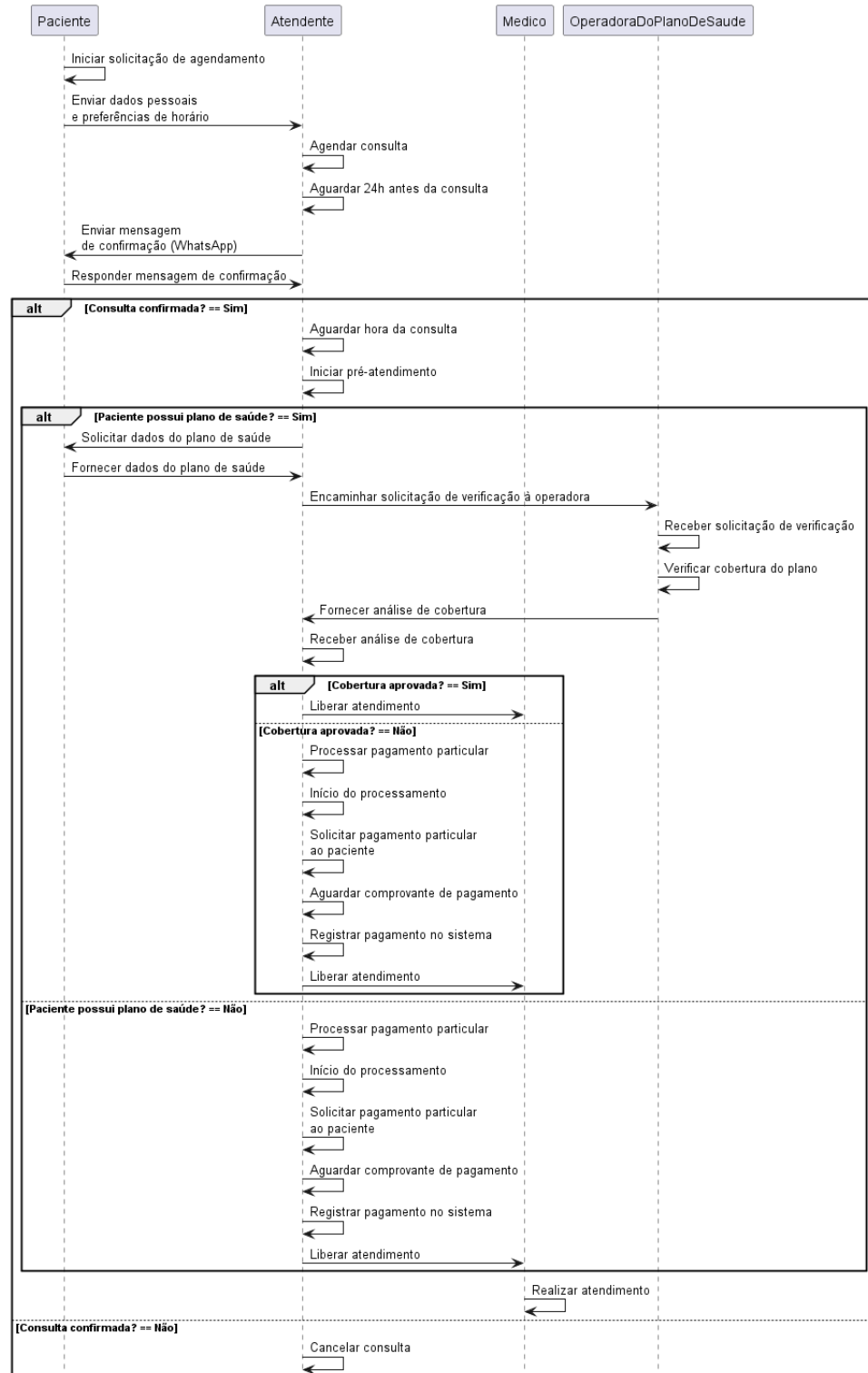


Figura 7: Diagrama de Sequência de referência (execução manual das regras).

A partir do Diagrama de Casos de Uso resultante, foi elaborada uma descrição textual para cada caso de uso, seguindo o formato padronizado adotado no trabalho (nome, atores, pré-condições, pós-condições, gatilhos e cenário). Essas descrições foram redigidas

em linguagem natural típica de Engenharia de Requisitos, evitando termos próprios da notação BPMN, de forma a manter a especificação independente do modelo de origem.

4.4.3 Execução do prompt e geração dos diagramas

Para esta etapa, foram realizadas execuções do prompt em diferentes ferramentas de IA, com o objetivo de gerar os diagramas UML a partir do mesmo insumo e das mesmas regras, em condições controladas. As ferramentas avaliadas foram: ChatGPT (planos Plus e Free), Gemini (planos Pro e Free) e Claude (plano Free). A Tabela 6 resume as IAs e os modelos utilizados (conforme identificado na própria interface de cada ferramenta no momento das execuções).

Tabela 6: Ferramentas de IA utilizadas e modelos.

Ferramenta	Plano	Modelo
ChatGPT	Plus	GPT-5.2 Thinking
ChatGPT	Free	GPT-5 Thinking mini
Gemini	Pro	Gemini 3 Pro
Gemini	Free	Gemini 3 Flash
Claude	Free	Claude Sonnet 4.5

Para cada ferramenta, foram realizadas **10 execuções** do experimento principal, totalizando 50 execuções. A repetição teve como finalidade obter uma média dos resultados ao final do estudo, reduzindo o efeito de variações pontuais de saída entre execuções. Nesta seção, descreve-se apenas o procedimento de execução, a análise comparativa dos resultados será apresentada na seção seguinte.

Durante os testes iniciais, observou-se uma limitação prática relacionada ao tamanho do XML exportado pelo Bizagi. O arquivo original do processo principal possuía aproximadamente **2000 linhas** e o do subprocesso **364 linhas**. Em múltiplas execuções, as IAs sinalizaram que o insumo estava grande demais quando considerado em conjunto com as regras (que também são extensas), o que comprometia a leitura completa do material. Verificou-se que mais da metade dessas linhas dizia respeito a informações de **layout e apresentação** do diagrama (por exemplo: coordenadas, tamanho, cor e estilos), que não eram relevantes para o experimento.

Para contornar esse problema, foi criado um **segundo prompt**, dedicado exclusivamente à **limpeza do XML**, instruindo a IA a remover apenas as linhas associadas a atributos de design/posicionamento e manter integralmente a estrutura e o conteúdo semântico do BPMN. Como esse comando é mais simples e restrito (ler e editar o XML), a IA pôde focar apenas na modificação do arquivo, e o procedimento funcionou de forma consistente. Ao final, o XML do processo principal foi reduzido para **347 linhas** (redução de **82,65%**) e o do subprocesso para **59 linhas** (redução de **83,79%**). Considerando ambos, o total passou de **2364** para **406 linhas** (redução de **82,83%**). Um *diff* (via Diffchecker) entre os arquivos originais e modificados foi utilizado para verificar que a remoção ocorreu apenas nas linhas não essenciais, preservando o conteúdo necessário ao mapeamento.

Prompt de limpeza do XML

Você receberá um arquivo XML BPMN 2.0 em anexo. Abra e faça a leitura completa do arquivo. Sua tarefa é exclusivamente remover as seguintes seções do XML:

- A tag <bpmndi:BPMNDiagram> (com qualquer variação de namespace) incluindo todas as suas subtags e conteúdo interno.
- A tag <BPMNDiagram> incluindo todas as suas subtags e conteúdo interno.
- A tag <extensionElements> incluindo todas as suas subtags e conteúdo interno.

Regras obrigatórias (siga literalmente):

1. Não modifique absolutamente mais nada no XML além da remoção das tags especificadas. Isso inclui:
 - Não alterar nomes de tags.
 - Não alterar nomes de tags.
 - Não alterar atributos.
 - Não remover ou alterar valores.
 - Não alterar indentação, ordem, espaçamento ou estrutura externa.
 - Não modificar comentários.
 - Não remover quaisquer outras tags além das explicitadas.
2. A remoção deve ser total: remova a tag de abertura, todo o conteúdo interno e a tag de fechamento.
3. O XML resultante deve continuar válido.
4. Se as tags explicitadas aparecerem múltiplas vezes, remova todas.
5. Se houver variações com prefixos de namespace, remova também (ex.:<ns0:extensionElements>, <bpmndi:BPMNPlane>, etc.).

6. Não deixe nenhuma linha em branco, vazia ou sem conteúdo.

Entrega: Retorne o XML completo final, sem comentários adicionais, em formato de arquivo .xml.

Com os XMLs reduzidos, o experimento foi executado normalmente, e o prompt principal (composto por instruções iniciais, o modelo BPMN e as regras de mapeamento organizadas por etapas) foi submetido em diferentes configurações para investigar qual forma de envio é melhor compreendida por cada ferramenta. Foram testadas, principalmente, as seguintes variações: (i) envio **separado** em mensagens (início, diagrama, regras + Etapa 1, Etapa 2, Etapa 3) e (ii) envio **consolidado** em uma única mensagem para cada etapa (início + diagrama + regras + Etapa 1, Etapa 2, Etapa 3). Também foram testadas combinações do BPMN em diferentes formatos: (a) texto do XML colado diretamente no chat, (b) XML anexado como arquivo, e (c) XML compactado em arquivo .zip. Adicionalmente, em parte das execuções o diagrama foi fornecido **com imagem** (para apoiar a interpretação visual) e, em outra parte, **sem imagem**.

5 ANÁLISE DOS RESULTADOS

Neste capítulo, foram analisados os resultados obtidos nas 10 execuções realizadas em cada modelo de IA, considerando os quatro artefatos gerados a partir do BPMN do estudo de caso: Diagrama de Casos de Uso (DCU), Descrição Textual dos Casos de Uso (DT), Diagrama de Classes (DC) e Diagrama de Sequência (DS). A análise foi conduzida em dois níveis: (i) por modelo (Seções 5.2 a 5.6), observando desempenho médio, estabilidade e principais dificuldades; e (ii) em conjunto, comparando padrões por artefato e por regra de mapeamento (Tabela 18).

5.1 SISTEMA DE MÉTRICAS

A avaliação dos resultados foi conduzida por meio de um sistema de métricas em formato de checklist, no qual cada regra de mapeamento (R1–R16) foi operacionalizada em perguntas objetivas e verificáveis. Para cada execução, registrou-se a pontuação por regra e as somas agregadas por artefato, além da pontuação total (máximo de 64 pontos no estudo). Em geral, as questões foram pontuadas de forma binária (“Sim” = 1; “Não” = 0). Quando a questão avaliou a presença de elementos indevidos (por exemplo, atores, casos de uso, classes ou mensagens não solicitados), a pontuação foi invertida (“Não” = 1; “Sim” = 0), de modo a penalizar inserções não previstas. O conjunto completo de perguntas e seus respectivos critérios de pontuação está apresentado na Tabela 7.

Adicionalmente, a atribuição de pontuações partiu de uma comparação direta entre os artefatos gerados pelas ferramentas de IA e os artefatos produzidos manualmente. Em termos operacionais, cada item esperado (elementos, relações e conteúdos previstos pelas regras) foi verificado no resultado da IA e confrontado com o que era requerido pela aplicação manual das regras ao modelo BPMN. A presença (ou ausência) desses itens foi então convertida nas respostas do checklist e, consequentemente, nas pontuações apresentadas nas tabelas; a comparação detalhada item a item não é exibida neste trabalho, pois a pontuação consolidada por regra e por artefato já representa o resultado dessa verificação.

No caso específico das regras relacionadas à **Descrição Textual dos Casos de Uso**

(R5–R8), a verificação não foi realizada apenas pela leitura global do texto. Para tornar a correção mais objetiva e viabilizar a análise comparativa entre modelos e execuções, foi definido previamente um conjunto de *itens-alvo* a serem identificados em cada resposta. Cada item encontrado no Caso de Uso correspondente contabilizou 1 ponto para a execução, respeitando os limites máximos de cada regra (R5 = 5, R6 = 2, R7 = 4 e R8 = 10). A identificação considerou variações semânticas (isto é, expressões equivalentes), e não apenas correspondência literal de palavras.

Ressalta-se que os *limites máximos* definidos para R5–R8 são específicos do modelo BPMN adotado neste estudo de caso. Para operacionalizar cada regra na descrição textual, primeiramente derivou-se o conjunto de elementos e informações que deveriam aparecer nos casos de uso correspondentes. Em seguida, cada elemento esperado foi tratado como um *item-alvo* com valor de 1 ponto quando identificado na resposta da IA. Assim, o máximo de cada regra corresponde ao número de itens-alvo definidos para este BPMN; em outros processos, esses máximos podem variar conforme a quantidade de elementos e condições presentes no modelo.

• **R5 – Pré-condições (máx=5):** Foi verificada a presença dos seguintes itens:

1. No UC “Cancelar consulta”: pré-condição indicando que a consulta *não foi confirmada pelo paciente* (por exemplo, o paciente respondeu cancelando ou respondeu que não confirma).
2. No UC “Iniciar pré-atendimento”: pré-condição indicando que a consulta *foi confirmada*.
3. No UC “Solicitar dados do plano de saúde”: pré-condição indicando que o paciente *possui plano de saúde*.
4. No UC “Processar pagamento particular”: pré-condição indicando que o paciente *não possui plano de saúde* **ou** que a *cobertura do plano não foi aprovada*.
5. No UC “Liberar atendimento”: pré-condição indicando que a *cobertura do plano foi aprovada* (isto é, o plano cobre a consulta).

• **R6 – Pós-condições (máx=2):** Foi verificada a presença dos seguintes itens:

1. No UC “Cancelar consulta”: pós-condição indicando *encerramento do processo* (por exemplo, consulta cancelada/processo encerrado).
2. No UC “Realizar atendimento”: pós-condição indicando *encerramento do processo* (por exemplo, consulta realizada/processo encerrado).

• **R7 – Gatilhos (máx=4):** Foi verificada a presença dos seguintes itens:

1. No UC “Enviar dados pessoais e preferências de horário”: gatilho indicando que o *paciente iniciou a solicitação de agendamento*.

2. No UC “Enviar mensagem de confirmação” ou “Confirmar consulta (WhatsApp)”: gatilho indicando que *faltam 24h para a consulta*.
 3. No UC “Iniciar pré-atendimento”: gatilho indicando que *chegou a hora da consulta*.
 4. No UC “Registrar pagamento no sistema”: gatilho indicando que o *paciente enviou o comprovante de pagamento*.
- **R8 – Cenário e informações do processo (máx=10):** Foi verificada a presença dos seguintes itens:
1. No cenário do UC “Enviar dados pessoais e preferências de horário”: menção ao fornecimento dos dados do paciente.
 2. No cenário do UC “Agendar consulta”: menção ao recebimento/consumo dos dados do paciente e geração dos dados da consulta.
 3. No cenário do UC “Fornecer dados do plano de saúde”: menção ao fornecimento dos dados do plano de saúde.
 4. No cenário do UC “Encaminhar solicitação de verificação”: menção ao encaminhamento dos dados do plano de saúde.
 5. No cenário do UC “Receber solicitação de verificação”: menção ao recebimento/análise dos dados do plano de saúde.
 6. No cenário do UC “Verificar cobertura do plano”: menção à geração do resultado da análise.
 7. No cenário do UC “Fornecer análise de cobertura”: menção ao fornecimento do resultado da análise.
 8. No cenário do UC “Receber análise de cobertura”: menção ao recebimento do resultado da análise.
 9. No cenário do UC “Registrar pagamento no sistema”: menção ao recebimento/registro do comprovante de pagamento.
 10. Em algum UC relacionado à confirmação (por exemplo, “Responder mensagem de confirmação” / “Confirmar consulta” / “Confirmar presença”): menção explícita de que *não foi considerado o caso em que o paciente não responde à mensagem de confirmação* (conforme limitação indicada no processo).

Tabela 7: Checklist de perguntas utilizado na avaliação das regras de mapeamento.

Regra	Perguntas
DIAGRAMA DE CASOS DE USO	
R1	Criou um Ator para cada Lane? Criou Atores para as Pools sem Lanes? Criou Atores indevidos?
R2	Todas as Tasks foram cobertas por algum Caso de Uso? Definiu corretamente o Ator do Caso de Uso? Agrupou corretamente Tasks sequenciais com o mesmo objetivo em um único Caso de Uso?

Tabela 7: Checklist de perguntas utilizado na avaliação das regras de mapeamento.

Regra	Perguntas
	Criou Casos de Uso indevidos?
R3	Criou um Caso de Uso representando o Subprocesso (Call Activity)? Criou Casos de Uso para as Tasks internas do Subprocesso? Criou relacionamentos <<include>> com setas pontilhadas partindo do Subprocesso para as Tasks internas? Criou includes indevidos?
R4	Identificou corretamente a Task que antecede o gateway? Identificou corretamente a primeira Task de cada caminho alternativo do gateway? Criou relacionamentos <<extend>> com setas pontilhadas partindo do fluxo alternativo para o Caso de Uso base? Criou extends indevidos?
DESCRIÇÃO TEXTUAL	
R5	Para cada UC esperado, identificou Pré-condições baseadas nas condições dos Gateways ou Eventos? Inseriu no Caso de Uso correto? (máx = 5)
R6	Para cada UC esperado, identificou Pós-condições baseadas nos End Events? Inseriu no Caso de Uso correto? (máx = 2)
R7	Para cada UC esperado, identificou o Gatilho para Casos de Uso iniciados por Eventos (Start/Intermediate)? (máx = 4)
R8	Para cada UC esperado, Data Objects/Associations/Text Annotations foram usados para enriquecer entradas/saídas/informações do cenário? (máx = 10)
DIAGRAMA DE CLASSES	
R9	Criou uma Classe para cada Poole Lane funcional? Criou classes indevidas?
R10	Criou relacionamentos entre a Classe da Pool e as Classes das suas Lanes internas? Usou Agregação ou Composição corretamente conforme a obrigatoriedade da Lane? Criou relacionamentos indevidos?
R11	Criou uma Operação para cada Task? A Operação foi posicionada na Classe correta? As operações foram definidas como públicas? Criou operações indevidas?
R12	Criou uma Classe para cada Data Object? Removeu prefixos "Dados do(a)" ao nomear as Classes? Ignorou Data Objects que criariam Classes duplicadas?

Tabela 7: Checklist de perguntas utilizado na avaliação das regras de mapeamento.

Regra	Perguntas
	Criou associações nomeadas e coerentes com as Classes que manipulam o dado?
R13	Incluiu atributos inferidos dentro das Classes? Atributos têm nome claro e tipo definido? Os atributos definidos são privados? Os atributos são coerentes com o contexto da Classe?
DIAGRAMA DE SEQUÊNCIA	
R14	Criou uma Linha de Vida para cada Lane? Criou Linhas de Vida para as Pools sem Lanes? Criou Linhas de Vida indevidas?
R15	Criou uma Mensagem para cada Task, Start Event e Catching Event (exceto os de link)? A direção da Mensagem (origem → destino) está correta entre as Linhas de Vida? A ordem das Mensagens respeita a sequência do processo? Criou Mensagens indevidas?
R16	Criou um fragmento alt para representar cada Gateway de decisão? Criou corretamente os operandos, separados por condições de guarda entre [], para cada caminho de saída do Gateway? Cada operando contém somente as mensagens do caminho correspondente? Criou fragmentos alt ou operandos indevidos?

Como cada regra possui pontuação máxima diferente, foi necessária a normalização das notas para uma escala comum (0-10), permitindo comparação direta entre regras, diagramas e modelos. A conversão adotada foi uma regra de três simples:

$$x = \frac{10 \cdot \text{nota}}{\text{total}}$$

em que *nota* correspondeu à pontuação obtida no escopo analisado, e *total* correspondeu à pontuação máxima possível no mesmo escopo. Assim, uma execução que atingiu o total máximo recebeu nota 10, e desempenhos parciais foram reescalados proporcionalmente.

Por fim, para reduzir efeitos de variabilidade inerentes a respostas de IA, a avaliação foi repetida em 10 execuções por modelo, calculando-se médias por regra, por artefato e no agregado geral (tabelas de médias em cada seção e Tabela 18).

5.2 CHATGPT PLUS

Tabela 8: Pontuação do ChatGPT Plus por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Diagrama de Casos de Uso													
R1	3	3	3	3	3	3	3	3	3	3	3	3	10
R2	4	4	4	4	4	4	4	4	4	3	4	3.9	9.8
R3	4	4	4	4	4	4	4	4	4	4	4	4	10
R4	4	4	4	4	4	4	4	4	4	4	4	4	10
Soma	15	15	15	15	15	15	15	15	15	14	15	14.9	9.9
Descrição Textual dos Casos de Uso													
R5	5	5	5	5	5	5	5	5	5	5	5	5	10
R6	2	2	2	2	2	2	2	2	2	2	2	2	10
R7	4	4	4	3	3	4	4	4	4	4	3	3.7	9.3
R8	10	9	10	10	10	10	10	10	10	10	10	9.9	9.9
Soma	21	20	21	20	20	21	21	21	21	21	20	20.6	9.8
Diagrama de Classes													
R9	2	2	2	2	2	2	2	2	2	2	1	1.9	9.5
R10	3	3	3	3	3	3	3	3	3	3	3	3	10
R11	4	3	4	4	4	3	4	3	3	3	3	3.4	8.5
R12	4	4	4	4	4	4	4	3	4	3	4	3.8	9.5
R13	4	4	4	4	4	4	4	4	4	4	4	4	10
Soma	17	16	17	17	17	16	17	15	16	15	15	16.1	9.5
Diagrama de Sequência													
R14	3	3	3	3	3	3	3	3	3	3	3	3	10
R15	4	3	4	4	3	2	4	3	2	4	3	3.2	8
R16	4	4	4	4	4	4	4	4	4	4	4	4	10
Soma	11	10	11	11	10	9	11	10	9	11	10	10.2	9.3
Soma Total das Execuções													
Soma	64	61	64	63	62	61	64	61	61	61	60	61.8	
Escala 0-10		9.5	10	9.8	9.7	9.5	10	9.5	9.5	9.5	9.4	9.6	

5.2.1 Visão geral e estabilidade

O ChatGPT Plus apresentou o melhor desempenho geral do estudo, com **nota média geral 9.6** e alta estabilidade entre execuções. Na soma bruta (máximo 64), a média foi de **61.8** pontos, com variação pequena: a melhor execução atingiu 64 e a pior 60 (Tabela 8).

Esse resultado indicou comportamento consistente e baixo risco de degradação abrupta entre execuções.

5.2.2 Desempenho por artefato

No recorte por artefato, o modelo manteve desempenho elevado em todos os diagramas: **DCU = 9.9**, **DT = 9.8**, **DC = 9.5** e **DS = 9.3**. Em termos práticos, o modelo foi capaz de preservar rastreabilidade e coerência ao longo de todo o mapeamento, com pequenas perdas concentradas em regras mais sensíveis a decisões relacionais.

5.2.3 Principais dificuldades por regra

Mesmo com desempenho alto, as quedas relativas ocorreram principalmente em regras que exigiram decisões mais finas:

- **R15 (mensagens no DS)**: embora a nota tenha permanecido alta, essa regra foi estruturalmente a mais propensa a erros por depender da identificação correta de origem, destino e ordem das mensagens.
- **R11 (operações em classes no DC)**: o modelo manteve bom desempenho; contudo, as perdas observadas ocorreram principalmente pela não identificação da operação correspondente à própria atividade de subprocesso (*Call Activity*).

5.2.4 Melhor execução e exemplos visuais

Para facilitar a interpretação qualitativa, foram inseridos no corpo do texto os três diagramas (DCU, DC e DS) correspondentes à melhor execução do modelo, enquanto a descrição textual completa foi mantida nos anexos. As Figuras 8, 9 e 10 devem representar a melhor execução (**E2**) do ChatGPT Plus.



Figura 8: DCU gerado na melhor execução do ChatGPT Plus.

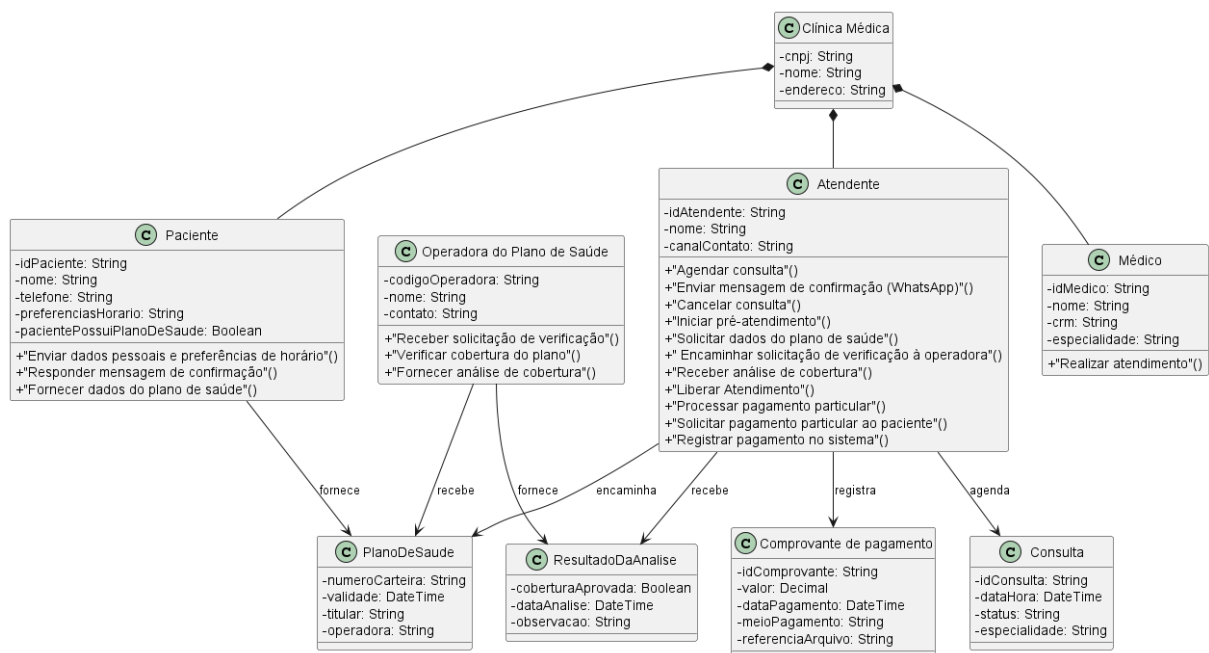


Figura 9: DC gerado na melhor execução do ChatGPT Plus.

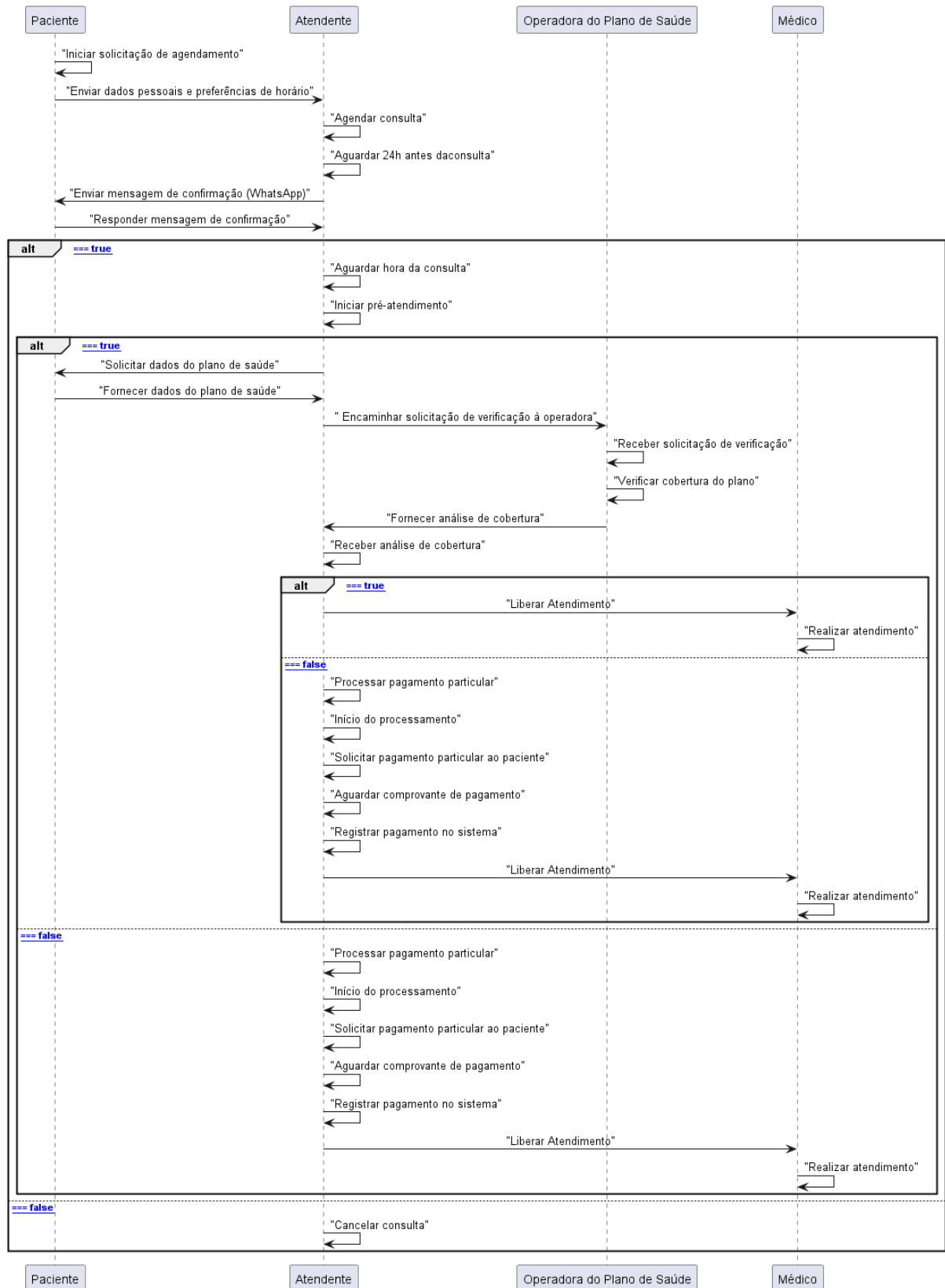


Figura 10: DS gerado na melhor execução do ChatGPT Plus.

5.2.5 Influência do formato de entrada

Com base nas 10 execuções do ChatGPT Plus (Tabela 9), avaliou-se o impacto de duas decisões de entrada: envio do prompt em mensagens separadas vs. envio único; e BPMN em texto vs. arquivo (.xml) vs. .zip.

Tabela 9: Execuções do ChatGPT Plus e respectivas configurações.

Execução	Mensagens	Formato XML	Nota Média
E1	separado	texto	9.5
E2	separado	texto	10.0
E3	junto	texto	9.8
E4	junto	texto	9.7
E5	separado	texto	9.5
E6	separado	texto	10.0
E7	separado	.zip	9.5
E8	separado	arquivo	9.5
E9	separado	texto	9.5
E10	separado	texto	9.4

No critério **mensagens separadas vs. envio único**, o envio único (E3–E4) apresentou média ligeiramente maior (9.8) em comparação às mensagens separadas (média 9,6 considerando E1, E2, E5–E10). Apesar dessa diferença, o desempenho permaneceu alto e estável em ambos os formatos, indicando baixa sensibilidade do modelo a essa variação.

Quanto ao **formato do XML**, a entrada em **texto** concentrou a maior parte dos testes (8/10) e obteve média de 9.7. As execuções com **.zip** (E7) e com **arquivo** (E8) resultaram em 9.5, sem evidenciar ganho em relação ao texto. Na prática, para este modelo, **colar o XML no chat** mostrou-se suficiente e consistente, e o uso de arquivo/.zip não trouxe benefício mensurável nas notas gerais.

Além disso, observou-se uma diferença relevante de **tempo de execução**. Nas duas execuções em que o BPMN foi enviado como arquivo, os tempos por etapa foram consideravelmente maiores: para o envio em **.zip**, registraram-se 13m 38s + 8m 21s + 9m 42s; e para o envio em **.xml**, 13m 57s + 9m 48s + 6m 16s. Em contraste, quando o BPMN foi fornecido em **texto**, os tempos ficaram **próximos de 2 minutos** por etapa. Assim, mesmo sem ganho nas notas, o uso de arquivos implicou aumento expressivo de

tempo, reduzindo a eficiência do processo.

Assim, para o ChatGPT Plus, a estratégia recomendada de entrada é: manter o BPMN em **texto** e, quando conveniente, enviar em **mensagem única** (sem necessidade de fragmentação), uma vez que os resultados não se degradaram e a média foi ligeiramente superior.

5.3 CHATGPT FREE

Tabela 10: Pontuação do ChatGPT Free por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Diagrama de Casos de Uso													
R1	3	3	2	3	3	2	3	3	3	2	3	2.7	9.0
R2	4	2	1	2	1	2	3	4	2	0	3	2	5.0
R3	4	3	2	1	3	3	4	4	4	3	4	3.1	7.8
R4	4	2	2	1	3	4	4	4	0	3	4	2.7	6.8
Soma	15	10	7	7	10	11	14	15	9	8	14	10.5	7.1
Descrição Textual dos Casos de Uso													
R5	5	5	5	5	5	4	4	3	3	5	5	4.4	8.8
R6	2	2	2	2	2	2	1	2	2	2	2	1.9	9.5
R7	4	2	4	3	2	3	3	3	2	3	3	2.8	7.0
R8	10	10	10	8	10	10	10	10	8	9	9	9.4	9.4
Soma	21	19	21	18	19	19	18	18	15	19	19	18.5	8.7
Diagrama de Classes													
R9	2	1	1	1	2	2	1	2	2	2	2	1.6	8.0
R10	3	3	3	1	3	3	3	3	3	3	3	2.8	9.3
R11	4	2	2	2	3	3	2	4	3	2	2	2.5	6.3
R12	4	2	0	2	2	2	3	3	3	3	3	2.3	5.8
R13	4	2	4	2	4	4	4	4	4	4	4	3.6	9.0
Soma	17	10	10	8	14	14	13	16	15	14	14	12.8	7.7
Diagrama de Sequência													
R14	3	3	2	3	3	3	3	3	3	2	3	2.8	9.3
R15	4	2	2	2	1	1	2	1	1	1	2	1.5	3.8
R16	4	4	4	3	4	4	4	4	4	4	2	3.7	9.3
Soma	11	9	8	8	8	8	9	8	8	7	7	8	7.4
Soma Total das Execuções													
Soma	64	48	46	41	51	52	54	57	47	48	54	49.8	

Tabela 10: Pontuação do ChatGPT Free por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Escala 0-10		7.5	7.2	6.4	8.0	8.1	8.4	8.9	7.3	7.5	8.4	7.7	

5.3.1 Visão geral e estabilidade

O ChatGPT Free obteve **nota média geral 7.7**, com maior variabilidade entre execuções quando comparado aos melhores modelos. Na soma bruta (máximo 64), a média foi de **49.8** pontos, com amplitude bem maior: melhor execução 57 e pior 41 (Tabela 10). Essa variação reforçou a importância de múltiplas execuções e do uso de média para reduzir o efeito de respostas inconsistentes.

5.3.2 Desempenho por artefato

Por artefato, o modelo apresentou **DCU = 7,1**, **DT = 8,7**, **DC = 7,7** e **DS = 7,4**. O melhor desempenho relativo apareceu na descrição textual, enquanto os artefatos que exigiram consistência estrutural mostraram maior sensibilidade a falhas de rastreabilidade e decisões equivocadas.

5.3.3 Principais dificuldades por regra

Dois pontos se destacaram:

- **R15 (mensagens no DS)**: o modelo apresentou o pior desempenho relativo nesta regra (Tabela 18), o que explicou a redução do DS mesmo enquanto R14 e R16 apresentaram resultados ótimos.
- **R2 (casos de uso no DCU)**: essa regra concentrou decisões sobre cobertura completa, ator correto e, principalmente, agrupamento semântico. A maior dificuldade esteve em agrupar *Tasks* de mesmo objetivo em um único Caso de Uso, o que ocorreu em apenas 3 das 10 execuções.

5.3.4 Melhor execução e exemplos visuais

Foram inseridos, a seguir, os diagramas (DCU, DC e DS) correspondentes à melhor execução (**E7**) obtida com o ChatGPT Free. A descrição textual completa foi mantida

nos anexos.

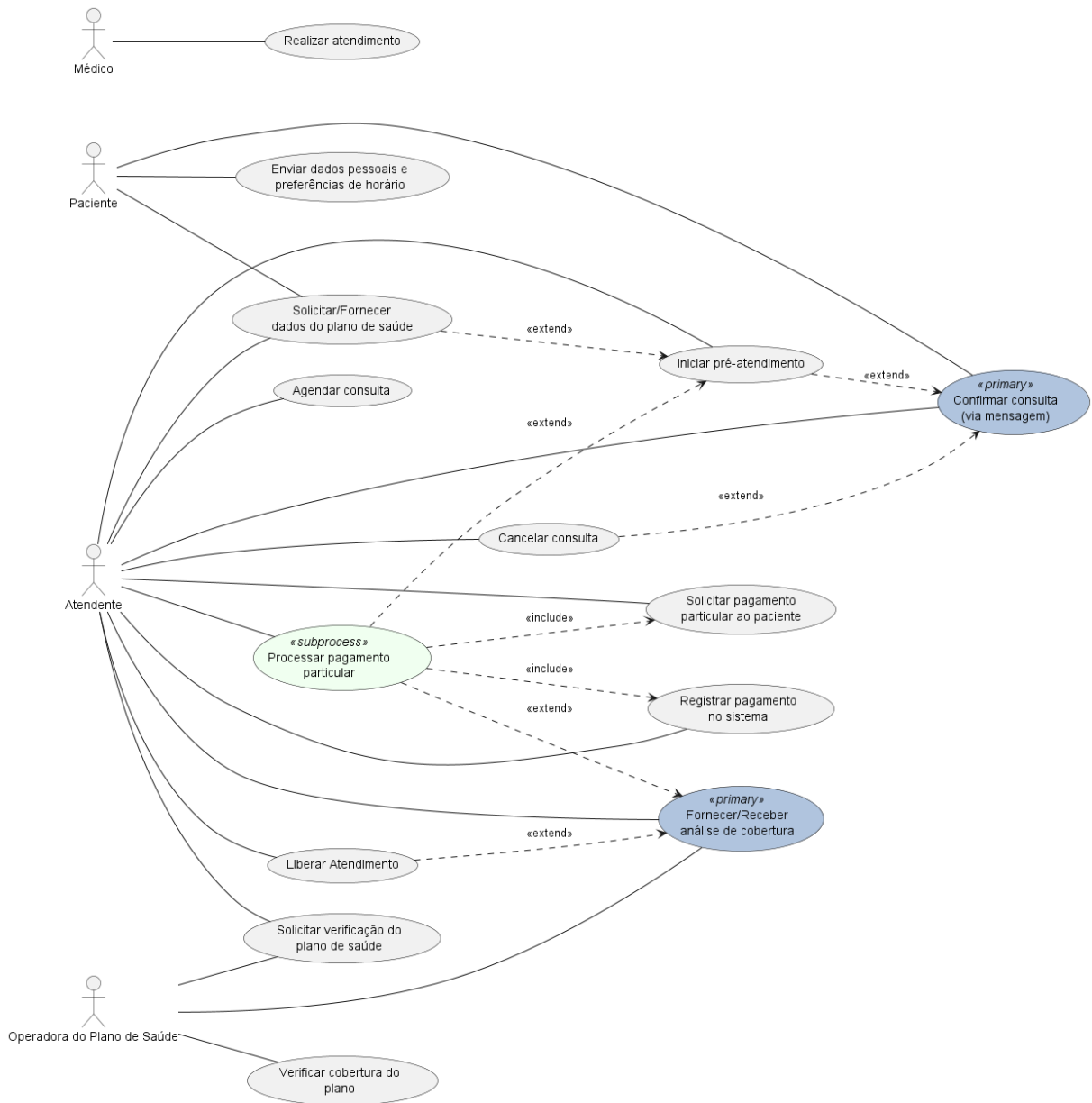


Figura 11: DCU gerado na melhor execução do ChatGPT Free.

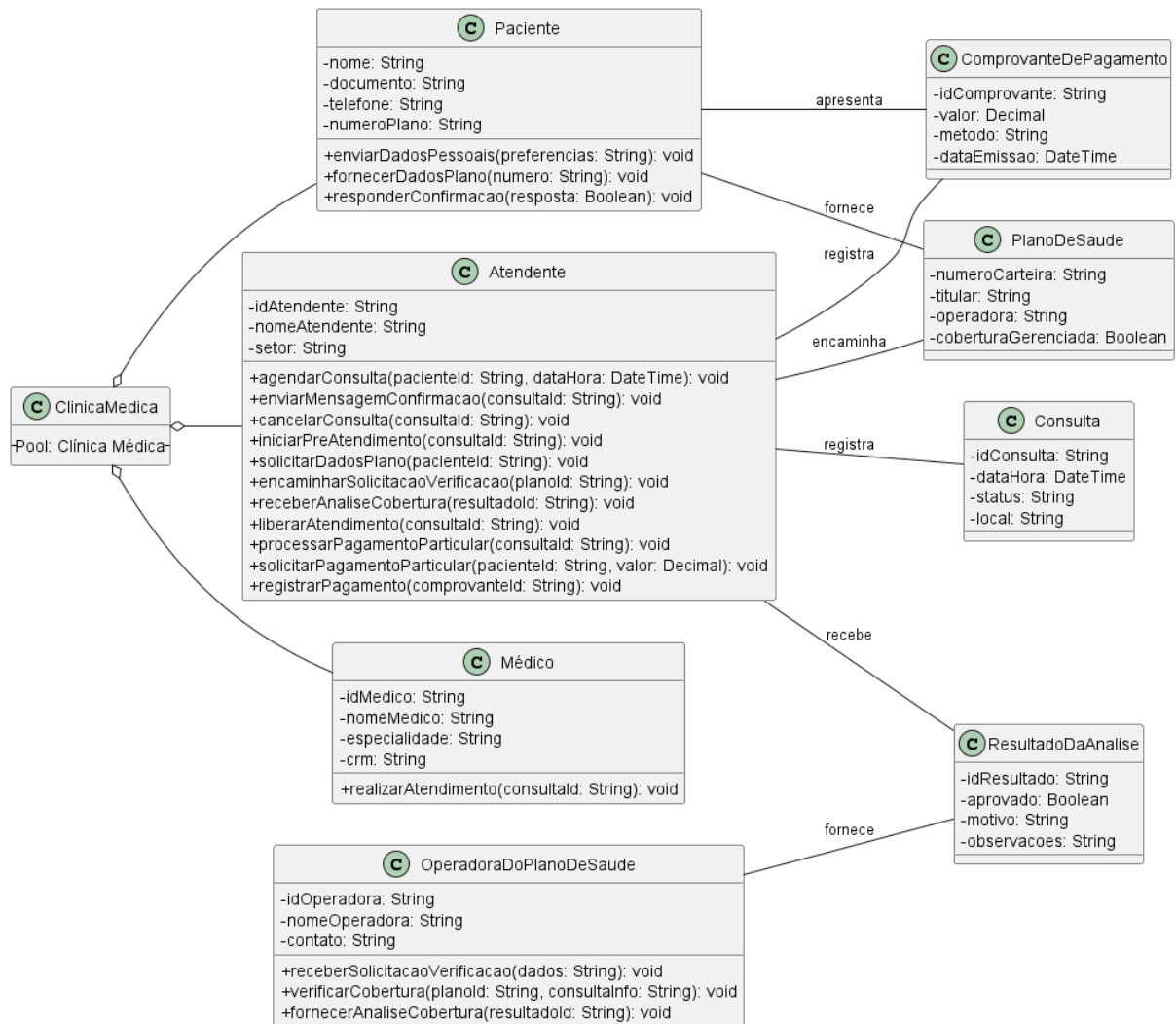


Figura 12: DC gerado na melhor execução do ChatGPT Free.

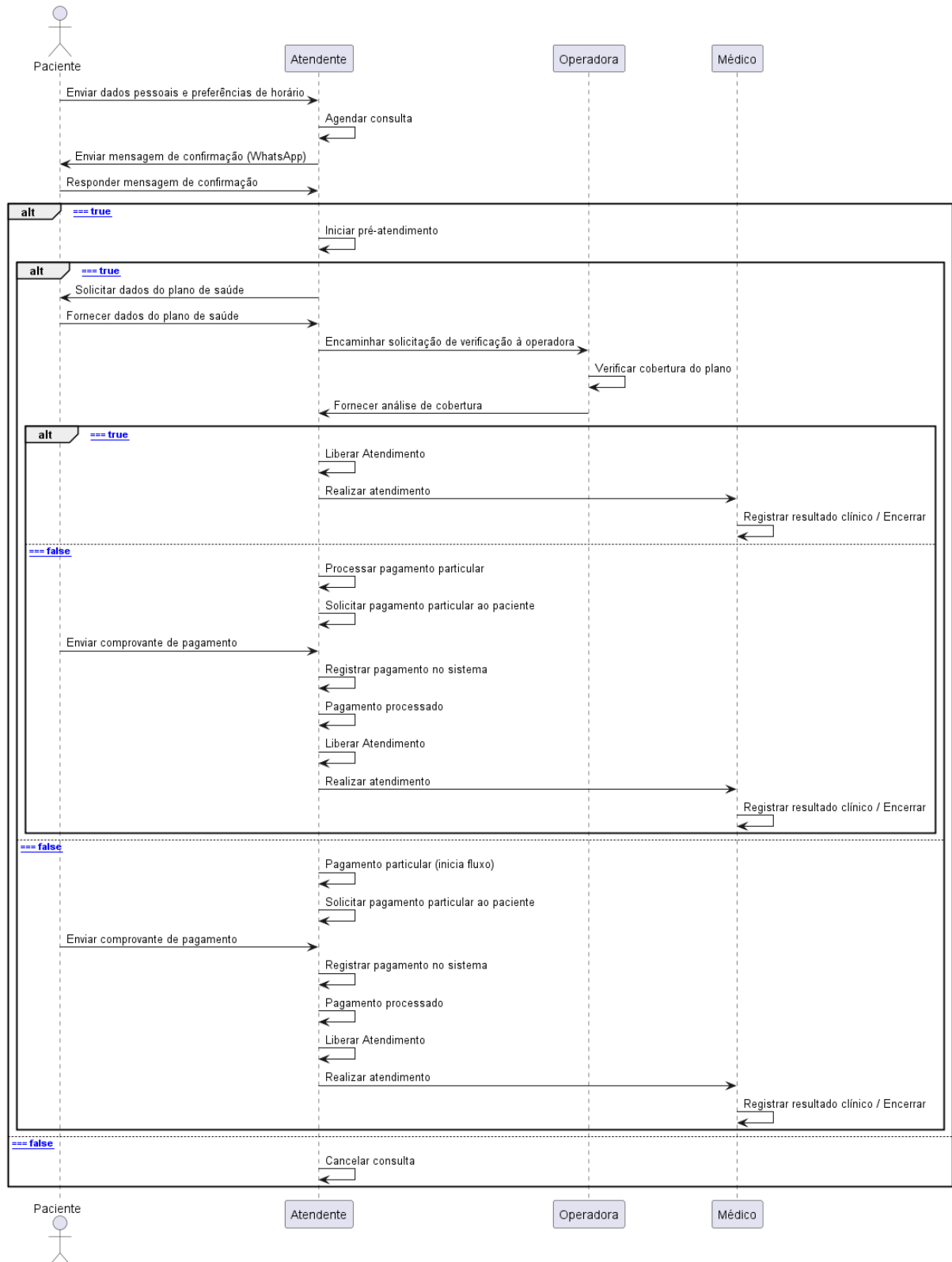


Figura 13: DS gerado na melhor execução do ChatGPT Free.

5.3.5 Influência do formato de entrada

Com base nas 10 execuções do ChatGPT Free (Tabela 11), avaliou-se o impacto de duas decisões de entrada: envio do prompt em mensagens separadas vs. envio único; e BPMN em texto vs. arquivo (.xml) vs. .zip.

Tabela 11: Execuções do ChatGPT Free e respectivas configurações.

Execução	Mensagens	Formato XML	Nota Média
E1	separado	texto	7.5
E2	junto	texto	7.2
E3	junto	texto	6.4
E4	junto	texto	8.0
E5	separado	texto	8.1
E6	separado	texto	8.4
E7	separado	texto	8.9
E8	separado	.zip	7.3
E9	separado	arquivo	7.5
E10	junto	arquivo	8.4

No critério **mensagens separadas vs. envio único**, as execuções com mensagens separadas (E1, E5–E9) apresentaram média superior (7.95) em comparação ao envio único (E2–E4, E10), cuja média foi 7.50. Isso sugere que, no ChatGPT Free, fragmentar o prompt em partes pode reduzir perda de contexto e contribuir para resultados mais consistentes.

Quanto ao **formato do XML**, a entrada em **texto** foi a mais frequente (7/10) e obteve média de 7.79. O envio em **.zip** (E8) resultou em 7.30, e o envio como **arquivo** (E9–E10) obteve média de 7.95. Entretanto, o formato de arquivo e o .zip não foram repetidos muitas vezes porque o **tempo de execução aumentava muito**, e, nas poucas tentativas realizadas, esse aumento já se mostrou **prejudicial** para a eficiência do processo. Por isso, a estratégia predominante foi manter o BPMN em **texto**, que apresentou execução mais ágil.

Um aspecto operacional relevante é que as **três primeiras execuções (E1–E3)** foram realizadas **sem a ferramenta “pensando”** do modelo. Nesses casos, observou-se maior instabilidade de contexto: foi necessário **reenviar as regras e o BPMN (XML) diversas vezes**, pois o modelo frequentemente indicava que as informações **não estavam**

mais no contexto e “esquecia” conteúdos já fornecidos. Nas execuções seguintes (E4–E10), com o recurso ativado, essa limitação foi reduzida, tornando o processo mais fluido e exigindo menos reenvios.

Assim, para o ChatGPT Free, a estratégia recomendada de entrada é priorizar o envio em **mensagens separadas** e manter o BPMN em **texto**; além disso, quando disponível, utilizar o modo com “**pensando**” habilitado para minimizar perdas de contexto e retrabalho.

5.4 GEMINI PRO

Tabela 12: Pontuação do Gemini Pro por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Diagrama de Casos de Uso													
R1	3	3	3	3	3	3	3	3	3	3	3	3	10.0
R2	4	4	3	3	3	3	4	4	3	4	4	3.5	8.8
R3	4	4	4	4	4	4	4	4	4	4	4	4	10.0
R4	4	4	2	3	3	4	4	4	4	4	4	3.6	9.0
Soma	15	15	12	13	13	14	15	15	14	15	15	14.1	9.4
Descrição Textual dos Casos de Uso													
R5	5	5	4	2	5	5	5	5	3	5	4	4.3	8.6
R6	2	2	1	2	2	2	2	2	2	2	1	1.8	9.0
R7	4	2	1	2	0	2	4	4	4	4	3	2.6	6.5
R8	10	10	8	8	7	8	10	10	9	10	8	8.8	8.8
Soma	21	19	14	14	14	17	21	21	18	21	16	17.5	8.2
Diagrama de Classes													
R9	2	2	1	2	2	2	2	2	2	2	2	1.9	9.5
R10	3	3	0	3	3	3	3	3	3	3	3	2.7	9.0
R11	4	3	4	3	4	3	3	3	3	4	4	3.4	8.5
R12	4	2	3	3	1	3	1	2	4	1	3	2.3	5.8
R13	4	3	4	3	3	3	3	3	3	3	3	3.1	7.8
Soma	17	13	12	14	13	14	12	13	15	13	15	13.4	8.1
Diagrama de Sequência													
R14	3	3	3	3	3	3	3	3	3	3	3	3	10
R15	4	1	2	3	2	1	1	2	1	2	2	1.7	4.3
R16	4	4	4	4	3	4	4	4	4	4	4	3.9	9.8
Soma	11	8	9	10	8	8	8	9	8	9	9	8.6	8.0

Tabela 12: Pontuação do Gemini Pro por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Soma Total das Execuções													
Soma	64	55	47	51	48	53	56	58	55	58	55	53.6	
Escala	0-10	8.6	7.3	8.0	7.5	8.3	8.8	9.1	8.6	9.1	8.6	8.4	

5.4.1 Visão geral e estabilidade

O Gemini Pro apresentou **nota média geral 8.4**. Na soma bruta (máximo 64), a média foi de **53.6** pontos, com variação intermediária: melhor execução 58 e pior 47 (Tabela 12). Em comparação ao ChatGPT Free e ao Gemini Free, observou-se maior consistência, ainda que com oscilações relevantes.

5.4.2 Desempenho por artefato

Por artefato, os resultados foram **DCU = 9.4**, **DT = 8.2**, **DC = 8.1** e **DS = 8.0**. O destaque positivo foi o DCU, com desempenho elevado e próximo aos melhores modelos. Em contrapartida, a DT, apesar de não ter sido o pior artefato do Gemini Pro, apresentou a menor nota entre todos os modelos avaliados, enquanto o DS foi o artefato de menor desempenho dentro do próprio modelo, concentrando as maiores perdas relativas. Esse padrão sugere maior dificuldade tanto em consolidar adequadamente informações textuais quanto em manter coerência no encadeamento e na direção das mensagens.

5.4.3 Principais dificuldades por regra

Os gargalos mais claros foram:

- **R15 (mensagens no DS)**: o desempenho fraco em R15 puxou o DS para baixo, pois essa regra concentrou origem/destino/ordem e cobertura de mensagens esperadas.
- **R12 (classes a partir de Data Objects)**: a nota inferior indicou dificuldade em evitar classes duplicadas originadas de diferentes elementos do BPMN, afetando diretamente o DC.

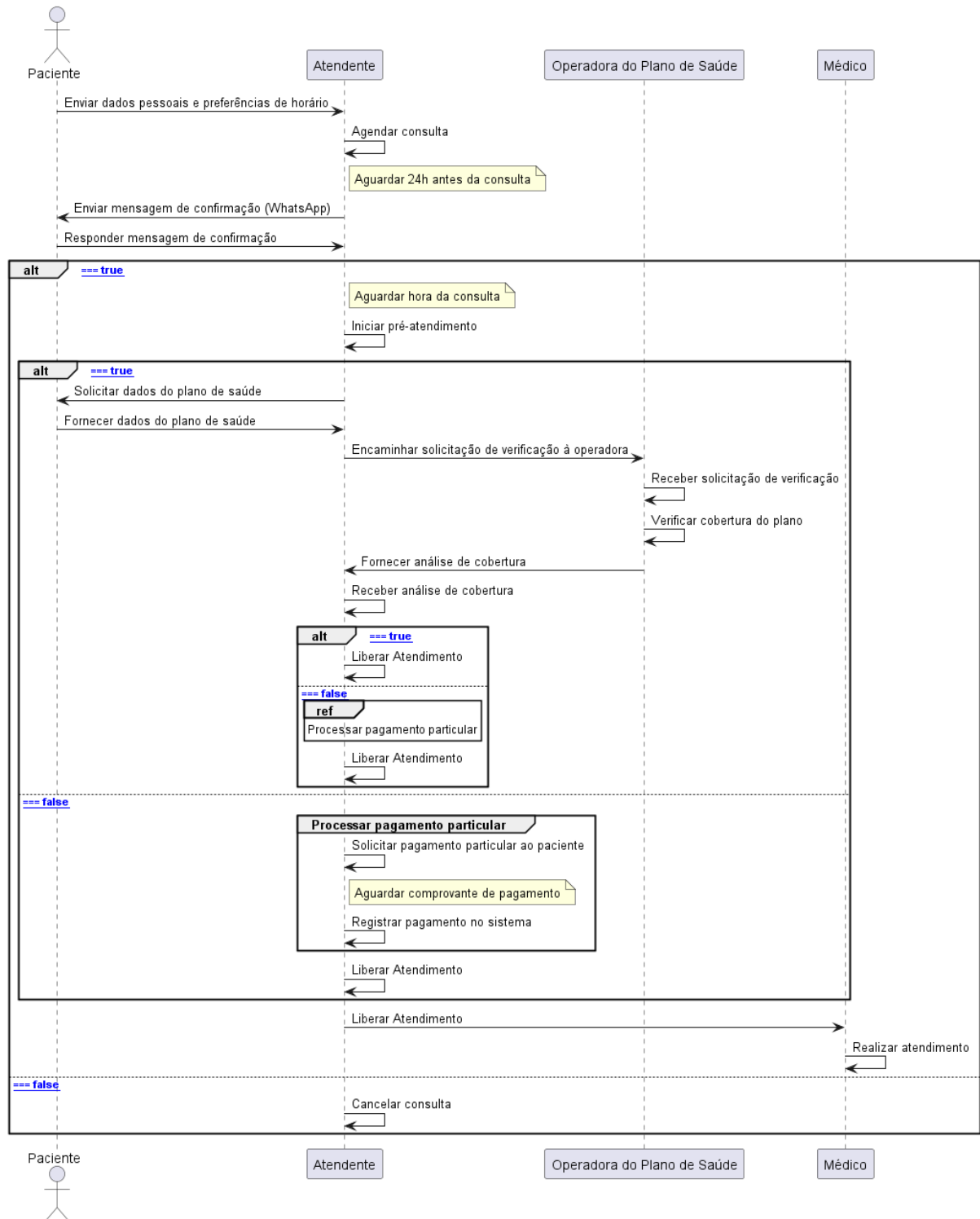


Figura 16: DS gerado na melhor execução do Gemini Pro.

5.4.5 Influência do formato de entrada

Com base nas 10 execuções do Gemini Pro (Tabela 13), avaliou-se o impacto de duas decisões de entrada: envio do prompt em mensagens separadas vs. envio único; e BPMN

em texto vs. arquivo (.xml). Neste modelo, não houve execuções com **.zip**.

Tabela 13: Execuções do Gemini Pro e respectivas configurações.

Execução	Mensagens	Formato XML	Nota Média
E1	separado	texto	8.6
E2	separado	texto	7.3
E3	junto	arquivo	8.0
E4	separado	texto	7.5
E5	separado	texto	8.3
E6	separado	texto	8.8
E7	separado	arquivo	9.1
E8	junto	arquivo	8.6
E9	junto	arquivo	9.1
E10	junto	arquivo	8.6

No critério **mensagens separadas vs. envio único**, o envio único (E3, E8–E10) apresentou média maior (8.58) em comparação às mensagens separadas (média 8.27 considerando E1, E2, E4–E7). Isso indica que, para o Gemini Pro, consolidar o envio pode favorecer um desempenho ligeiramente melhor.

Quanto ao **formato do XML**, as execuções com BPMN em **arquivo** (E3, E7–E10) obtiveram média de 8.68, superior às execuções em **texto** (E1, E2, E4–E6), com média de 8.10. Assim, neste modelo, o envio do BPMN como **arquivo (.xml)** esteve associado a melhores notas gerais em comparação ao XML colado no chat.

Assim, para o Gemini Pro, a estratégia recomendada de entrada é: preferir o BPMN em **arquivo (.xml)** e, quando possível, enviar o prompt em **mensagem única**.

5.5 GEMINI FREE

Tabela 14: Pontuação do Gemini Free por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Diagrama de Casos de Uso													
R1	3	3	3	3	3	3	1	3	0	3	3	2.5	8.3
R2	4	4	1	3	3	4	0	3	0	3	3	2.4	6.0
R3	4	4	1	4	4	4	1	4	1	4	4	3.1	7.8

Tabela 14: Pontuação do Gemini Free por regra e por execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
R4	4	4	1	4	4	4	0	4	0	4	4	2.9	7.3
Soma	15	15	6	14	14	15	2	14	1	14	14	10.9	7.3
Descrição Textual dos Casos de Uso													
R5	5	5	3	5	5	4	5	5	5	4	5	4.6	9.2
R6	2	1	1	1	2	2	2	1	2	2	2	1.6	8.0
R7	4	3	3	4	4	3	1	4	4	4	4	3.4	8.5
R8	10	5	6	6	7	8	9	8	8	10	10	7.7	7.7
Soma	21	14	13	16	18	17	17	18	19	20	21	17.3	8.4
Diagrama de Classes													
R9	2	1	0	2	2	2	0	2	0	2	2	1.3	6.5
R10	3	0	0	3	3	3	3	2	2	3	3	2.2	7.3
R11	4	3	1	3	3	3	2	3	1	3	2	2.4	6.0
R12	4	2	1	2	3	1	0	4	1	2	2	1.8	4.5
R13	4	3	3	3	3	3	3	3	3	3	3	3	7.5
Soma	17	9	5	13	14	12	8	14	7	13	12	10.7	6.4
Diagrama de Sequência													
R14	3	3	2	3	3	3	1	3	0	3	3	2.4	8.0
R15	4	2	1	1	2	3	1	3	1	3	1	1.8	4.5
R16	4	4	3	4	4	4	1	4	0	4	4	3.2	8.0
Soma	11	9	6	8	9	10	3	10	1	10	8	7.4	6.8
Soma Total das Execuções													
Soma	64	47	30	51	55	54	30	56	28	57	55	46.3	
Escala 0-10		7.3	4.7	8.0	8.6	8.4	4.7	8.8	4.4	8.9	8.6	7.2	

5.5.1 Visão geral e estabilidade

O Gemini Free apresentou a menor nota média geral, com **7.2** (Tabela 18), além da maior variabilidade entre execuções. Na soma bruta (máximo 64), a média foi de **46.3** pontos, com amplitude muito alta: melhor execução 57 e pior 28 (Tabela 14). Essa instabilidade tornou o uso operacional mais arriscado sem replicação e validação manual.

5.5.2 Desempenho por artefato

Por artefato, os resultados foram $DCU = 7.3$, $DT = 8.4$, $DC = 6.4$ e $DS = 6.8$. Dois padrões apareceram com clareza: (i) a DT foi relativamente resistente, mantendo nota acima de 8; e (ii) DC e DS foram os mais penalizados, indicando dificuldade em tarefas que exigiram consistência estrutural e rastreabilidade fina.

5.5.3 Principais dificuldades por regra

Os principais problemas concentraram-se em:

- **R12 (classes a partir de Data Objects)**: a regra apresentou uma das menores notas do modelo, o que explicou a queda do DC para 6,4.
- **R15 (mensagens no DS)**: a regra também apresentou desempenho baixo, impactando diretamente o DS.

Na prática, isso sugeriu que o Gemini Free sofreu mais quando precisou relacionar elementos entre si (origem/destino de mensagens; equivalência e consistência de classes), mesmo quando regras mais diretas não foram o problema principal.

5.5.4 Melhor execução e exemplos visuais

Mesmo com instabilidade, os diagramas da melhor execução (**E9**) foram registrados como referência visual. Eles evidenciam que o Gemini Free pôde alcançar resultados competitivos em execuções específicas, embora sem consistência entre repetições.

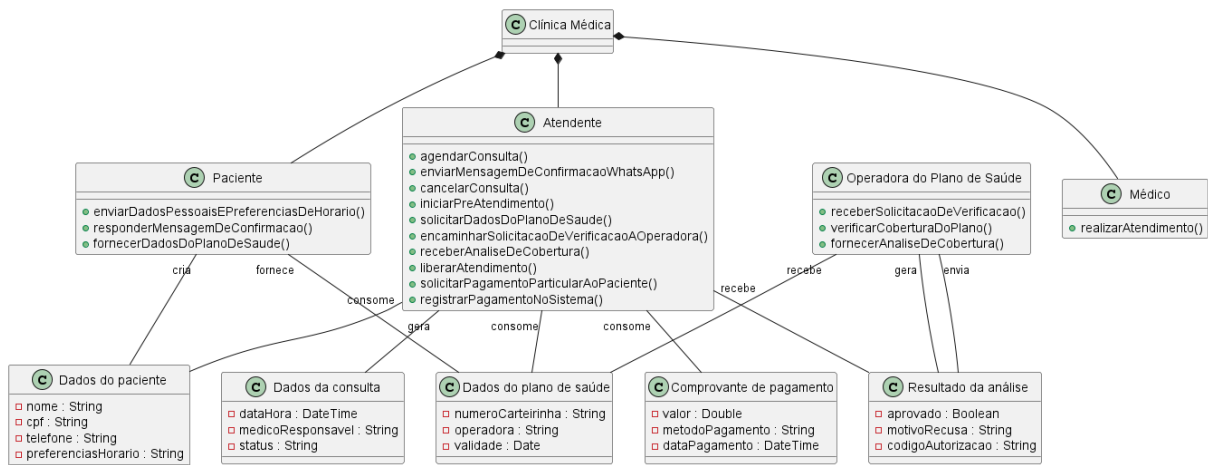


Figura 17: DC gerado na melhor execução do Gemini Free.



Figura 18: DCU gerado na melhor execução do Gemini Free.

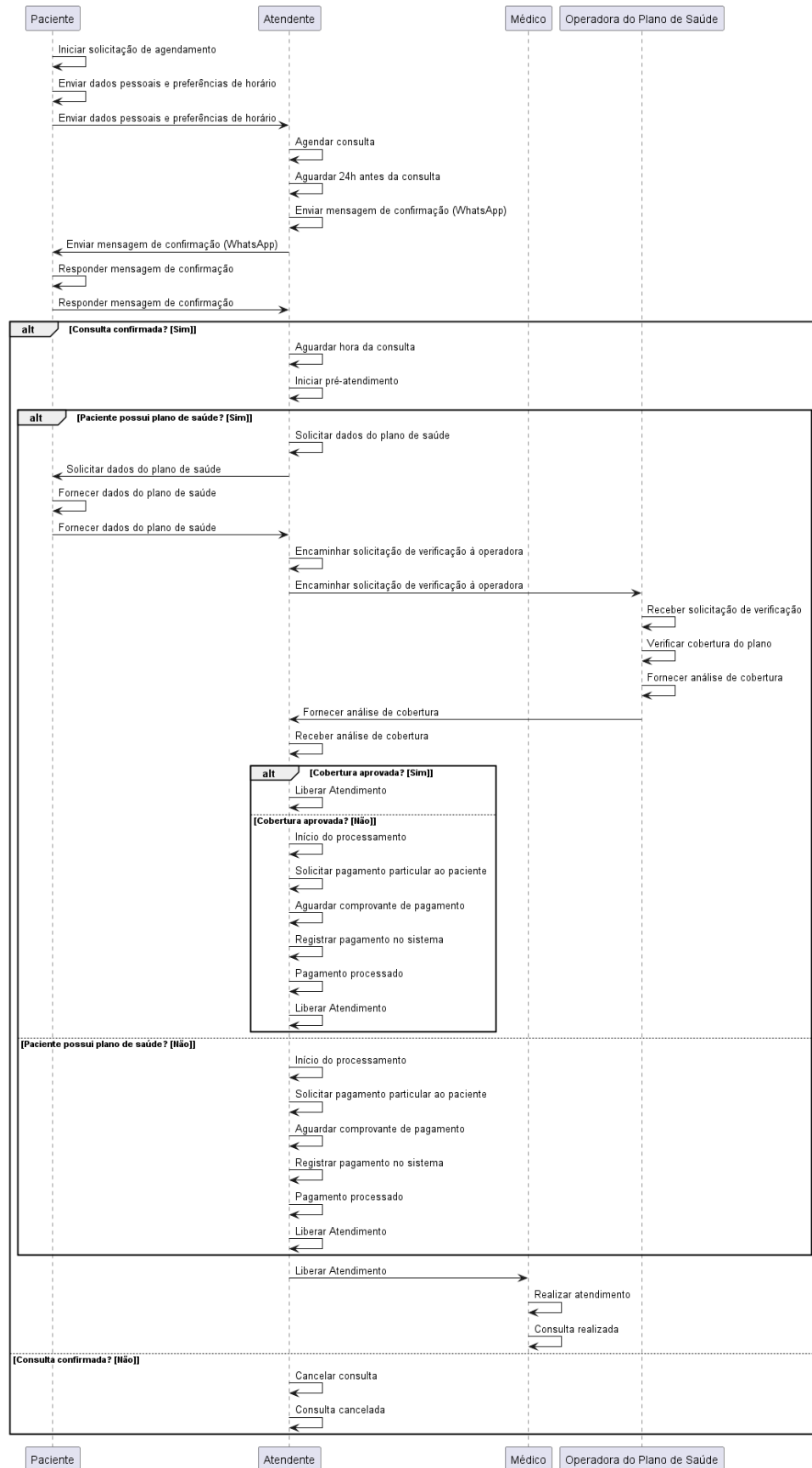


Figura 19: DS gerado na melhor execução do Gemini Free.

5.5.5 Influência do formato de entrada

Com base nas 10 execuções do Gemini Free (Tabela 15), avaliou-se o impacto de duas decisões de entrada: envio do prompt em mensagens separadas vs. envio único; e BPMN em texto vs. arquivo (.xml) vs. .zip.

Tabela 15: Execuções do Gemini Free e respectivas configurações.

Execução	Mensagens	Formato XML	Nota Média
E1	separado	texto	7.3
E2	junto	texto	4.7
E3	separado	texto	8.0
E4	separado	texto	8.6
E5	separado	texto	8.4
E6	junto	arquivo	4.7
E7	separado	.zip	8.8
E8	junto	.zip	4.4
E9	separado	arquivo	8.9
E10	separado	arquivo	8.6

No critério **mensagens separadas vs. envio único**, as execuções com mensagens separadas (E1, E3–E5, E7, E9–E10) apresentaram média significativamente maior (8.09) em comparação ao envio único (E2, E6, E8), cuja média foi 4.60. Na prática, no Gemini Free, o envio único esteve associado a quedas acentuadas de desempenho, enquanto a fragmentação do prompt se mostrou decisiva para manter resultados aceitáveis.

Quanto ao **formato do XML**, a entrada em **texto** (E1–E5) obteve média de 7.40. O envio em **.zip** (E7–E8) teve média de 6.60, puxada para baixo principalmente pela execução em envio único (E8). Já o envio como **arquivo** (E6, E9–E10) apresentou média de 7.40, mas também com forte dependência do formato de mensagens (E6, com envio único, foi o caso de pior desempenho dentro desse grupo). Ou seja, neste modelo, a variável que mais pesou foi **mensagens separadas vs. envio único**, mais do que o tipo de mídia do XML.

Assim, para o Gemini Free, a estratégia recomendada de entrada é priorizar o envio em **mensagens separadas**, independentemente do formato do BPMN, pois essa escolha foi determinante para evitar degradações severas de desempenho.

5.6 CLAUDE FREE

Tabela 16: Pontuação do Claude Free por Regra e por Execução.

Regra	Nota Máxima	Nota por execução										Nota Média	
		E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	0-máx	0-10
Diagrama de Casos de Uso													
R1	3	3	3	3	3	3	3	3	3	3	3	3.0	10
R2	4	3	3	3	4	3	3	3	3	3	2	3.0	7.5
R3	4	4	4	4	4	4	4	4	4	4	3	3.9	9.8
R4	4	3	4	4	3	4	4	4	4	4	4	3.8	9.5
SOMA	15	13	14	14	14	14	14	14	14	14	12	13.7	9.2
Descrição Textual dos Casos de Uso													
R5	5	4	4	4	5	5	5	5	5	5	5	4.7	9.4
R6	2	2	2	2	2	2	2	2	2	2	2	2.0	10
R7	4	4	4	4	4	4	4	4	4	4	4	4.0	10
R8	10	10	10	10	10	10	10	10	10	10	10	10.0	10
SOMA	21	20	20	20	21	21	21	21	21	21	21	20.7	9.9
Diagrama de Classes													
R9	2	2	2	2	2	2	1	2	2	2	2	1.9	9.5
R10	3	3	2	3	3	3	0	3	3	2	3	2.5	8.3
R11	4	4	4	3	4	3	3	4	4	3	2	3.4	8.5
R12	4	3	4	3	3	4	3	3	3	4	3	3.3	8.3
R13	4	3	4	3	3	4	4	3	3	4	4	3.5	8.8
SOMA	17	15	16	14	15	16	11	15	15	15	14	14.6	8.7
Diagrama de Sequência													
R14	3	3	3	3	3	3	3	3	3	3	3	3.0	10
R15	4	3	3	2	3	2	3	2	2	3	3	2.6	6.5
R16	4	4	4	4	4	4	4	4	4	4	4	4.0	10
SOMA	11	10	10	9	10	9	10	9	9	10	10	9.6	8.8
Soma Total das Execuções													
Soma	64	58	60	57	60	60	56	59	59	60	57	58.6	
Escala 0-10		9.1	9.4	8.9	9.4	9.4	8.8	9.2	9.2	9.4	8.9	9.1	

5.6.1 Visão geral e estabilidade

O Claude Free apresentou **nota média geral 9.1**, posicionando-se como o melhor modelo gratuito do estudo (Tabela 18). Na soma bruta (máximo 64), a média foi de **58.6** pontos, com baixa variação: melhor execução 60 e pior 56 (Tabela 16). Isso indicou

alta consistência, especialmente quando comparado aos demais modelos gratuitos.

5.6.2 Desempenho por artefato

Por artefato, os resultados foram **DCU = 9.2**, **DT = 9.9**, **DC = 8.7** e **DS = 8.8**. O maior destaque foi a DT, com desempenho praticamente máximo, evidenciando boa capacidade de estruturar pré-condições, pós-condições, gatilhos e cenários a partir do BPMN.

5.6.3 Principais dificuldades por regra

Apesar do bom desempenho geral, o padrão de dificuldade seguiu a tendência do estudo:

- **R15 (mensagens no DS)** concentrou parte relevante das perdas, mesmo que o Claude tenha apresentado desempenho superior aos demais modelos gratuitos.
- **R2 (casos de uso no DCU)** também apareceu como ponto sensível, refletindo a dificuldade em agrupar corretamente Tasks de mesmo objetivo.

5.6.4 Melhor execução e exemplos visuais

Para apoiar a interpretação qualitativa, foram inseridos no corpo do texto os diagramas (DCU, DC e DS) correspondentes à melhor execução (**E4**) do Claude Free. Como o modelo se destacou na Descrição Textual, a DT completa foi mantida nos anexos; ainda assim, seu conteúdo foi considerado representativo do desempenho elevado observado nesse artefato.

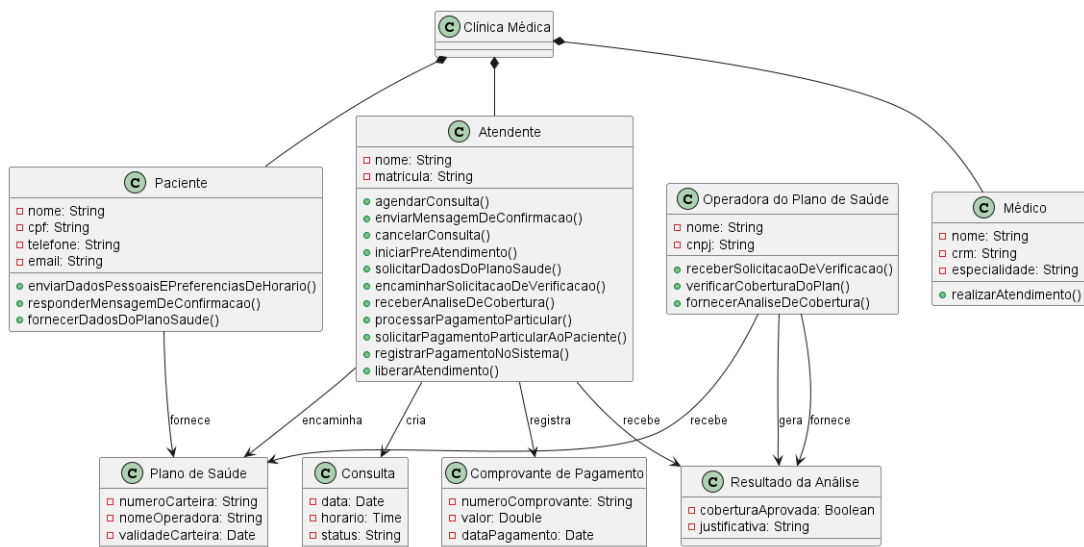


Figura 20: DC gerado na melhor execução do Claude Free.



Figura 21: DCU gerado na melhor execução do Claude Free.

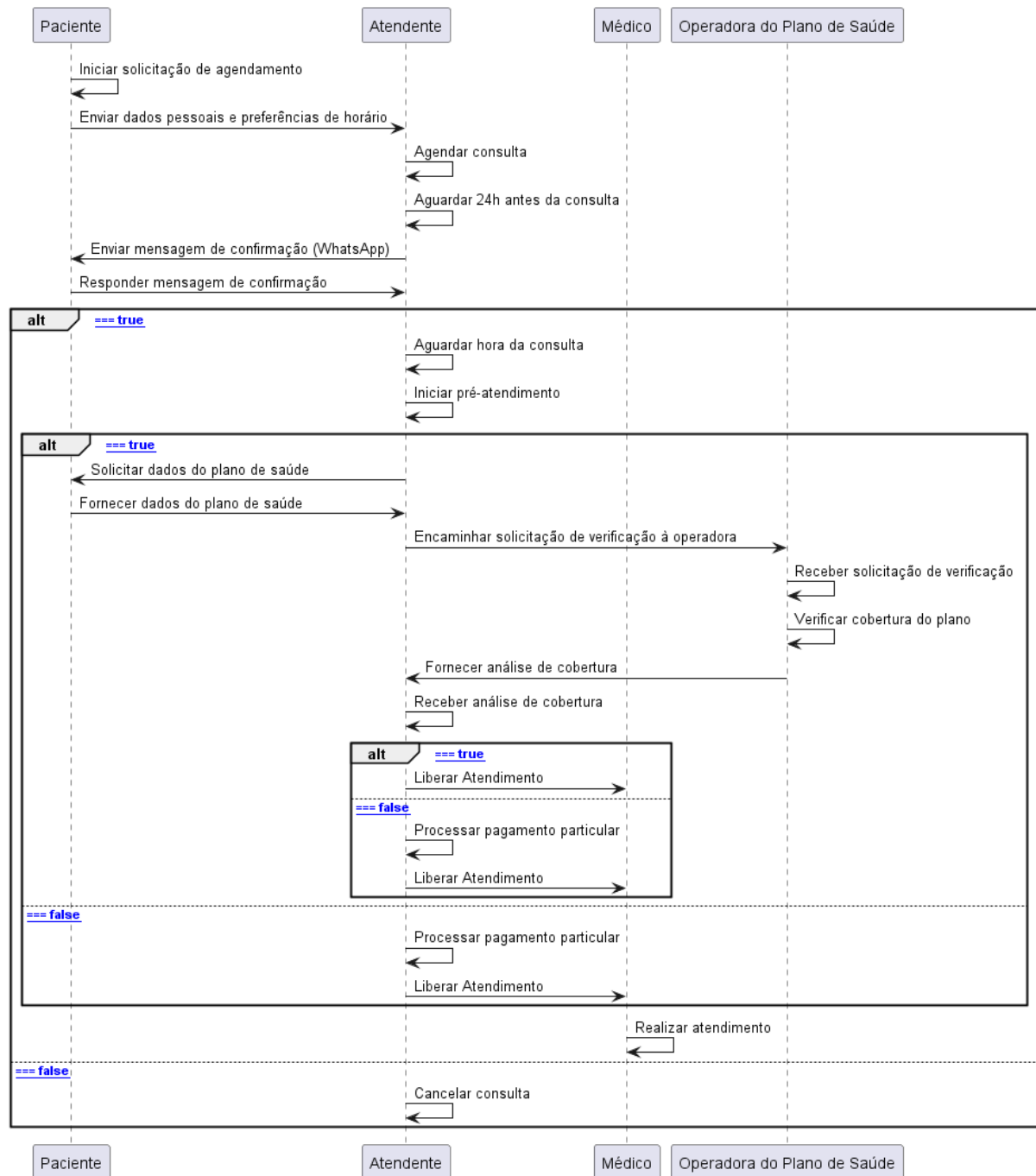


Figura 22: DS gerado na melhor execução do Claude Free.

5.6.5 Influência do formato de entrada

Com base nas 10 execuções do Claude Free (Tabela 17), avaliou-se o impacto de duas decisões de entrada: envio do prompt em mensagens separadas vs. envio único; e BPMN em texto vs. arquivo (.xml). Neste modelo, não houve execuções com **.zip**.

Tabela 17: Execuções do Claude Free e respectivas configurações.

Execução	Mensagens	Formato XML	Nota Média
E1	junto	texto	7.5
E2	junto	texto	7.2
E3	junto	texto	6.4
E4	junto	texto	8.0
E5	separado	texto	8.1
E6	separado	arquivo	8.4
E7	junto	arquivo	8.9
E8	junto	arquivo	7.3
E9	junto	arquivo	7.5
E10	junto	arquivo	8.4

No critério **mensagens separadas vs. envio único**, o envio em mensagens separadas (E5–E6) apresentou média de 8.25, enquanto o envio único (E1–E4, E7–E10) obteve média de 7.65. Apesar de a diferença existir, o Claude Free manteve desempenho relativamente estável, sem quedas abruptas associadas ao formato de envio, indicando menor sensibilidade do que modelos que degradaram fortemente no envio único.

Quanto ao **formato do XML**, as execuções com BPMN em **texto** (E1–E5) obtiveram média de 7.44. Já as execuções com BPMN em **arquivo** (E6–E10) apresentaram média de 8.10, indicando melhora quando o XML foi enviado como arquivo. Na prática, para o Claude Free, o envio em arquivo se mostrou mais vantajoso do que colar o BPMN em texto, com ganho nas notas gerais.

Assim, para o Claude Free, a estratégia recomendada de entrada é: preferir o BPMN em **arquivo (.xml)** e, quando possível, utilizar **mensagens separadas**, pois essa combinação apresentou melhores médias.

5.7 COMPARAÇÃO GLOBAL

Tabela 18: Comparação geral das médias por regra, diagrama e modelo de IA.

Regra	ChatGPT Plus	ChatGPT Free	Gemini Pro	Gemini Free	Claude Free	Média
R1	10.0	9.0	10.0	8.3	10.0	9.5
R2	9.8	5.0	8.8	6.0	7.5	7.4

Tabela 18: Comparação geral das médias por regra e por modelo de IA.

Regra	ChatGPT Plus	ChatGPT Free	Gemini Pro	Gemini Free	Claude Free	Média
R3	10.0	7.8	10.0	7.8	9.8	9.1
R4	10.0	6.8	9.0	7.3	9.5	8.5
DCU	9.9	7.1	9.4	7.3	9.2	8.6
R5	10.0	8.8	8.6	9.2	9.4	9.2
R6	10.0	9.5	9.0	8.0	10.0	9.3
R7	9.3	7.0	6.5	8.5	10.0	8.3
R8	9.9	9.4	8.8	7.7	10.0	9.2
DT	9.8	8.7	8.2	8.4	9.9	9.0
R9	9.5	8.0	9.5	6.5	9.5	8.6
R10	10.0	9.3	9.0	7.3	8.3	8.8
R11	8.5	6.3	8.5	6.0	8.5	7.6
R12	9.5	5.8	5.8	4.5	8.3	6.8
R13	10.0	9.0	7.8	7.5	8.8	8.6
DC	9.5	7.7	8.1	6.4	8.7	8.1
R14	10.0	9.3	10.0	8.0	10.0	9.5
R15	8.0	3.8	4.3	4.5	6.5	5.4
R16	10.0	9.3	9.8	8.0	10.0	9.4
DS	9.3	7.4	8.0	6.8	8.8	8.1
Média	9.6	7.7	8.4	7.2	9.1	8.4

A Tabela 18 sintetizou a comparação geral das médias (0–10) por regra, por artefato e por modelo. No geral, o ranking foi: **ChatGPT Plus (9.6) > Claude Free (9.1) > Gemini Pro (8.4) > ChatGPT Free (7.7) > Gemini Free (7.2)**. Além da média, a variabilidade observada nas tabelas por execução reforçou que modelos gratuitos, em especial o Gemini Free, apresentaram maior instabilidade, o que impactou diretamente a confiabilidade prática.

Para facilitar a visualização dessas diferenças por artefato, a Figura 23 apresenta as médias normalizadas (0–10) obtidas por cada modelo de IA em cada artefato avaliado.

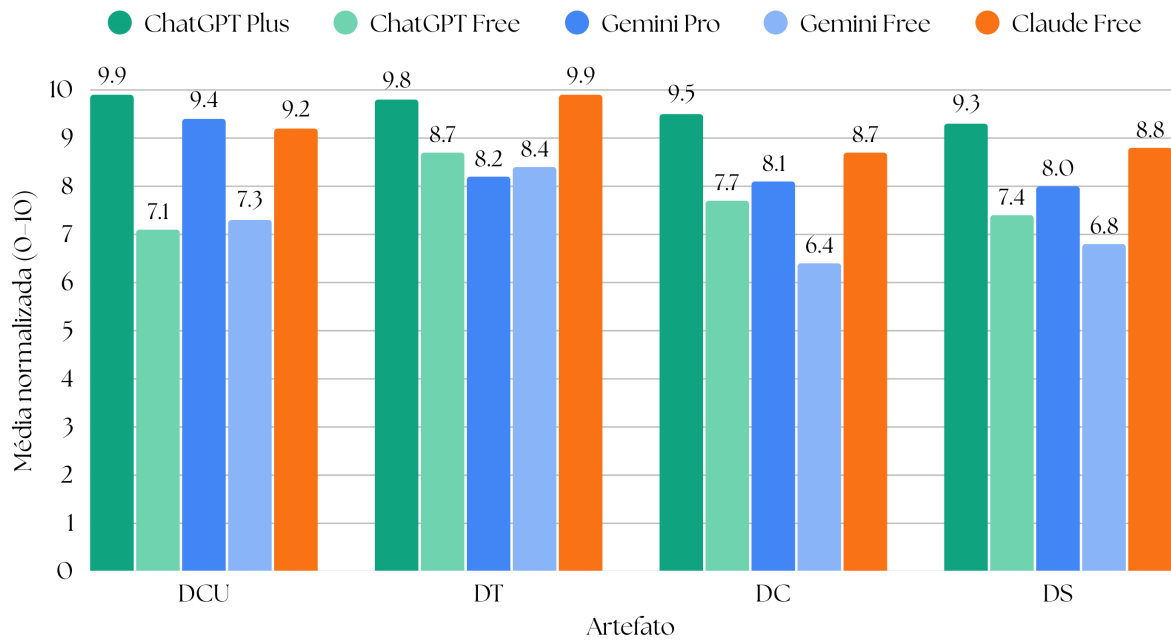


Figura 23: Comparação das médias normalizadas (0–10) por artefato e por modelo de IA.

5.7.1 Comparação por artefato

Considerando as médias agregadas por artefato (Tabela 18), a **Descrição Textual** foi o artefato com melhor desempenho médio (**DT = 9.0**), seguida do **DCU (8.6)**. Já **DC** e **DS** apresentaram as menores médias (**8.1** em ambos), indicando que os artefatos que exigiram consistência estrutural e cruzamento de relações foram os mais desafiadores para as IAs.

Esse padrão foi coerente com a natureza das regras: DT e DCU foram mais diretamente rastreáveis a elementos explícitos no BPMN, enquanto DC e DS demandaram decisões mais globais, como evitar classes duplicadas, manter coerência entre classes e garantir a direção correta das mensagens.

5.7.2 Comparação por regra

A análise por regra evidenciou com clareza quais etapas do mapeamento BPMN–UML foram mais críticas. As regras com melhor desempenho médio global incluíram: **R1** (média 9.5), **R3** (9.1), **R5** (9.2), **R6** (9.3), **R14** (9.5) e **R16** (9.4). Em geral, foram regras com correspondência direta entre elementos.

Em contrapartida, os maiores gargalos do estudo foram:

- **R15** (média global 5,4): criação, direção e ordem corretas das mensagens no DS.
- **R12** (6,8): criação consistente de classes a partir de Data Objects, evitando duplicações e mantendo coerência.
- **R2** (7,4): cobertura completa, junção de Atividades e ator correto no DCU.
- **R11** (7,6): operações posicionadas corretamente no DC.

Essas regras tiveram em comum a necessidade de raciocínio relacional: não bastou reconhecer elementos; foi necessário relacioná-los corretamente (origem/destino/ordem, equivalência semântica, consistência entre artefatos). Na prática, foi nesse ponto que mais apareceram omissões e criações indevidas para completar respostas.

5.8 IMPACTO DO FORMATO DE ENTRADA NAS EXECUÇÕES

Além da comparação por modelo, este estudo avaliou como o **formato de entrada** influenciou no desempenho das execuções. Em cada execução, variou-se: (i) envio do conteúdo em **mensagens separadas** vs. **envio único** e (ii) envio do BPMN como **XML em texto** vs. **arquivo .xml** vs. **arquivo .zip** (quando aplicável). As conclusões abaixo foram obtidas a partir das notas finais por execução (Tabelas 9, 11, 13, 15 e 17), complementadas por observações operacionais (tempo de execução e estabilidade de contexto).

No **ChatGPT Plus** (Tabela 9), o envio **único** teve média ligeiramente superior ao envio em mensagens separadas, e o BPMN em **texto** apresentou desempenho alto e estável. O envio por **arquivo/.zip** não trouxe ganho nas notas e aumentou fortemente o tempo: em .zip foram 13m 38s + 8m 21s + 9m 42s por etapa, e em .xml foram 13m 57s + 9m 48s + 6m 16s, enquanto com **texto** os tempos ficaram próximos de **2 min** por etapa. Assim, para este modelo, **texto + envio único** foi o melhor compromisso entre nota e eficiência.

No **ChatGPT Free** (Tabela 11), **mensagens separadas** produziram média superior ao envio único. Um ponto crítico foi que as execuções **E1–E3** ocorreram **sem a ferramenta “pensando”**, exigindo reenvio recorrente das regras e do XML devido a perdas de contexto. Além disso, o envio por **arquivo/.zip** foi pouco explorado porque

o **tempo de execução aumentava muito**, o que já se mostrou prejudicial nas poucas tentativas realizadas. Na prática, o padrão mais viável foi **mensagens separadas + XML em texto**.

No **Gemini Pro** (Tabela 13), observou-se vantagem do **envio único** e, principalmente, do BPMN em **arquivo (.xml)**, que apresentou médias superiores ao XML colado no chat. Não houve execuções com .zip. Assim, o melhor padrão para este modelo foi **envio único + arquivo .xml**.

No **Gemini Free** (Tabela 15), a variável mais determinante foi **mensagens separadas vs. envio único**: o envio único esteve associado a quedas severas de desempenho, enquanto mensagens separadas mantiveram médias altas. O **formato do BPMN** (texto, arquivo ou .zip) teve impacto secundário quando comparado ao efeito do modo de envio; portanto, a recomendação central é **evitar envio único** e manter o prompt em **mensagens separadas**.

Por fim, no **Claude Free** (Tabela 17), as melhores médias ocorreram com BPMN em **arquivo (.xml)** e com **mensagens separadas** (nos testes em que essa variação foi aplicada). Assim, o padrão recomendado é **arquivo .xml + mensagens separadas**, priorizando estabilidade e qualidade do resultado.

Em síntese, não houve uma estratégia universal: alguns modelos favorecem **envio único** (ChatGPT Plus e Gemini Pro), enquanto outros dependem fortemente de **mensagens separadas** (ChatGPT Free e Gemini Free). Quanto ao BPMN, **XML em texto** foi suficiente e mais eficiente no ChatGPT Plus/Free, enquanto **arquivo .xml** foi mais vantajoso no Gemini Pro e Claude Free. A Tabela 19 consolida as recomendações.

Tabela 19: Estratégias de entrada recomendadas por modelo de IA.

Modelo de IA	Formato recomendado
ChatGPT Plus	BPMN em texto
ChatGPT Free	Mensagens separadas + BPMN em texto; com ferramenta “pensando”
Gemini Pro	Envio único + arquivo (.xml)
Gemini Free	Mensagens separadas
Claude Free	Arquivo (.xml)

6 CONCLUSÃO

Este trabalho apresentou e avaliou uma metodologia para mapear modelos BPMN em artefatos de especificação de software, com foco na geração de diagramas UML apoiada por modelos de IA. A pesquisa partiu da consolidação de regras de mapeamento da literatura, seguiu com a sistematização e adaptação dessas regras para instruções executáveis em *prompts* e, por fim, validou a abordagem por meio de um estudo de caso com avaliação objetiva e reproduzível. Dessa forma, o objetivo geral de propor e testar uma metodologia capaz de transformar informação de processos (BPMN) em artefatos UML, com critérios claros de verificação, foi atingido.

Além disso, os achados obtidos permitem afirmar que as questões de pesquisa definidas na Seção 1.3 foram respondidas no escopo deste trabalho. A **RQ1** foi atendida pela elaboração e consolidação das regras mestras e pela sua adaptação em *prompts* executáveis, demonstrando que é viável definir regras de mapeamento úteis para orientar ferramentas de IA na transformação de BPMN em artefatos de especificação. A **RQ2** foi respondida pela comparação experimental entre diferentes ferramentas e modelos, evidenciando diferenças de comportamento e desempenho ao aplicar as mesmas regras, com resultados quantitativos consolidados pelas métricas. Por fim, a **RQ3** foi respondida pela avaliação objetiva baseada em checklist e pela comparação dos artefatos gerados pelas IAs com o resultado gerado manualmente, permitindo analisar similaridade e consistência por regra e por artefato.

Os resultados demonstraram que a metodologia permite comparar as IAs de forma objetiva e dizer exatamente em quais regras e artefatos cada uma mais falhou. Considerando a **média geral normalizada (0–10)** obtida no estudo (Tabela 18), o **ChatGPT Plus** alcançou **9.6**, seguido por **Claude Free** com **9.1** e **Gemini Pro** com **8.4**; já **ChatGPT Free** e **Gemini Free** obtiveram **7.7** e **7.2**, respectivamente. No geral, o **ChatGPT Plus** apresentou o melhor equilíbrio entre completude e estabilidade, configurando-se como a alternativa mais confiável para uso prático quando se deseja reduzir retrabalho de revisão. Ainda assim, o **Claude Free** não ficou muito atrás nos resultados globais, mantendo médias altas e consistentes, e se consolidou como uma

excelente alternativa gratuita. Na prática, considerando o custo de acesso, o Claude Free oferece uma relação benefício-custo muito favorável, especialmente pela consistência e pelo desempenho elevado na descrição textual. O **Gemini Pro** obteve desempenho expressivo no DCU, porém apresentou quedas relevantes em regras críticas associadas ao DC e ao DS, indicando maior sensibilidade a instruções que exigem integração entre elementos estruturais e comportamentais. Por fim, **ChatGPT Free** e **Gemini Free** apresentaram maior variabilidade e perdas mais acentuadas em regras que exigem coerência global, reduzindo a confiabilidade sem replicação, revisão e correções manuais.

De forma consistente entre as ferramentas, observou-se que os maiores gargalos do mapeamento não ocorreram nas regras mais diretas, mas nas que demandam rastreabilidade fina e coerência entre participantes, fluxos e elementos distribuídos nos artefatos, com destaque para **R12** e **R15**. Isso indica que, na prática, o uso de IA tende a ser mais robusto para tarefas de criação inicial de estruturas e elementos básicos, mas ainda requer validação humana cuidadosa em regras que dependem de consistência semântica global, manutenção de contexto e encadeamento correto do comportamento.

É fundamental ressaltar que a geração automatizada de artefatos de especificação por meio de IA não substitui o papel do engenheiro de software. Os modelos de linguagem atuam como ferramentas de apoio que aceleram a produção inicial de diagramas e descrições, mas a responsabilidade final pela qualidade, consistência e correção dos artefatos permanece com o profissional humano. A validação crítica por parte de engenheiros experientes é indispensável para identificar inconsistências semânticas, corrigir interpretações equivocadas das regras de mapeamento e garantir que os artefatos reflitam adequadamente os requisitos e o contexto do sistema. Dessa forma, a metodologia proposta deve ser compreendida como um recurso complementar ao trabalho de engenharia de requisitos, capaz de aumentar a produtividade e padronizar a geração de artefatos, porém sempre dependente de supervisão, revisão e refinamento humanos para assegurar a qualidade e a confiabilidade do resultado final.

Como contribuição central, o trabalho entrega um conjunto de regras mestras organizadas por artefato e uma forma de avaliação objetiva por checklist, que pode ser

reutilizada em novos estudos e em cenários reais de engenharia de requisitos. Assim, conclui-se que a proposta é viável como suporte ao processo de especificação: ela acelera a produção de artefatos, fornece transparência sobre onde a IA falha e reduz subjetividade na avaliação, desde que aplicada com controle de qualidade.

Como trabalhos futuros, recomenda-se: (i) testar a metodologia com **processos BPMN de maior complexidade**, incluindo múltiplos subprocessos, eventos intermediários variados e maior número de participantes, para verificar a escalabilidade das regras e do checklist; (ii) ampliar o conjunto de artefatos avaliados, incorporando, por exemplo, **diagramas de atividade**, bem como artefatos textuais complementares (regras de negócio e requisitos não funcionais); (iii) investigar **estratégias de rastreabilidade automática**, como mecanismos de validação cruzada entre artefatos, visando reduzir as falhas concentradas em R12 e R15; (iv) avaliar o desempenho em condições mais próximas de uso real, incluindo **restrições de tempo**, perfis distintos de usuários e revisão por especialistas, buscando medir ganho de produtividade e qualidade final do conjunto de especificação.

Por fim, conclui-se que a metodologia proposta cumpre seu papel ao oferecer um caminho sistemático e verificável para explorar BPMN como base de geração de artefatos UML com apoio de IA, evidenciando benefícios concretos: agilidade e padronização. Com refinamentos centrados nas regras críticas e com a integração de validações automáticas e revisão humana, a abordagem tem potencial para apoiar de forma efetiva atividades de engenharia de requisitos e especificação em diferentes domínios.

REFERÊNCIAS

- Association of Business Process Management Professionals (ABPMP), 2013. Guia para o Gerenciamento de Processos de Negócio – Corpo Comum de Conhecimento (BPM CBOK, primeira Edição. Vol. 3.0.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., 2001. **Manifesto for Agile Software Development**. Acesso em: 12 out. 2025. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>
- Bernhardt, G. H., Pfiz, F. S., Ho, S. L., Rodrigues, J. P. L., Zanini, E. d. O., 2023. **A utilização da linguagem de modelagem unificada (UML) na engenharia de software: uma avaliação da adoção e relevância**. Encontro Científico Cultural Interinstitucional 21.
- Bizagi, 2025a. **Bizagi Modeler – Overview (Help Center)**. https://help.bizagi.com/platform/en/intro_welcome.htm, acesso em: 27 dez. 2025.
- Bizagi, 2025b. **Process Modeling Software – Bizagi Modeler**. <https://www.bizagi.com/en/platform/modeler>, acesso em: 27 dez. 2025.
- Booch, G., Jacobson, I., Rumbaugh, J., 1996. The unified modeling language user guide. Addison Wesley.
- Bouzidi, A., Haddar, N., Abdallah, M. B., Haddar, K., **Deriving use case models from BPMN models**. In: 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017. IEEE, pp. 238–243.
- Chagas, M. C. R., Rocha, M. F., Rocha, G. d. O. S., Vasconcelos, A. P. V. d., Silva, S. V., Moraes, A. S. C., 2024. **Métodos de elicitação de requisitos a partir da modelagem de processos de negócio: revisão sistemática da literatura**. Observatório de la Economía Latinoamericana 22 (9), e6543. Disponível em: <https://ojs.observatoriolatinoamericano.com/ojs/index.php/olel/article/view/6543>
- Creswell, J. W., 2003. Research design: Qualitative, quantitative, and mixed methods approaches. Sage Publications, Thousand Oaks.
- Cruz, E. F., da Cruz, A. M. R., **Deriving Integrated Software Design Models from BPMN Business Process Models**. In: ICSoft, 2018. pp. 605–616.
- Cruz, E. F., Machado, R. J., Santos, M. Y., **From business process models to use case models: A systematic approach**. In: Enterprise engineering working conference, 2014. Springer, pp. 167–181.
- Davenport, T., 1993. Process Innovation: Reengineering Work Through Information Technology. Harvard Business Review Press. Disponível em: <https://books.google.com.br/books?id=kLLIOMGaKnsC>
- diagrams.net (draw.io), 2025a. **draw.io / diagrams.net: online diagram software**. <https://app.diagrams.net/>, acesso em: 27 dez. 2025.
- diagrams.net (draw.io), 2025b. **draw.io Documentation**. <https://www.drawio.com/doc/>, acesso em: 27 dez. 2025.

- Dumas, M., Rosa, L. M., Mendling, J., Reijers, A. H., 2018. Fundamentals of business process management. Springer.
- Fowler, M., 2014. UML Essencial: um breve guia para linguagem padrão. Bookman editora.
- Gil, A. C., 1991. Como elaborar projetos de pesquisa. Atlas, São Paulo.
- GitHub Docs, 2025. **Creating Mermaid diagrams**. <https://docs.github.com/en/get-started/writing-on-github/working-with-advanced-formatting/creating-diagrams>, acesso em: 27 dez. 2025.
- John, B., 2025. **A study of uml diagrams in software development**.
- Khlif, W., Ayed, N. E. B., Ben-Abdallah, H., **From a BPMN Model to an Aligned UML Analysis Model**. In: ICSoft, 2018. pp. 657–665.
- Lakatos, E. M., Marconi, M. d. A., 2017. Fundamentos de metodologia científica, 8ªEdição. Atlas, São Paulo.
- Mendling, J., Reijers, H. A., van der Aalst, W. M., 2010. **Seven process modeling guidelines (7PMG)**. Information and software technology 52 (2), 127–136. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584909001268>
- Mermaid contributors, 2025. **Mermaid: Diagramming and charting tool**. <https://mermaid.js.org/>, acesso em: 27 dez. 2025.
- Object Management Group (OMG), 2014. **Business Process Model and Notation (BPMN) Specification**. Versão 2.0.2. Acesso em: 11 ago. 2025. Disponível em: <https://www.omg.org/spec/BPMN/2.0.2/>
- PlantUML, 2025. **PlantUML: Generate diagrams from text**. <https://plantuml.com/>, acesso em: 27 dez. 2025.
- PlantUML contributors, 2025. **plantuml/plantuml: Generate diagrams from textual descriptions**. <https://github.com/plantuml/plantuml>, acesso em: 27 dez. 2025.
- Pressman, R. S., Maxim, B. R., 2021. Engenharia de Software: Uma Abordagem Profissional, 9ªEdição. AMGH, Porto Alegre.
- Rodríguez, A., de Guzmán, I. G.-R., Fernández-Medina, E., Piattini, M., 2010. **Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach**. Information and Software Technology 52 (9), 945–971.
- Santander, V. F., de Andrade, S. C., da Silva, I. F., Silva, A. L., Pozzan, V. A., Céspedes, M. A. T., **Obtendo Casos de Uso a Partir de Modelos de Processos de Negócio: Uma Revisão Sistemática da Literatura**. In: Workshop on Requirements Engineering (WER), 2022.
- Shaikh, A., Hafeez, A., Wagan, A. A., Alrizq, M., Alghamdi, A., Al Reshan, M. S., 2021. **More than two decades of research on verification of UML class models: A systematic literature review**. IEEE Access 9, 142461–142474.
- Souza, L., Brito, T., Lucena, M., Gomes, A., **Desenvolvimento Ágil e Documentação de Requisitos Rumo a Práticas Mais Efetivas: Uma Revisão Sistemática**. In: Workshop on Requirements Engineering (WER), 2024.

- SYDLE, 2025. **O que significa BPMS? Quais são seus componentes?** Acesso em: 03 nov. 2025. Disponível em: <https://www.sydle.com/br/blog/bpms-componentes-5f04d53c25565a666987ab83>
- Weske, M., 2019. Business Process Management: Concepts, Languages, Architectures, terceira Edição. Springer, Heidelberger Platz 3, 14197 Berlin, Germany.
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D. C., 2023. **A prompt pattern catalog to enhance prompt engineering with chatgpt.** arXiv preprint arXiv:2302.11382.
- Zhang, Q., Fang, C., Xie, Y., Zhang, Y., Yang, Y., Sun, W., Yu, S., Chen, Z., 2023. **A survey on large language models for software engineering.** arXiv preprint arXiv:2312.15223.
- Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., *et al.*, 2023. **A survey of large language models.** arXiv preprint arXiv:2303.18223 1 (2).

A ANEXO A – Descrição Textual dos Casos de Uso (Geração Manual)

- Enviar dados pessoais e preferências de horário

Atores: Paciente

Pré-condições: O paciente deseja agendar uma consulta na clínica.

Pós-condições: As informações cadastrais do paciente e suas preferências de dia e horário ficam registradas e disponíveis para a clínica realizar o agendamento.

Gatilhos: O paciente inicia uma solicitação de agendamento de consulta.

Cenário:

1. O paciente acessa o canal definido pela clínica para solicitar o agendamento.
 2. O paciente informa seus dados pessoais necessários para o cadastro.
 3. O paciente informa as preferências de data e horário para a consulta.
 4. O paciente confirma o envio dos dados à clínica, que são recebidos pelo atendente.
-

- Agendar consulta

Atores: Atendente

Pré-condições: Os dados pessoais do paciente e suas preferências de horário já estão registrados.

Pós-condições: A consulta do paciente fica agendada, com data e horário definidos e registrada nos dados da consulta.

Gatilhos: A clínica recebe os dados do paciente e suas preferências de horário.

Cenário:

1. O atendente recebe os dados do paciente e suas preferências de horário.
 2. O atendente verifica a disponibilidade de agenda do médico.
 3. O atendente seleciona data e horário adequados para a consulta.
 4. O atendente registra o agendamento da consulta no sistema.
-

- Confirmar consulta (WhatsApp)

Atores: Atendente; Paciente

Pré-condições: A consulta do paciente está agendada.

Pós-condições: A clínica passa a saber se o paciente confirmou ou não a realização da consulta.

Gatilhos: 24 horas antes da data e horário agendados para a consulta, ocorre o momento planejado para envio da confirmação.

Cenário:

1. O atendente identifica consultas agendadas para o dia seguinte.
2. O atendente seleciona o agendamento do paciente.
3. O atendente gera a mensagem de confirmação com data, horário e demais orientações necessárias.
4. O atendente envia a mensagem de confirmação ao paciente por WhatsApp.
5. O paciente lê a mensagem de confirmação enviada pela clínica.
6. O paciente decide se deseja manter ou não o agendamento da consulta.

7. O paciente envia uma resposta positiva ou negativa à clínica pelo WhatsApp.
8. A resposta do paciente é registrada pela clínica como confirmação ou não confirmação da consulta.

Observação Importante: Situações em que o paciente não responde à mensagem **não são consideradas** neste caso de uso e estão fora do escopo do processo.

- Cancelar consulta

Atores: Atendente

Pré-condições: A consulta estava previamente agendada. O paciente enviou uma resposta informando que não confirma a realização da consulta.

Pós-condições: A consulta é cancelada, e o processo referente a esse agendamento é encerrado.

Gatilhos: O atendente identifica uma resposta do paciente indicando que não realizará a consulta.

Cenário:

1. O atendente acessa o registro de agendamento do paciente.
 2. O atendente confirma que a resposta do paciente indica não realização da consulta.
 3. O atendente registra o cancelamento da consulta no sistema.
 4. O agendamento passa a constar como cancelado, sem ações posteriores para esse atendimento.
-

- Iniciar pré-atendimento

Atores: Atendente

Pré-condições: A consulta estava previamente agendada. A consulta foi confirmada pelo paciente.

Pós-condições: O paciente é colocado em pré-atendimento e está pronto para ter sua forma de pagamento ou cobertura de plano de saúde verificada.

Gatilhos: A data e o horário agendados para a consulta foram alcançados.

Cenário:

1. O atendente verifica a lista de consultas confirmadas para o horário atual.
 2. O atendente identifica o paciente que chegou para a consulta.
 3. O atendente registra o início do pré-atendimento do paciente.
 4. O atendente prepara o atendimento, verificando se será necessário validar plano de saúde ou pagamento particular.
-

- Obter dados do plano de saúde

Atores: Atendente; Paciente

Pré-condições: O pré-atendimento do paciente foi iniciado. Foi identificado que o paciente possui plano de saúde válido para tentativa de utilização.

Pós-condições: Os dados do plano de saúde do paciente ficam registrados e prontos para serem enviados à operadora para verificação de cobertura.

Gatilhos: Durante o pré-atendimento, constata-se que o paciente possui plano de saúde.

Cenário:

1. O atendente solicita ao paciente os dados necessários do plano (por exemplo, número da carteirinha, operadora, tipo de plano).
2. O paciente fornece os dados solicitados.
3. O atendente registra os dados do plano de saúde no sistema da clínica.

- Solicitar verificação de cobertura do plano

Atores: Atendente; Operadora do Plano de Saúde

Pré-condições: Os dados do plano de saúde do paciente já foram registrados pela clínica.

Pós-condições: A operadora recebe uma solicitação formal de verificação de cobertura para a consulta do paciente.

Gatilhos: A clínica decide verificar a cobertura do plano de saúde para a consulta do paciente.

Cenário:

1. O atendente seleciona o cadastro do paciente e seus dados de plano de saúde.
2. O atendente prepara uma solicitação de verificação de cobertura, incluindo os dados do paciente, do plano e da consulta.
3. A solicitação é enviada à operadora de plano de saúde.
4. A operadora recebe a solicitação e a registra para processamento interno.

- Verificar cobertura do plano

Atores: Operadora do Plano de Saúde

Pré-condições: A operadora recebeu uma solicitação de verificação de cobertura para o plano de saúde do paciente.

Pós-condições: A operadora gera um resultado de análise indicando se a consulta é ou não coberta pelo plano.

Gatilhos: Uma solicitação de verificação de cobertura é registrada internamente na operadora.

Cenário:

1. A operadora analisa os dados do plano de saúde do paciente.
2. A operadora verifica a elegibilidade e as regras de cobertura para o tipo de consulta solicitada.
3. A operadora define se a consulta será coberta ou não.
4. A operadora registra o resultado da análise de cobertura para envio à clínica.

- Retornar análise de cobertura

Atores: Operadora do Plano de Saúde; Atendente

Pré-condições: A análise de cobertura do plano de saúde foi concluída pela operadora.

Pós-condições: A clínica recebe o resultado da cobertura do plano do paciente.

Gatilhos: A operadora decide informar à clínica o resultado da análise de cobertura.

Cenário:

1. A operadora seleciona o resultado da análise de cobertura gerado para a solicitação da clínica.

2. A operadora envia o resultado de cobertura à clínica, indicando aprovação ou negativa.
3. O atendente da clínica recebe o resultado enviado pela operadora.
4. O atendente registra no sistema o resultado de cobertura do plano de saúde para a consulta do paciente.

- Processar pagamento particular

Atores: Atendente

Pré-condições: O pré-atendimento foi iniciado e uma das situações é válida: O paciente não possui plano de saúde; **ou** O plano de saúde do paciente não cobre a consulta solicitada.

Pós-condições: O pagamento particular da consulta está registrado.

Gatilhos: A clínica identifica que a consulta não será coberta por plano de saúde e que o paciente deverá pagar de forma particular.

Cenário:

1. O atendente informa ao paciente que a consulta será cobrada de forma particular.
2. O atendente inicia o fluxo de cobrança particular, incluindo os valores e formas de pagamento disponíveis.
3. O atendente segue os passos definidos para solicitar o pagamento particular ao paciente (caso de uso incluído **Solicitar pagamento particular ao paciente**).
4. Após o recebimento do comprovante, o atendente registra o pagamento no sistema (caso de uso incluído **Registrar pagamento no sistema**).
5. Com o pagamento registrado, o atendimento segue para liberação para realização pelo médico.

-Solicitar pagamento particular ao paciente

Atores: Atendente

Pré-condições: Foi decidido que a consulta será cobrada de forma particular. O processo de cobrança particular está em andamento (caso de uso **Processar pagamento particular**).

Pós-condições: O paciente é informado sobre o valor e as condições de pagamento, e a clínica fica aguardando o comprovante de pagamento.

Gatilhos: O atendente inicia o processamento de pagamento particular.

Cenário:

1. O atendente identifica o valor a ser cobrado pela consulta particular.
2. O atendente informa ao paciente o valor, o meio de pagamento aceito e o prazo para envio do comprovante.
3. O atendente orienta o paciente sobre como enviar o comprovante de pagamento à clínica.
4. A clínica passa a aguardar o recebimento do comprovante de pagamento.

Registrar pagamento particular no sistema **Atores:** Atendente

Pré-condições: O paciente já efetuou o pagamento particular. A clínica recebeu o comprovante de pagamento do paciente.

Pós-condições: O pagamento da consulta fica registrado como processado no sistema da clínica. A consulta passa a constar como com pagamento concluído.

Gatilhos: O atendente recebe o comprovante de pagamento da consulta particular.

Cenário:

1. O atendente recebe o comprovante de pagamento enviado pelo paciente.

2. O atendente confere os dados do comprovante, como valor e identificação do paciente.
 3. O atendente registra o pagamento da consulta no sistema.
 4. O sistema marca a consulta como paga.
-

- Liberar atendimento

Atores: Atendente

Pré-condições: O pré-atendimento foi iniciado e uma das situações é válida: O plano de saúde do paciente cobre a consulta solicitada **ou** O pagamento da consulta particular foi registrado no sistema.

Pós-condições: O atendimento é considerado liberado para realização pelo médico.

Gatilhos: A clínica identifica que a consulta será coberta por plano de saúde **ou** que o paciente realizou o pagamento particular.

Cenário:

1. O atendente libera o atendimento para realização da consulta pelo médico.
-

- Realizar atendimento

Atores: Médico

Pré-condições: O atendimento foi liberado, seja por aprovação da cobertura do plano de saúde, seja por pagamento particular registrado.

Pós-condições: A consulta médica é realizada e o atendimento ao paciente é concluído.

Gatilhos: O médico recebe a indicação de que o paciente está liberado para atendimento.

Cenário:

1. O médico verifica a lista de pacientes liberados para atendimento.
2. O médico chama o paciente para a consulta.
3. O médico realiza a consulta conforme o procedimento clínico adequado.
4. Ao final, o médico conclui o atendimento do paciente, encerrando o registro da consulta.

B ANEXO B – Descrição Textual dos Casos de Uso (ChatGPT Plus)

- UC01 — Enviar dados pessoais e preferências de horário

Atores: Paciente **Pré-condições:** Nenhuma. **Pós-condições:** Dados do paciente registrados para uso no agendamento. **Gatilhos:** Início da solicitação de agendamento. **Cenário:**

1. O Paciente informa seus dados pessoais.
 2. O Paciente informa suas preferências de horário.
 3. O sistema registra os dados do paciente.
-

- UC02 — Agendar consulta

Atores: Atendente **Pré-condições:** Dados do paciente disponíveis. **Pós-condições:** Consulta agendada com dados da consulta registrados. **Gatilhos:** Recebimento dos dados do paciente. **Cenário:**

1. O Atendente consulta os dados do paciente.
 2. O Atendente agenda a consulta conforme preferências informadas.
 3. O sistema registra os dados da consulta.
-

- UC03 — Confirmar consulta via WhatsApp

Atores: Atendente, Paciente **Pré-condições:** Consulta previamente agendada. **Pós-condições:** Confirmação registrada como aceita ou recusada. **Gatilhos:** Chegar o momento de 24 horas antes do horário da consulta. **Cenário:**

1. O Atendente envia ao Paciente uma mensagem solicitando confirmação da consulta.
 2. O Paciente responde à mensagem confirmando ou recusando a consulta.
 3. **Este caso de uso não cobre a situação em que o Paciente não responde à mensagem.**
-

- UC04 — Iniciar pré-atendimento

Atores: Atendente **Pré-condições:** `consultaConfirmada = true`. **Pós-condições:** Pré-atendimento iniciado. **Gatilhos:** Chegar o horário da consulta. **Cenário:**

1. O Atendente inicia o pré-atendimento do Paciente.
-

- UC05 — Cancelar consulta

Atores: Atendente **Pré-condições:** `consultaConfirmada = false`. **Pós-condições:** Consulta cancelada. **Gatilhos:** Recusa de confirmação pelo Paciente. **Cenário:**

1. O Atendente cancela a consulta.
 2. O sistema registra que a consulta foi cancelada.
-

- UC06 — Informar dados do plano de saúde

Atores: Atendente, Paciente **Pré-condições:** `pacientePossuiPlanoDeSaude = true`. **Pós-condições:** Dados do plano de saúde registrados e disponíveis para verificação. **Gatilhos:** Início do pré-atendimento com necessidade de validar plano de saúde. **Cenário:**

1. O Atendente solicita ao Paciente os dados do plano de saúde.
 2. O Paciente fornece os dados do plano de saúde.
 3. O sistema registra os dados do plano de saúde.
-

- UC07 — Transmitir solicitação de verificação do plano

Atores: Atendente, Operadora do Plano de Saúde **Pré-condições:** Dados do plano de saúde disponíveis. **Pós-condições:** Solicitação de verificação entregue à Operadora do Plano de Saúde. **Gatilhos:** Disponibilização dos dados do plano de saúde para verificação. **Cenário:**

1. O Atendente encaminha à Operadora do Plano de Saúde a solicitação de verificação, contendo os dados do plano.
 2. A Operadora do Plano de Saúde recebe a solicitação.
-

- UC08 — Verificar cobertura do plano

Atores: Operadora do Plano de Saúde **Pré-condições:** Solicitação de verificação recebida. **Pós-condições:** Resultado da análise produzido. **Gatilhos:** Recebimento da solicitação de verificação. **Cenário:**

1. A Operadora do Plano de Saúde analisa a cobertura do plano para a solicitação recebida.
 2. O sistema registra o resultado da análise.
-

- UC09 — Transmitir análise de cobertura

Atores: Operadora do Plano de Saúde, Atendente **Pré-condições:** Resultado da análise disponível. **Pós-condições:** Resultado da análise recebido pelo Atendente. **Gatilhos:** Conclusão da análise de cobertura. **Cenário:**

1. A Operadora do Plano de Saúde envia ao Atendente o resultado da análise de cobertura.
 2. O Atendente recebe o resultado da análise.
-

- UC10 — Processar pagamento particular

Atores: Atendente **Pré-condições:** `pacientePossuiPlanoDeSaude = false` ou `coberturaAprovada = false`. **Pós-condições:** Pagamento processado. **Gatilhos:** Necessidade de pagamento particular para prosseguir. **Cenário:**

1. O Atendente solicita o pagamento particular ao Paciente. (inclui UC11)
 2. O Atendente aguarda o recebimento do comprovante de pagamento.
 3. O Atendente registra o pagamento no sistema usando o comprovante. (inclui UC12)
 4. O sistema registra que o pagamento foi processado.
-

- UC11 — Solicitar pagamento particular ao paciente

Atores: Atendente **Pré-condições:** Necessidade de pagamento particular identificada. **Pós-condições:** Solicitação de pagamento enviada ao Paciente. **Gatilhos:** Início do processamento do pagamento particular. **Cenário:**

1. O Atendente solicita ao Paciente o pagamento particular.
-

- UC12 — Registrar pagamento no sistema

Atores: Atendente **Pré-condições:** Comprovante de pagamento disponível. **Pós-condições:** Pagamento registrado no sistema e processado. **Gatilhos:** Recebimento do comprovante de pagamento. **Cenário:**

1. O Atendente registra o pagamento no sistema usando o comprovante de pagamento.
 2. O sistema registra que o pagamento foi processado.
-

- UC13 — Liberar Atendimento

Atores: Atendente **Pré-condições:** `coberturaAprovada = true` ou pagamento particular processado. **Pós-condições:** Atendimento liberado. **Gatilhos:** Recebimento do resultado da análise de cobertura ou conclusão do pagamento particular. **Cenário:**

1. O Atendente libera o atendimento para o Médico.
-

- UC14 — Realizar atendimento

Atores: Médico **Pré-condições:** Atendimento liberado. **Pós-condições:** Consulta realizada. **Gatilhos:** Liberação do atendimento. **Cenário:**

1. O Médico realiza o atendimento.
2. O sistema registra que a consulta foi realizada.

C ANEXO C – Descrição Textual dos Casos de Uso (ChatGPT Free)

- 1. Enviar dados pessoais e preferências de horário

Atores: Paciente **Pré-condições:** O paciente tem acesso ao canal de solicitação (formulário, aplicativo ou contato). **Pós-condições:** Os dados pessoais e as preferências de horário ficam registrados no sistema da clínica. **Gatilhos:** O paciente inicia a intenção de agendar uma consulta. **Cenário (fluxo principal):**

1. O paciente fornece nome, documento de identificação, telefone, e preferências de data/hora.
 2. O sistema/atendente recebe os dados e grava o registro de solicitação do paciente.
 3. O caso de uso termina com a informação disponível para a etapa de agendamento.
-

- 2. Agendar consulta

Atores: Atendente **Pré-condições:** Existem dados pessoais do paciente disponíveis (fornecidos no caso de uso "Enviar dados pessoais e preferências de horário"). **Pós-condições:** É gerado um registro da consulta (data/hora/status) associado ao paciente. **Gatilhos:** Chegada de nova solicitação com dados do paciente a ser agendado. **Cenário (fluxo principal):**

1. O atendente consulta as preferências e disponibilidade do calendário.
 2. O atendente escolhe data/hora adequadas e registra a consulta no sistema.
 3. O sistema gera o registro da consulta para uso nas próximas etapas.
-

- 3. Confirmar consulta (via mensagem)

Atores: Atendente, Paciente **Pré-condições:** Existe uma consulta agendada; há um contato (telefone/WhatsApp) cadastrado para o paciente. **Pós-condições:** A confirmação do paciente é registrada; dependendo da resposta, segue o fluxo de atendimento ou de cancelamento/pagamento. **Gatilhos:** Aproxima-se a data/hora agendada ou o processo solicita confirmação. **Cenário (fluxo principal):**

1. O atendente envia uma mensagem de confirmação ao paciente contendo data e horário.
 2. O paciente responde à mensagem confirmando ou não a presença.
 3. Se o paciente confirma, o sistema registra a confirmação e o processo segue para a etapa de pré-atendimento.
 4. Se o paciente não confirma, registra-se o cancelamento do agendamento. **Observação obrigatória:** O processo descrito **não** considera o cenário em que o paciente não responde à mensagem.
-

- 4. Cancelar consulta

Atores: Atendente **Pré-condições:** O paciente informou não confirmar a presença ou houve solicitação explícita de cancelamento. **Pós-condições:** O registro da consulta é atualizado para cancelado; fluxos posteriores são encerrados. **Gatilhos:** Resposta negativa do paciente à confirmação ou solicitação explícita de cancelamento. **Cenário (fluxo principal):**

1. O atendente seleciona a consulta a ser cancelada.
2. O atendente executa o cancelamento no sistema.
3. O sistema marca a consulta como cancelada e encerra o fluxo relacionado.

- 5. Iniciar pré-atendimento

Atores: Atendente **Pré-condições:** A consulta foi confirmada e chegou a hora de iniciar o atendimento preliminar. **Pós-condições:** O paciente é encaminhado para a verificação administrativa e clínica inicial; são identificadas necessidades adicionais (ex.: dados do plano). **Gatilhos:** Chegada do horário de atendimento ou registro de confirmação válida. **Cenário (fluxo principal):**

1. O atendente inicia os procedimentos de pré-atendimento: checagem de cadastro e documentação.
2. Se for identificado que o paciente possui plano de saúde, o atendente solicita os dados do plano e dá seguimento para verificação externa.
3. Se constatar que não há plano ou a cobertura não será aprovada, o atendente aciona o caminho para pagamento particular (ou outro procedimento apropriado).

- 6. Solicitar/Fornecer dados do plano de saúde

Atores: Atendente, Paciente **Pré-condições:** Identificação prévia da existência de um plano (ou necessidade de confirmar). **Pós-condições:** Os dados do plano passam a constar no processo e ficam disponíveis para verificação externa. **Gatilhos:** Identificação em pré-atendimento da necessidade de verificar cobertura. **Cenário (fluxo principal):**

1. O atendente solicita ao paciente informações da operadora e número do cartão do plano.
2. O paciente fornece os dados solicitados.
3. O atendente registra os dados no sistema para envio à operadora.

- 7. Solicitar verificação do plano de saúde

Atores: Atendente, Operadora do Plano de Saúde **Pré-condições:** Dados do plano foram recebidos e registrados pelo atendente. **Pós-condições:** A solicitação de verificação é enviada à operadora e um resultado será devolvido. **Gatilhos:** Registro dos dados do plano no sistema após pré-atendimento. **Cenário (fluxo principal):**

1. O atendente encaminha à operadora os dados do plano do paciente solicitando análise de cobertura para o atendimento previsto.
2. A operadora recebe a solicitação e inicia sua análise interna.

- 8. Verificar cobertura do plano

Atores: Operadora do Plano de Saúde **Pré-condições:** Recebimento dos dados do plano e solicitação de verificação. **Pós-condições:** A operadora produz um resultado de análise (aprovado/negado e observações). **Gatilhos:** Recebimento de solicitação de verificação por parte da clínica. **Cenário (fluxo principal):**

1. A operadora consulta suas regras e registros internos com base nos dados recebidos.
2. A operadora decide sobre a cobertura (aprovada ou negada) e registra o resultado.

- 9. Fornecer/Receber análise de cobertura

Atores: Operadora do Plano de Saúde, Atendente **Pré-condições:** A operadora concluiu a verificação da cobertura. **Pós-condições:** A clínica possui o resultado da análise e pode decidir liberar o atendimento ou acionar pagamento particular. **Gatilhos:** Conclusão da verificação por parte da operadora. **Cenário (fluxo principal):**

1. A operadora envia o resultado da análise para a clínica.
2. O atendente recebe o resultado e registra a decisão.
3. Se a cobertura for aprovada, segue-se para liberação do atendimento; se negada, inicia-se o fluxo para pagamento particular.

- 10. Liberar Atendimento

Atores: Atendente **Pré-condições:** A análise de cobertura foi recebida e indica cobertura aprovada, ou o pagamento particular foi processado com sucesso. **Pós-condições:** O paciente está autorizado a ser atendido pelo médico. **Gatilhos:** Resultado de análise aprovado ou pagamento particular confirmado. **Cenário (fluxo principal):**

1. O atendente registra a liberação do atendimento no sistema.
2. O paciente é encaminhado para o médico para realização do atendimento.

- 11. Processar pagamento particular (subprocesso)

Atores: Atendente (executante do subprocesso) **Pré-condições:** A cobertura não foi aprovada ou o paciente declarou que irá pagar de forma particular. **Pós-condições:** O pagamento particular é confirmado e registrado no sistema; comprovante fica associado ao atendimento. **Gatilhos:** Decisão de proceder com pagamento particular (após negação de cobertura ou ausência de plano). **Cenário (fluxo principal):**

1. O atendente solicita ao paciente o pagamento particular.
2. O atendente aguarda o comprovante de pagamento.
3. Ao receber o comprovante, o atendente registra o pagamento no sistema.
4. O subprocesso é concluído com o registro do pagamento processado.

- 12. Solicitar pagamento particular ao paciente

Atores: Atendente **Pré-condições:** O caminho de pagamento particular foi acionado. **Pós-condições:** O paciente foi informado do valor e solicitado a enviar comprovante. **Gatilhos:** A clínica determina que será necessário pagamento particular. **Cenário (fluxo principal):**

1. O atendente comunica o valor e forma de pagamento ao paciente.
2. O paciente realiza o pagamento e, quando disponível, encaminha o comprovante.

-
- 13. Registrar pagamento no sistema

Atores: Atendente **Pré-condições:** O comprovante de pagamento foi recebido. **Pós-condições:** O sistema registra o comprovante e o pagamento como confirmado. **Gatilhos:** Recebimento do comprovante de pagamento. **Cenário (fluxo principal):**

1. O atendente verifica o comprovante.
 2. O atendente registra os detalhes do pagamento (valor, data, método) no sistema.
-

- 14. Realizar atendimento

Atores: Médico **Pré-condições:** Atendimento liberado pelo atendente (cobertura aprovada ou pagamento confirmado). **Pós-condições:** A consulta é realizada; resultado clínico e encerramento do atendimento são registrados. **Gatilhos:** Paciente apresenta-se ao médico após liberação. **Cenário (fluxo principal):**

1. O médico realiza a consulta e registra o resultado clínico.
2. O médico finaliza o atendimento e encerra o registro associado à consulta.

D ANEXO D – Descrição Textual dos Casos de Uso (Gemini Pro)

- UC1: Enviar dados pessoais e preferências de horário

Atores: Paciente **Pré-condições:** O Paciente iniciou a solicitação de agendamento. **Pós-condições:** Os dados foram enviados para o agendamento. **Gatilhos:** Início do processo pelo Paciente. **Cenário:**

1. O Paciente fornece seus dados pessoais e preferências de horário.
2. O sistema registra os **Dados do Paciente**.

- UC2: Agendar consulta

Atores: Atendente **Pré-condições:** Os dados pessoais e preferências de horário foram recebidos. **Pós-condições:** O sistema aguarda o período de 24 horas antes da consulta. **Gatilhos:** Recebimento dos dados do Paciente. **Cenário:**

1. O Atendente analisa os **Dados do Paciente**.
2. O Atendente realiza o agendamento, gerando os **Dados da Consulta**.
3. O processo entra em espera de 24 horas antes do horário agendado.

- UC3: Confirmar agendamento via mensagem

Atores: Atendente, Paciente **Pré-condições:** O período de 24 horas antes da consulta foi atingido. **Pós-condições:** A confirmação da consulta é verificada (Confirmada ou Não Confirmada). **Gatilhos:** Fim do período de espera de 24 horas. **Cenário:**

1. O Atendente utiliza os **Dados da Consulta** para enviar uma mensagem de confirmação (WhatsApp).
2. O Paciente responde à mensagem de confirmação.
3. Nota: Não é considerado o caso em que o paciente não responde à mensagem.

- UC4: Cancelar consulta

Atores: Atendente **Pré-condições:** A consulta não foi confirmada pelo Paciente. **Pós-condições:** A consulta é marcada como cancelada e o processo encerra. **Gatilhos:** Verificação negativa da confirmação. **Cenário:**

1. O Atendente cancela o agendamento no sistema.

- UC5: Iniciar pré-atendimento

Atores: Atendente **Pré-condições:** A consulta foi confirmada pelo Paciente e o horário da consulta chegou. **Pós-condições:** Verifica-se se o paciente possui plano de saúde para direcionar o fluxo (Plano ou Particular). **Gatilhos:** Chegada da hora da consulta. **Cenário:**

1. O Atendente inicia os procedimentos de pré-atendimento.

- UC6: Obter dados do plano de saúde

Atores: Atendente, Paciente **Pré-condições:** O paciente possui plano de saúde. **Pós-condições:** Os dados do plano foram coletados para verificação. **Gatilhos:** Confirmação de que o paciente possui plano. **Cenário:**

1. O Atendente solicita os **Dados do Plano de Saúde**.
2. O Paciente fornece as informações solicitadas.

- UC8: Encaminhar solicitação de verificação

Atores: Atendente, Operadora do Plano de Saúde **Pré-condições:** Os dados do plano de saúde foram coletados. **Pós-condições:** A solicitação foi enviada para a Operadora. **Gatilhos:** Conclusão da coleta de dados do plano. **Cenário:**

1. O Atendente encaminha os **Dados do Plano de Saúde** para a Operadora.
2. A Operadora recebe a solicitação de verificação.

- UC10: Verificar cobertura do plano

Atores: Operadora do Plano de Saúde **Pré-condições:** A solicitação de verificação foi recebida. **Pós-condições:** O resultado da análise de cobertura foi gerado. **Gatilhos:** Recebimento dos dados do plano. **Cenário:**

1. A Operadora verifica a validade e cobertura utilizando os **Dados do Plano de Saúde**.
2. A Operadora gera o **Resultado da Análise**.

- UC9: Receber análise de cobertura da operadora

Atores: Operadora do Plano de Saúde, Atendente **Pré-condições:** O resultado da análise foi gerado pela Operadora. **Pós-condições:** Verifica-se o status da aprovação (Aprovada ou Reprovada). **Gatilhos:** Conclusão da verificação de cobertura. **Cenário:**

1. A Operadora fornece o **Resultado da Análise**.
2. O Atendente recebe a análise de cobertura.

- UC11: Liberar Atendimento

Atores: Atendente **Pré-condições:** A cobertura do plano foi aprovada OU o pagamento particular foi processado. **Pós-condições:** O paciente está liberado para o atendimento médico. **Gatilhos:** Aprovação da cobertura ou conclusão do pagamento particular. **Cenário:**

1. O Atendente libera o paciente para a consulta.

- UC12: Realizar atendimento

Atores: Médico **Pré-condições:** O atendimento foi liberado. **Pós-condições:** Consulta realizada e processo concluído. **Gatilhos:** Liberação do atendimento. **Cenário:**

1. O Médico realiza o atendimento clínico ao paciente.

- UC7: Processar pagamento particular

Atores: Atendente **Pré-condições:** O paciente não possui plano de saúde OU a cobertura do plano foi reprovada. **Pós-condições:** O pagamento é processado e o atendimento pode ser liberado. **Gatilhos:** Condição negativa de plano de saúde ou reprovação de cobertura. **Cenário:**

1. O Atendente executa o processo de pagamento particular (inclui solicitar e registrar pagamento).

- UC_Sub1: Solicitar pagamento particular ao paciente

Atores: Atendente **Pré-condições:** Início do processamento de pagamento particular. **Pós-condições:** Aguarda-se o comprovante de pagamento. **Gatilhos:** Início do subprocesso. **Cenário:**

1. O Atendente solicita o pagamento ao paciente.

- UC_Sub2: Registrar pagamento no sistema

Atores: Atendente **Pré-condições:** O comprovante de pagamento foi recebido. **Pós-condições:** O pagamento consta como processado. **Gatilhos:** Recebimento do **Comprovante de Pagamento**. **Cenário:**

1. O Atendente registra a transação no sistema utilizando o **Comprovante de Pagamento**.

E ANEXO E – Descrição Textual dos Casos de Uso (Gemini Free)

- UC01: Enviar dados pessoais e preferências de horário

Atores: Paciente **Pré-condições:** O Paciente decide iniciar a solicitação de agendamento. **Pós-condições:** Os dados são enviados para análise do atendente. **Gatilhos:** Início do processo pelo Paciente. **Cenário:** O Paciente insere e envia as informações necessárias, criando o registro de "Dados do paciente".

- UC02: Agendar consulta

Atores: Atendente **Pré-condições:** Os dados pessoais e preferências foram recebidos. **Pós-condições:** A consulta fica agendada e o sistema entra em estado de espera. **Gatilhos:** Recebimento dos dados do paciente. **Cenário:** O Atendente analisa os dados recebidos, consulta a agenda e efetiva o agendamento, gerando os "Dados da consulta".

- UC03: Enviar mensagem de confirmação (WhatsApp)

Atores: Atendente **Pré-condições:** A consulta está agendada e o período de antecedência foi atingido. **Pós-condições:** A mensagem de confirmação é enviada ao paciente. **Gatilhos:** Passagem de tempo de 24 horas antes da consulta agendada. **Cenário:** O Atendente utiliza o sistema para enviar uma notificação via WhatsApp solicitando confirmação da presença.

- UC04: Responder mensagem de confirmação

Atores: Paciente **Pré-condições:** O Paciente recebeu a mensagem de confirmação. **Pós-condições:** A resposta é registrada para tomada de decisão. **Gatilhos:** Recebimento da mensagem de confirmação. **Cenário:** O Paciente responde à mensagem informando se comparecerá ou não.

- UC05: Cancelar consulta

Atores: Atendente **Pré-condições:** O Paciente informou que não comparecerá (resposta negativa). **Pós-condições:** A consulta é cancelada e o processo encerrado. **Gatilhos:** Confirmação negativa por parte do paciente. **Cenário:** O Atendente executa o cancelamento do registro da consulta no sistema.

- UC06: Iniciar pré-atendimento

Atores: Atendente **Pré-condições:** O Paciente confirmou presença e o horário da consulta chegou. **Pós-condições:** O status do paciente é atualizado para pré-atendimento. **Gatilhos:** Chegada da hora da consulta após confirmação positiva. **Cenário:** O Atendente inicia os procedimentos preparatórios para a consulta.

- UC07: Solicitar dados do plano de saúde

Atores: Atendente **Pré-condições:** Foi identificado que o paciente possui plano de saúde. **Pós-condições:** O sistema aguarda o fornecimento dos dados do plano. **Gatilhos:** Identificação de cobertura por plano de saúde no pré-atendimento. **Cenário:** O Atendente solicita ao paciente as informações da carteirinha ou convênio.

- UC08: Fornecer dados do plano de saúde

Atores: Paciente **Pré-condições:** O Atendente solicitou os dados do plano. **Pós-condições:** Os dados do plano são disponibilizados para o atendente. **Gatilhos:** Solicitação do atendente. **Cenário:** O Paciente entrega ou informa os "Dados do plano de saúde".

- UC09: Encaminhar solicitação de verificação à operadora

Atores: Atendente **Pré-condições:** Os dados do plano de saúde foram fornecidos. **Pós-condições:** A solicitação é enviada para a operadora externa. **Gatilhos:** Recebimento dos dados do plano de saúde. **Cenário:** O Atendente submete os "Dados do plano de saúde" para validação externa.

- UC10: Receber solicitação de verificação

Atores: Operadora do Plano de Saúde **Pré-condições:** Uma solicitação foi enviada pela clínica. **Pós-condições:** A solicitação está pronta para análise. **Gatilhos:** Chegada da mensagem com dados do plano. **Cenário:** A Operadora recebe os "Dados do plano de saúde" para processamento.

- UC11: Verificar cobertura do plano

Atores: Operadora do Plano de Saúde **Pré-condições:** A solicitação foi recebida. **Pós-condições:** O resultado da análise é gerado. **Gatilhos:** Conclusão do recebimento da solicitação. **Cenário:** A Operadora analisa a validade e a cobertura do plano, gerando o "Resultado da análise".

- UC12: Fornecer análise de cobertura

Atores: Operadora do Plano de Saúde **Pré-condições:** A verificação foi concluída. **Pós-condições:** O resultado é enviado de volta à clínica. **Gatilhos:** Conclusão da verificação de cobertura. **Cenário:** A Operadora transmite o "Resultado da análise" para o solicitante.

- UC13: Receber análise de cobertura

Atores: Atendente **Pré-condições:** A operadora enviou o resultado. **Pós-condições:** O resultado está disponível para decisão de liberação. **Gatilhos:** Chegada da mensagem com a análise. **Cenário:** O Atendente recebe e visualiza o "Resultado da análise".

- UC14: Liberar Atendimento

Atores: Atendente **Pré-condições:** A cobertura do plano foi aprovada OU o pagamento particular foi processado com sucesso. **Pós-condições:** O paciente está apto a ser atendido pelo médico. **Gatilhos:** Aprovação da cobertura ou conclusão do pagamento. **Cenário:** O Atendente autoriza o encaminhamento do paciente ao consultório médico.

- UC15: Processar pagamento particular

Atores: Atendente **Pré-condições:** O paciente não possui plano de saúde OU a cobertura do plano foi negada. **Pós-condições:** O pagamento é regularizado para permitir o atendimento. **Gatilhos:** Falta de plano de saúde ou negativa de cobertura. **Cenário:** Este caso de uso coordena o subfluxo de cobrança direta ao paciente.

- UC16: Solicitar pagamento particular ao paciente

Atores: Atendente **Pré-condições:** O processo de pagamento particular foi iniciado. **Pós-condições:** O sistema aguarda o comprovante de pagamento. **Gatilhos:** Início do subfluxo de pagamento. **Cenário:** O Atendente informa o valor e solicita a quitação ao paciente.

- UC17: Registrar pagamento no sistema

Atores: Atendente **Pré-condições:** O comprovante de pagamento foi recebido. **Pós-condições:** O pagamento é confirmado no sistema. **Gatilhos:** Recebimento do comprovante. **Cenário:** O Atendente insere os dados do "Comprovante de pagamento" para dar baixa na pendência financeira.

- UC18: Realizar atendimento

Atores: Médico **Pré-condições:** O atendimento foi liberado pelo atendente. **Pós-condições:** O serviço médico é concluído. **Gatilhos:** Liberação do paciente para o consultório. **Cenário:** O Médico realiza a consulta ou procedimento necessário e finaliza o atendimento.

F ANEXO F – Descrição Textual dos Casos de Uso (Claude Free)

- UC1: Enviar dados pessoais e preferências de horário

Atores: Paciente

Pré-condições: Solicitação de agendamento iniciada.

Pós-condições: Dados pessoais e preferências de horário enviados para processamento.

Gatilhos: Início da solicitação de agendamento.

Cenário:

1. O paciente acessa o sistema e inicia a solicitação de agendamento.
 2. O paciente fornece seus dados pessoais (nome, CPF, telefone, email).
 3. O paciente informa suas preferências de horário para a consulta.
 4. Os dados são enviados ao atendente para processamento.
-

- UC2: Agendar consulta

Atores: Atendente

Pré-condições: Dados do paciente recebidos.

Pós-condições: Consulta agendada e registrada no sistema.

Gatilhos: Recebimento dos dados pessoais e preferências do paciente.

Cenário:

1. O atendente recebe os dados pessoais e preferências de horário do paciente.
 2. O atendente consulta a disponibilidade na agenda.
 3. O atendente registra a consulta no sistema com data e horário definidos.
 4. Os dados da consulta são armazenados.
 5. O sistema aguarda 24 horas antes da consulta para enviar confirmação.
-

- UC3: Confirmar consulta via WhatsApp

Atores: Atendente, Paciente

Pré-condições: Consulta agendada e período de 24 horas antes da consulta atingido.

Pós-condições: Confirmação ou recusa da consulta registrada.

Gatilhos: Decorrido o tempo de 24 horas antes do horário agendado.

Cenário:

1. O sistema dispara o evento de confirmação 24 horas antes da consulta.
2. O atendente envia uma mensagem de confirmação via WhatsApp ao paciente.
3. O paciente recebe a mensagem e responde confirmando ou não sua presença.
4. A resposta do paciente é registrada no sistema.

Observações: Não é considerado o caso em que o paciente não responde à mensagem.

- UC4: Cancelar consulta

Atores: Atendente

Pré-condições: Paciente não confirmou a consulta.

Pós-condições: Consulta cancelada no sistema.

Gatilhos: Resposta negativa do paciente à mensagem de confirmação.

Cenário:

1. O atendente identifica que o paciente não confirmou sua presença.
 2. O atendente cancela a consulta no sistema.
 3. O horário é liberado na agenda.
 4. O processo de agendamento é encerrado.
-

- UC5: Iniciar pré-atendimento

Atores: Atendente

Pré-condições: Consulta confirmada pelo paciente e horário da consulta atingido.

Pós-condições: Processo de pré-atendimento iniciado.

Gatilhos: Chegada do horário agendado para a consulta.

Cenário:

1. O sistema dispara o evento de início de pré-atendimento no horário da consulta.
 2. O atendente inicia o processo de pré-atendimento.
 3. O atendente verifica se o paciente possui plano de saúde ou se será atendimento particular.
-

- UC6: Solicitar e fornecer dados do plano de saúde

Atores: Atendente, Paciente

Pré-condições: Paciente declarou possuir plano de saúde.

Pós-condições: Dados do plano de saúde obtidos e registrados.

Gatilhos: Verificação positiva de que o paciente possui plano de saúde.

Cenário:

1. O atendente solicita ao paciente os dados do seu plano de saúde.
 2. O paciente fornece o número da carteira, nome da operadora e validade.
 3. Os dados do plano de saúde são registrados para verificação de cobertura.
-

- UC7: Solicitar verificação de cobertura

Atores: Atendente, Operadora do Plano de Saúde

Pré-condições: Dados do plano de saúde fornecidos pelo paciente.

Pós-condições: Solicitação de verificação enviada e recebida pela operadora.

Gatilhos: Recebimento dos dados do plano de saúde.

Cenário:

1. O atendente encaminha a solicitação de verificação de cobertura à operadora do plano de saúde.
2. A operadora recebe a solicitação com os dados do plano.
3. A operadora inicia o processo de análise de cobertura.

-
- UC8: Verificar cobertura do plano

Atores: Operadora do Plano de Saúde

Pré-condições: Solicitação de verificação recebida.

Pós-condições: Análise de cobertura realizada e resultado gerado.

Gatilhos: Recebimento da solicitação de verificação.

Cenário:

1. A operadora analisa os dados do plano de saúde do paciente.
 2. A operadora verifica se a consulta está coberta pelo plano.
 3. A operadora gera o resultado da análise indicando aprovação ou não da cobertura.
-

- UC9: Fornecer e receber análise de cobertura

Atores: Operadora do Plano de Saúde, Atendente

Pré-condições: Análise de cobertura concluída.

Pós-condições: Resultado da análise recebido pelo atendente.

Gatilhos: Conclusão da verificação de cobertura pela operadora.

Cenário:

1. A operadora fornece o resultado da análise de cobertura ao sistema da clínica.
 2. O atendente recebe o resultado indicando se a cobertura foi aprovada ou não.
 3. O atendente prossegue com o fluxo adequado baseado no resultado.
-

- UC10: Processar pagamento particular

Atores: Atendente

Pré-condições: Paciente não possui plano de saúde ou cobertura não foi aprovada.

Pós-condições: Pagamento particular processado e registrado.

Gatilhos: Identificação de necessidade de pagamento particular ou rejeição de cobertura.

Cenário:

1. O atendente identifica a necessidade de processar pagamento particular.
 2. O atendente executa as atividades de solicitação e registro do pagamento.
 3. O pagamento é confirmado e registrado no sistema.
-

- UC11: Solicitar pagamento particular ao paciente

Atores: Atendente

Pré-condições: Processo de pagamento particular iniciado.

Pós-condições: Solicitação de pagamento enviada ao paciente.

Gatilhos: Início do subprocesso de processamento de pagamento particular.

Cenário:

1. O atendente solicita o pagamento particular ao paciente.

2. O atendente informa o valor da consulta.
3. O atendente aguarda o comprovante de pagamento do paciente.

- UC12: Registrar pagamento no sistema

Atores: Atendente

Pré-condições: Comprovante de pagamento recebido.

Pós-condições: Pagamento registrado no sistema.

Gatilhos: Recebimento do comprovante de pagamento.

Cenário:

1. O atendente recebe o comprovante de pagamento do paciente.
2. O atendente registra o pagamento no sistema com número do comprovante, valor e data.
3. O sistema confirma o registro do pagamento.

- UC13: Liberar Atendimento

Atores: Atendente

Pré-condições: Cobertura do plano aprovada ou pagamento particular processado.

Pós-condições: Atendimento liberado para realização pelo médico.

Gatilhos: Confirmação de cobertura aprovada ou conclusão do pagamento particular.

Cenário:

1. O atendente verifica a aprovação da cobertura ou confirmação do pagamento.
2. O atendente libera o atendimento no sistema.
3. O médico é notificado para realizar o atendimento.

- UC14: Realizar atendimento

Atores: Médico

Pré-condições: Atendimento liberado pelo atendente.

Pós-condições: Consulta médica realizada.

Gatilhos: Liberação do atendimento.

Cenário:

1. O médico recebe a notificação de atendimento liberado.
2. O médico realiza a consulta com o paciente.
3. O processo de agendamento é concluído.