

**INSTITUTO FEDERAL GOIANO – CAMPUS CERES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
DOUGLAS CÁSSIO REIS SILVA OLIVEIRA**

**CHATBOTS INTELIGENTES: Uma Análise Comparativa de Tecnologias e seus
Impactos na Experiência do Usuário em Plataformas Digitais**

**CERES – GO
2025**

DOUGLAS CÁSSIO REIS SILVA OLIVEIRA

**CHATBOTS INTELIGENTES: Uma Análise Comparativa de Tecnologias e seus
Impactos na Experiência do Usuário em Plataformas Digitais**

Trabalho de curso apresentado ao curso de Sistemas de Informação do Instituto Federal Goiano – Campus Ceres, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação sob orientação do Prof. Dr. Vilson de Soares Siqueira.

**Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema Integrado de Bibliotecas do IF Goiano - SIBI**

O48c Olvieira, Douglas Cássio Reis Silva
 CHATBOTS INTELIGENTES: Uma Análise Comparativa de
 Tecnologias e seus Impactos na Experiência do Usuário em
 Plataformas Digitais / Douglas Cássio Reis Silva Olvieira. Ceres
 2025.

99f. il.

Orientador: Prof. Dr. Vilson Soares de Siqueira.
Tcc (Bacharel) - Instituto Federal Goiano, curso de 0320203 -
Bacharelado em Sistemas de Informação - Ceres (Campus
Ceres).
1. Experiência do usuário. 2. Sistemas. 3. Comparaçao de
tecnologias. 4. Inteligência artifical. I. Título.

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem resarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- Tese (doutorado)
 Dissertação (mestrado)
 Monografia (especialização)
 TCC (graduação)
- Produto técnico e educacional - Tipo:
- Artigo científico
 Capítulo de livro
 Livro
 Trabalho apresentado em evento

Nome completo do autor:

Douglas Cássio Reis Silva Oliveira

Matrícula:

2022103202030055

Título do trabalho:

CHATBOTS INTELIGENTES: Uma Análise Comparativa de Tecnologias e seus Impactos na Experiência do Usuário em Plataformas Digitais

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: Não Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 10 /12 /2025

O documento está sujeito a registro de patente? Sim Não

O documento pode vir a ser publicado como livro? Sim Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

- Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
- Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
- Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Documento assinado digitalmente
gov.br
DOUGLAS CASSIO REIS SILVA OLIVEIRA
Data: 11/12/2025 19:06:55-0300
Verifique em <https://validar.iti.gov.br>

Ceres
Local

11 /12 /2025
Data

Ciente e de acordo:

Assinatura do autor e/ou detentor dos direitos autorais

Documento assinado digitalmente

gov.br
VILSON SOARES DE SIQUEIRA
Data: 11/12/2025 20:59:57-0300
Verifique em <https://validar.iti.gov.br>



SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÉNCIA E TECNOLOGIA GOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

No primeiro dia do mês de dezembro do ano de dois mil e vinte e cinco, realizou-se a defesa de Trabalho de Curso do acadêmico **Douglas Cássio Reis Silva Oliveira**, do Curso de Bacharelado em Sistemas de Informação, matrícula **2022103202030055**, cujo título é “**CHATBOTS INTELIGENTES: Uma Análise Comparativa de Tecnologias e seus Impactos na Experiência do Usuário em Plataformas Digitais**”. A defesa iniciou-se às 14 horas e 02 minutos, finalizando-se às 14 horas e 55 minutos. A banca examinadora considerou o trabalho **APROVADO** com média **8,06** no trabalho escrito, média **9,03** no trabalho oral, apresentando assim média aritmética final de **8,65** pontos, estando o estudante **APTO** para fins de conclusão do Trabalho de Curso.

Após atender às considerações da banca e respeitando o prazo disposto em calendário acadêmico, o estudante deverá fazer a submissão da versão corrigida em formato digital (.pdf) no Repositório Institucional do IF Goiano – RIIF, acompanhado do Termo Ciência e Autorização Eletrônico (TCAE), devidamente assinado pelo autor e orientador.

Os integrantes da banca examinadora assinam a presente.

(Assinado Eletronicamente)
Prof. Dr. Vilson Soares de Siqueira

Orientador

(Assinado Eletronicamente)
Profª. Drª. Maryele Lazara Rezende,

Membro

(Assinado Eletronicamente)
Prof. Igor Justino Rodrigues

Membro

Documento assinado digitalmente
gov.br IGOR JUSTINO RODRIGUES
Data: 11/12/2025 15:36:56-0300
Verifique em <https://validar.itи.gov.br>

Documento assinado eletronicamente por:

- **Vilson Soares de Siqueira, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/12/2025 15:09:10.
- **Maryele Lazara Rezende, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/12/2025 16:38:19.

Este documento foi emitido pelo SUAP em 30/11/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifgoiano.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 768285

Código de Autenticação: 2b7c1d4406



INSTITUTO FEDERAL GOIANO
Campus Ceres
Rodovia GO-154, Km 03, SN, Zona Rural, CERES / GO, CEP 76300-000
(62) 3307-7100

RESUMO

Este trabalho realiza uma análise comparativa do desempenho funcional e da experiência do usuário em plataformas de IA generativa (LLMs), especificamente Google Gemini, OpenAI ChatGPT e Microsoft Copilot, aplicadas como ferramentas de produtividade para estudantes, profissionais e entusiastas da tecnologia. Utilizou-se uma metodologia mista, combinando revisão bibliográfica com testes práticos controlados pelo pesquisador, avaliando métricas de tempo de resposta, precisão factual, usabilidade na geração de código, naturalidade conversacional e satisfação do usuário. As interações foram registradas e avaliadas qualitativa e quantitativamente por meio de rubricas baseadas em conceitos de Engenharia de Software e Experiência do Usuário. Os dados obtidos revelaram que, embora haja nivelamento na precisão factual (Cenário A) e na fluidez conversacional (Cenário C), a divergência crítica está associada com a usabilidade (Cenário B) e na confiabilidade geral. O Google Gemini demonstrou superioridade ao ser a única plataforma a aderir às boas práticas de Engenharia de Software, enquanto o OpenAI ChatGPT apresentou uma falha técnica crítica, comprometendo sua confiabilidade. Conclui-se que a eficácia dessas ferramentas para o utilizador profissional é determinada não apenas pela precisão da resposta, mas pela eficiência, confiabilidade e, crucialmente, pela aderência às boas práticas de engenharia.

Palavras-chave: Experiência do usuário. Sistemas. Comparação de tecnologias. Inteligência artificial.

ABSTRACT

This paper performs a comparative analysis of functional performance and user experience in generative AI platforms (LLMs) specifically Google Gemini, OpenAI ChatGPT, and Microsoft Copilot applied as productivity tools for students, professionals, and technology enthusiasts. A mixed-method approach was employed, combining a literature review with researcher-controlled practical tests to evaluate metrics regarding response time, factual accuracy, code generation usability, conversational naturalness, and user satisfaction. Interactions were recorded and assessed both qualitatively and quantitatively using rubrics based on Software Engineering and User Experience concepts. The obtained data revealed that, while there is parity in factual accuracy (Scenario A) and conversational fluency (Scenario C), critical divergence is associated with usability (Scenario B) and overall reliability. Google Gemini demonstrated superiority as the only platform to adhere to Software Engineering best practices, while OpenAI ChatGPT presented a critical technical failure, compromising its reliability. It is concluded that the effectiveness of these tools for the professional user is determined not only by response accuracy but also by efficiency, reliability, and, crucially, adherence to engineering best practices.

Keywords: User experience. Systems. Technology comparison. Artificial intelligence.

SUMÁRIO

1 INTRODUÇÃO.....	1
2 REVISÃO DE LITERATURA.....	3
2.1 Chatbots Inteligentes: Conceito e Evolução.....	3
2.2 Processamento de Linguagem Natural (NLP).....	4
2.3 Machine Learning aplicado a Chatbots.....	4
2.4 Precisão na Resposta.....	5
2.5 Naturalidade da Conversa.....	6
2.6 Usabilidade.....	6
2.7 Satisfação do Usuário.....	7
3 METODOLOGIA.....	10
3.1 Delineamento da Pesquisa e Ambiente Controlado.....	10
3.2 Etapas da Pesquisa.....	11
3.2.1 Levantamento Teórico:.....	11
3.2.2 Seleção das Ferramentas:.....	11
3.3 Definição dos Cenários e Testes Padronizados.....	12
3.3.1 Protocolo de Execução e Validação Técnica.....	14
3.4 Coleta e Análise de Dados.....	14
3.5 Participantes e Ambiente.....	15
4 ANÁLISE DE RESULTADOS.....	15
4.1 Análise de Desempenho.....	18
4.2 Análise de Precisão Factual.....	19
4.3 Análise de Clareza e Usabilidade.....	19
4.4 Análise de Naturalidade da Conversa.....	20
4.5 Análise de Confiabilidade e Satisfação do Usuário.....	20
5 CONCLUSÃO.....	22
REFERÊNCIAS.....	24
APÊNDICE A: TRANSCRIÇÃO DOS TESTES DO CENÁRIO A.....	28
APÊNDICE B: TRANSCRIÇÃO DOS TESTES DO CENÁRIO B.....	53

APÊNDICE C: TRANSCRIÇÃO DOS TESTES DO CENÁRIO C.....	73
APÊNDICE D: RUBRICAS DE AVALIAÇÃO QUALITATIVA.....	94
APÊNDICE E: GABARITO DE REFERÊNCIA PARA AVALIAÇÃO TÉCNICA.....	98

1 INTRODUÇÃO

Com a expansão da inteligência artificial gerativa, os *chatbots* inteligentes baseados em grandes modelos de linguagem, ou LLMs, tornaram-se ferramentas amplamente utilizadas para interação automatizada em ambientes digitais. Mais do que atender utilizadores em contextos empresariais, essas IAs agora desempenham papéis centrais em tarefas de suporte, criação de conteúdo, resolução de dúvidas técnicas e geração de código. Essas soluções são impulsionadas por tecnologias fundamentais como o Processamento de Linguagem Natural (NLP) e o *Machine Learning* (Aprendizado de Máquina). Elas evoluíram para sistemas generativos capazes de simular conversas humanas com alto grau de coerência, contexto e personalização, sendo exemplificadas por plataformas proeminentes como ChatGPT, Gemini e Copilot. *Chatbots* modernos não apenas executam tarefas predefinidas, mas também se adaptam ao contexto, reconhecem intenções, extraem entidades e aprendem com interações anteriores. Conforme detalhado por McTear (2017), isso os torna ferramentas estratégicas tanto para o atendimento ao cliente quanto para funções internas nas organizações. O avanço de modelos de linguagem de grande escala, incluindo *Bidirectional Encoder Representations for Transformers* (DEVLIN et al., 2019) e o *Generative Pre-trained Transformer* (RADFORD et al., 2018), aumentou significativamente a sofisticação das respostas geradas. Este progresso permite conversas mais fluidas e relevantes, um ponto reforçado pela extensiva revisão de (Deriu et al., 2021) sobre os métodos de avaliação para sistemas de diálogo.

A onipresença dessas ferramentas é confirmada por dados recentes que indicam uma rápida integração massiva na sociedade. Estudos apontam que cerca de 10% da população adulta mundial já utiliza regularmente modelos de linguagem como o *ChatGPT* (BICK et al., 2024). No ambiente corporativo, pesquisas revelam que 54% das empresas já implementaram alguma forma de IA conversacional para otimizar processos (VENTION, 2024). Da mesma forma, no contexto acadêmico, Abdaljaleel et al. (2024) destacam uma alta taxa de aceitação e uso por estudantes como ferramenta de apoio à aprendizagem. Apesar dessa ampla adoção de *chatbots* e assistentes baseados em IA em diferentes domínios, que vão desde plataformas de atendimento até assistentes de produtividade, observa-se que a qualidade das respostas e a experiência oferecida variam significativamente entre as principais IAs

generativas atuais. Esses desafios levantam a problemática central desta pesquisa: Quais são as diferenças entre as principais plataformas de IA conversacional generativa quanto à experiência do utilizador e ao desempenho funcional em interações digitais?. A partir dessa questão, este estudo parte da hipótese de que, embora diversas soluções utilizem tecnologias semelhantes, elas apresentam desempenhos funcionalmente distintos. Espera-se encontrar variações mensuráveis em métricas críticas como tempo de resposta, precisão na resposta, usabilidade, naturalidade da conversa e satisfação do utilizador. Conforme destacado por Trivedi (2020), que investigou a qualidade da interação e seus impactos diretos na satisfação, essas diferenças podem impactar diretamente a percepção do utilizador, a fluidez da interação e a efetividade do atendimento automatizado.

A justificativa desta pesquisa apoia-se na crescente importância das plataformas de IA generativa (LLMs) como ferramentas de produtividade para desenvolvedores, estudantes e profissionais de tecnologia. Elas atuam como assistentes vitais na geração de código e na resolução de dúvidas técnicas complexas, definindo o escopo prático deste estudo. Estudar e comparar essas soluções permite identificar quais abordagens são mais eficazes na prática, oferecendo subsídios técnicos valiosos para profissionais de Sistemas de Informação. A comparação rigorosa oferece referenciais para pesquisadores e melhorias diretas para os utilizadores finais. Além disso, compreender quais fatores técnicos influenciam positivamente a experiência do utilizador, como sugerido por Cooper et al. (2014), contribui para o desenvolvimento de sistemas conversacionais mais éticos, eficientes e acessíveis.

Diante disso, este trabalho tem como objetivo geral comparar diferentes soluções de *chatbots* inteligentes (Google Gemini, OpenAI ChatGPT e Microsoft Copilot) em relação à sua eficácia e impacto na experiência do usuário.

Para alcançar esta meta, definem-se os seguintes objetivos específicos:

- Identificar e selecionar as principais soluções de *chatbot* baseadas em LLMs amplamente utilizadas no mercado;
- Avaliar aspectos técnicos das ferramentas, especificamente: tempo de resposta, precisão na resposta, naturalidade da conversa e usabilidade;

- Medir a satisfação percebida do utilizador por meio de testes práticos e avaliação sistemática baseada em rubricas;
- Apontar vantagens, limitações e melhores práticas observadas na implementação de cada ferramenta para fins de produtividade profissional.

2 REVISÃO DE LITERATURA

O avanço contínuo das tecnologias de inteligência artificial, especialmente no domínio da interação homem-máquina, tem transformado profundamente os paradigmas de atendimento ao público nas organizações (RUSSELL; NORVIG, 2020). Dentre o arsenal de ferramentas tecnológicas emergentes, os *chatbots* inteligentes destacam-se como soluções estratégicas, valorizadas por sua capacidade de automatizar interações complexas e otimizar custos operacionais (ADAMOPOULOU; MOUSSIADES, 2020). Fundamentalmente, essas ferramentas buscam melhorar a experiência do usuário em plataformas digitais, um fator crítico para a aceitação da tecnologia (ZHENG et al., 2022).

Para fornecer uma base teórica robusta para a análise comparativa deste trabalho, este referencial teórico está metodicamente organizado nos eixos centrais que definem e avaliam esses sistemas. A revisão inicia-se pela contextualização do conceito e evolução dos *chatbots* inteligentes, seguida pelos pilares tecnológicos que permitem sua operação sofisticada, especificamente o Processamento de Linguagem Natural (NLP) e o *Machine Learning* aplicado a *chatbots*. Subsequentemente, o capítulo aprofunda-se nas métricas de avaliação que formam o núcleo desta pesquisa: precisão na resposta, naturalidade da conversa, usabilidade da interface e a satisfação do usuário (TRIVEDI, 2020).

2.1 Chatbots Inteligentes: Conceito e Evolução

A definição fundamental de *chatbots* os descreve como sistemas computacionais projetados especificamente para simular interações humanas, seja por meio de mensagens textuais ou comandos de voz. O advento de tecnologias como Processamento de Linguagem Natural (NLP) e *Machine Learning* permitiu que esses sistemas fossem classificados como 'inteligentes'. Conforme detalhado

classicamente por McTear (2017) e atualizado por pesquisas recentes como as de Kuhail et al. (2023), essa inteligência é demonstrada não apenas pela execução de regras, mas pela capacidade de aprendizado contínuo, raciocínio emergente e adaptação dinâmica ao contexto das interações. A trajetória evolutiva desses sistemas é vasta, partindo de implementações rudimentares baseadas em regras, como o notório exemplo ELIZA, até as plataformas robustas e modernas, exemplificadas por Watson Assistant e Dialogflow. Esta transformação crucial marca a passagem de respostas estáticas e predefinidas para agentes conversacionais dinâmicos e personalizados, que agora são capazes de operar com eficácia em ambientes corporativos complexos.

2.2 Processamento de Linguagem Natural (NLP)

O Processamento de Linguagem Natural, ou NLP, é a tecnologia fundamental que permite aos computadores interpretar e gerar a linguagem humana. Este campo envolve um conjunto de tarefas complexas, como detalhado por Jurafsky e Martin (2023), que incluem a análise sintática, a identificação de intenções, o reconhecimento de entidades e a geração de respostas contextualizadas. A execução bem-sucedida dessas capacidades é essencial para que os *chatbots* compreendam os comandos dos usuários de forma eficiente, indo além da simples correspondência de palavras-chave. A evolução desta área é diretamente responsável pela eficácia dos sistemas modernos; modelos recentes como o BERT (DEVLIN et al., 2019) e o Generative Pre-trained Transformer, ou GPT (RADFORD et al., 2018), elevaram drasticamente o nível de sofisticação dos *chatbots*. Esta nova geração de modelos permite uma capacidade muito maior de compreensão semântica profunda e de manutenção da coerência ao longo das interações.

2.3 *Machine Learning* aplicado a *Chatbots*

O *Machine Learning* é um componente crucial que permite aos *chatbots* transcenderem roteiros estáticos e se aprimorarem continuamente. Esta capacidade de melhoria é alcançada através do reconhecimento de padrões em grandes volumes de dados de interações anteriores, permitindo ao sistema "aprender" com seu próprio desempenho histórico. As abordagens metodológicas para este aprendizado são diversas, abrangendo o aprendizado supervisionado, o não supervisionado e o aprendizado por reforço (RUSSELL; NORVIG, 2020). No

contexto dos *chatbots*, o aprendizado supervisionado é fundamental e constitui a base de ferramentas como Rasa e Dialogflow. Nesses sistemas, modelos são treinados com dados históricos previamente rotulados para executar as tarefas centrais de classificação de intenções, que determina o objetivo do usuário, e extração de entidades, que identifica informações relevantes na entrada do usuário. O aprendizado não supervisionado pode ser aplicado para agrupar solicitações de usuários, ajudando a descobrir novas intenções ou padrões emergentes. O aprendizado por reforço, por sua vez, oferece um mecanismo para o *chatbot* otimizar suas estratégias de diálogo ao longo do tempo, ajustando suas respostas com base no *feedback* implícito ou explícito do usuário para maximizar a eficácia da interação. É esse treinamento constante sobre dados históricos que permite o aumento progressivo da eficácia e da precisão das respostas do *chatbot*.

2.4 Precisão na Resposta

A Precisão na Resposta é uma métrica central que se refere à capacidade do *chatbot* de interpretar corretamente a intenção do usuário e, subsequentemente, fornecer uma resposta que seja simultaneamente relevante e adequada ao contexto solicitado. O sucesso nesta dimensão está diretamente relacionado à qualidade do modelo de NLP implementado, que rege a classificação da intenção, bem como à profundidade e acurácia da base de conhecimento do agente conversacional. Erros na interpretação de intenções ou a geração de respostas factualmente incorretas, um fenômeno por vezes referido como "alucinação" em LLMs, comprometem severamente a eficácia da interação, podendo levar à frustração do usuário e ao fracasso na conclusão de tarefas. Por esta razão, a acurácia das respostas torna-se um critério fundamental para a avaliação objetiva do desempenho funcional dos *chatbots*. Conforme destacado pela abrangente revisão de Deriu et al. em 2021, a mensuração desta precisão é intrinsecamente complexa. Muitas métricas automáticas desenvolvidas para a avaliação de diálogo ainda não conseguem substituir o julgamento humano, que é mais apto a capturar nuances, ambiguidades e o contexto implícito, evidenciando a dificuldade em validar a precisão em interações de linguagem natural.

2.5 Naturalidade da Conversa

A Naturalidade da Conversa é uma métrica qualitativa definida como o grau em que a interação com um *chatbot* consegue se aproximar dos complexos padrões comunicativos humanos. Esta métrica avalia a capacidade do sistema de ir além de respostas puramente funcionais, focando na forma como a informação é entregue. *Chatbots* que utilizam modelos de NLP avançados para processar nuances, gerar respostas contextuais e empregar expressões idiomáticas naturais são capazes de promover um nível superior de engajamento e satisfação dos usuários. A importância desta métrica é confirmada por McTear (2017), que destaca como a fluidez e o tom da conversa, juntamente com a coerência mantida ao longo do fluxo do diálogo, influenciam diretamente a aceitação da tecnologia pelo usuário final. Esta avaliação torna-se ainda mais pertinente no escopo desta pesquisa, pois Trivedi em 2020 reforça que a qualidade da interação, que inclui a naturalidade e a clareza, tem impacto direto na satisfação do usuário. A retenção de contexto, em particular, é um aspecto crítico para a correta avaliação de agentes conversacionais avançados baseados em grandes modelos de linguagem, as mesmas tecnologias investigadas neste trabalho.

2.6 Usabilidade

A Usabilidade é um conceito central da Interação Humano-Computador que, no contexto deste trabalho, refere-se à facilidade de uso da interface do *chatbot* e à capacidade do usuário de completar suas tarefas com eficiência, eficácia e satisfação. Conforme estabelecido por Nielsen (1993), uma boa usabilidade é um atributo multifacetado que envolve aspectos cruciais. Entre eles estão a consistência da interface, que garante que elementos e ações similares tenham o mesmo resultado; o *feedback* claro, que informa ao usuário o que está acontecendo; um design intuitivo, que reduz a carga cognitiva; e uma baixa curva de aprendizado, permitindo que o usuário se torne proficiente rapidamente. No domínio específico dos *chatbots*, a usabilidade transcende o design visual e foca na arquitetura da conversação: quanto fácil é para o usuário navegar no diálogo, corrigir erros e alcançar seu objetivo final. A avaliação formal dessa usabilidade pode ser realizada por meio de ferramentas padronizadas, como o System Usability Scale (SUS), um questionário amplamente validado para medir a percepção subjetiva da facilidade de

uso. Além disso, a usabilidade pode ser avaliada pela observação direta da experiência do usuário, analisando o sucesso, o tempo e os erros na execução de tarefas predefinidas.

2.7 Satisfação do Usuário

A satisfação do usuário constitui um indicador subjetivo, porém essencial, para a avaliação holística da efetividade de sistemas conversacionais. Esta métrica representa a percepção global do usuário quanto à qualidade da experiência oferecida pelo *chatbot*. Ela abrange um julgamento multifacetado sobre se a interação foi útil, o que se relaciona diretamente com a Precisão da Resposta; amigável, o que se conecta à Naturalidade da Conversa; confiável, indicando a consistência e ausência de erros; e eficaz, o que está diretamente ligado à Usabilidade e à capacidade de completar tarefas com sucesso. Dada a sua natureza perceptual, pesquisas na área de Experiência do Usuário, comumente referida como UX, frequentemente empregam instrumentos de mensuração como questionários com escalas de Likert para capturar e quantificar essas percepções subjetivas. A importância de atributos qualitativos na formação dessa percepção é reforçada pela literatura recente (ZHENG et al., 2022). Ao analisarem a literatura sobre a interação humano-IA em agentes conversacionais, os autores identificaram que fatores como empatia, adaptabilidade do sistema e, crucialmente, a clareza na formulação da resposta possuem um impacto direto e significativo na satisfação final do usuário.

2.8 Comparativo de Arquiteturas e Direcionamento Estratégico

A capacidade de resposta dos LLMs não é determinada apenas pelo volume bruto de parâmetros, mas pelo direcionamento estratégico definido em seu treinamento (ajuste fino). Nas versões gratuitas avaliadas, cada plataforma adota uma filosofia distinta de gerenciamento de hiperparâmetros, priorizando diferentes aspectos da experiência do usuário, como velocidade, criatividade ou precisão factual.

Embora os detalhes técnicos exatos sejam protegidos por segredo industrial, a literatura técnica e o comportamento das ferramentas permitem mapear o intuito de cada implementação:

Tabela – Arquitetura e Intuito Estratégico

Plataforma	Modelo Base	Intuito do Ajuste de Parâmetros	Comportamento Esperado
Google Gemini	<i>Gemini 1.5 Flash</i>	Eficiência e Multimodalidade: Otimizado para baixa latência e processamento de grandes janelas de contexto. Prioriza respostas diretas e estruturação lógica.	Respostas rápidas e alta capacidade de síntese.
OpenAI ChatGPT	<i>GPT-4o mini</i>	Custo-Efetividade e Fluidez: Projetado como um modelo "destilado" para oferecer inteligência próxima ao GPT-4, mas com menor custo computacional. Foca na naturalidade do diálogo.	Conversa fluida, porém com possíveis oscilações em tarefas de alta complexidade lógica.
Microsoft Copilot	<i>GPT-4 (Balanced)</i>	Ancoragem Factual (Grounding): O modelo atua em conjunto com o motor de busca Bing (<i>Prometheus</i>). O ajuste prioriza a verificação de fatos externos em detrimento da velocidade pura.	Respostas fundamentadas em fontes web, geralmente mais lentas devido à pesquisa em tempo real.

Fonte: Elaborado pelo autor (2025).

Essa distinção estratégica impacta diretamente os resultados observados nos testes práticos. O Google Gemini, ao utilizar uma arquitetura focada em eficiência (*Flash*), tende a entregar respostas mais rápidas e tecnicamente estruturadas. O Microsoft Copilot, por sua vez, sacrifica parte da velocidade de inferência (tempo de resposta) para executar o processo de (buscas na web para validar a resposta), visando mitigar alucinações. Já o ChatGPT, com o modelo mini, busca um equilíbrio entre a fluidez conversacional humana e a economia de recursos computacionais.

3 METODOLOGIA

A presente pesquisa configura-se como um Projeto Experimental Qualitativo de natureza aplicada. Seu propósito central é produzir conhecimento técnico voltado à solução de problemas práticos, investigando a eficácia e o impacto de chatbots inteligentes na experiência do usuário. Para atingir tal objetivo, adota-se um delineamento experimental em ambiente controlado, no qual o pesquisador atua como especialista técnico, mitigando variáveis externas. A metodologia privilegia a análise qualitativa das interações, abrangendo naturalidade, usabilidade e a qualidade da engenharia de software, utilizando indicadores quantitativos, como o tempo de resposta, como alicerce para uma avaliação holística e fundamentada da experiência.

3.1 Delineamento da Pesquisa e Ambiente Controlado

Adota-se um delineamento de pesquisa experimental, uma decisão metodológica justificada pelo caráter exploratório do estudo e pela natureza das ferramentas de IA generativa analisadas. Para assegurar a máxima consistência e padronização dos critérios de avaliação, os testes comparativos entre as plataformas de IA conversacional serão realizados integralmente pelo próprio pesquisador, eliminando assim a variabilidade interpretativa que seria introduzida por múltiplos usuários externos. Diferentemente de um experimento tradicional que exigiria a implementação, o treinamento e o controle de múltiplos *frameworks* de software, esta abordagem concentra-se estrategicamente na avaliação de desempenho de plataformas públicas de larga escala. O conceito de "espaço controlado" é, portanto, redefinido: ele não reside no controle do software em si, que é tratado como uma "caixa-preta", mas sim no controle rigoroso do ambiente de teste e dos estímulos aplicados. O controle dos estímulos é alcançado pela aplicação de comandos, ou *prompts*, padronizados e idênticos em todas as plataformas para cada cenário de teste. Essa configuração metodológica garante o controle experimental, a padronização rigorosa das interações e a replicabilidade das condições de teste. Para reforçar esse rigor, o ambiente de experimentação será padronizado utilizando-se o mesmo equipamento de hardware, a mesma conexão de rede e o mesmo software de navegador. Adicionalmente, uma medida crítica para evitar a contaminação dos resultados pelo histórico contextual das IAs será adotada: as

sessões de chat serão reiniciadas a cada novo cenário, garantindo que cada avaliação seja independente e não sofra influência de interações prévias.

3.2 Etapas da Pesquisa

3.2.1 Levantamento Teórico:

A primeira etapa da pesquisa consistirá em um aprofundado levantamento teórico, que servirá como alicerce para a análise prática. Será realizada uma revisão sistemática da literatura focada nos pilares centrais deste trabalho: a arquitetura de *chatbots*, os fundamentos do Processamento de Linguagem Natural (NLP), as aplicações de *Machine Learning* (Aprendizado de Máquina) em sistemas conversacionais e as metodologias de avaliação de Experiência do Usuário (UX). As fontes para esta revisão incluirão uma gama de materiais acadêmicos, como livros-texto e artigos científicos recentes, com especial atenção a autores cujos trabalhos são seminais para a avaliação de agentes conversacionais modernos. Serão consultados artigos de referência. Além desses artigos, a pesquisa utilizará bases acadêmicas de alto impacto, como a IEEE Xplore, ACM Digital Library, Science Direct e PUBMED, para garantir que o *framework* de avaliação desta dissertação esteja alinhado com o estado da arte da pesquisa em sistemas conversacionais.

3.2.2 Seleção das Ferramentas:

A seleção das ferramentas para esta análise comparativa constitui uma etapa metodológica crucial, pois define o escopo empírico do estudo. Com base nos critérios de relevância de mercado, ampla adoção e disponibilidade de acesso, foram selecionadas as três principais plataformas públicas de Inteligência Artificial conversacional generativa: Google Gemini, OpenAI ChatGPT e Microsoft Copilot. Estas três soluções foram escolhidas por representarem o atual estado da arte na tecnologia de LLMs e por pertencerem às corporações que lideram a inovação no setor, nomeadamente Google, OpenAI e Microsoft. A sua indiscutível relevância de mercado e a ampla adoção por milhões de utilizadores, incluindo desenvolvedores e profissionais de tecnologia, tornam a comparação direta pertinente para os objetivos desta pesquisa. A disponibilidade de acesso público dessas plataformas, através de suas interfaces web, é ainda um requisito pragmático que viabiliza a execução da

metodologia experimental e a coleta de dados padronizada, conforme definido neste trabalho. Para assegurar a reproduzibilidade técnica desta análise, optou-se pela utilização das versões de acesso público e gratuito, vigentes durante o período de coleta de dados, realizado em novembro de 2025. As versões registradas foram:

- Google Gemini: Versão baseada no modelo Gemini 1.5 Flash (otimizado para eficiência e latência).
- OpenAI ChatGPT: Versão baseada no modelo GPT-4o mini (modelo leve focado em custo-efetividade).
- Microsoft Copilot: Versão gratuita integrada ao Bing, baseada na arquitetura GPT-4 (com foco em ancoragem de busca).

Ressalta-se que, por se tratarem de serviços em nuvem, a métrica de tempo de resposta apresentada nos resultados está sujeita à latência de rede e à carga momentânea dos servidores das provedoras no instante do teste.

3.3 Definição dos Cenários e Testes Padronizados

A definição dos cenários e testes padronizados constitui o núcleo da execução metodológica, assegurando que cada plataforma seja avaliada sob condições idênticas. Cada ferramenta será testada individualmente através de cenários de interação simulados, os quais foram previamente definidos e replicados com exatidão em todas as plataformas. Estes cenários foram projetados especificamente para isolar e avaliar as métricas centrais do estudo, alinhando-se diretamente com os objetivos da pesquisa de avaliar as IAs como ferramentas de produtividade para profissionais de Sistemas de Informação. O Cenário A, focado na Precisão Factual, foi desenhado para testar a acurácia e a profundidade da base de conhecimento das plataformas, exigindo respostas a dúvidas técnicas e objetivas da área de Sistemas de Informação, como os conceitos de Normalização de Banco de Dados e a comparação de modelos de processo de software. O Cenário B, centrado na Resolução de Problemas, foi utilizado para medir a Clareza e Usabilidade. Esta avaliação foi realizada através da capacidade da IA de gerar e explicar algoritmos ou *scripts*, focando não apenas na funcionalidade do código, mas na sua aderência às boas práticas de engenharia e na clareza da explicação fornecida. Finalmente, o Cenário C, focado na Interação Conversacional, foi desenhado para avaliar a

Naturalidade da Conversa e a vital capacidade de retenção de contexto, o que foi testado através de um diálogo multi-turno sobre um tema complexo da área. Em conformidade com o delineamento experimental, todas as interações serão conduzidas e registradas sistematicamente pelo pesquisador. Este, por sua vez, fará a avaliação qualitativa e quantitativa conforme os critérios definidos no referencial teórico deste trabalho.

Tabela de *Prompts*

<i>Prompts para testes de cenários</i>					
Cenário A: Teste de Precisão Factual		Cenário B: Teste de Resolução de Problemas		Cenário C: Teste de Interação Conversacional	
<i>Prompt A1</i> (Banco de Dados)	<i>Prompt A2</i> (Engenharia de Software)	<i>Prompt B1</i> (Algoritmo)	<i>Prompt B2</i> (Código Web)	<i>Prompt C1</i> (Tópico Complexo)	<i>Prompt C2</i> (Manutenção de Contexto)
Descreva o que é Normalização de Banco de Dados e explique as três primeiras formas normais (1FN, 2FN e 3FN), fornecendo um exemplo simples para cada uma.	Compare os modelos de processo de software 'Waterfall' e 'Scrum', destacando as principais diferenças na abordagem de requisitos, planejamento e entregas.	Gere um <i>script</i> em Python que implemente o algoritmo de busca binária. O código deve estar comentado. Após o bloco de código, explique brevemente a lógica da função e sua complexidade e de tempo.	Gere o código HTML, CSS e JavaScript para um formulário de login simples. O JavaScript deve validar no front-end se os campos 'email' e 'senha' estão preenchidos antes de permitir o envio, exibindo um alerta se estiverem vazios.	Qual é o impacto da Lei Geral de Proteção de Dados no desenvolvimento de software e na gestão de bancos de dados no Brasil?	Baseado no que você acabou de explicar, quais são as responsabilidades específicas de um 'Analista de Sistemas' ou 'DBA' para garantir a conformidade com essa lei em um projeto de e-commerce ?

3.3.1 Protocolo de Execução e Validação Técnica

A interação com as plataformas seguiu a abordagem tiro único, onde o prompt é enviado uma única vez sem refinamentos posteriores ou engenharia de prompt iterativa. Essa escolha visa simular a dúvida rápida de um desenvolvedor e avaliar a capacidade da IA de compreender e resolver o problema na primeira tentativa.

Para a validação das respostas do Cenário B (Geração de Código), adotou-se o método de Inspeção de Software (Code Inspection). Segundo Pressman (2016), esta é uma técnica formal de verificação estática eficaz para identificar erros de lógica e desvios de padrões sem a necessidade imediata de execução dinâmica. O pesquisador, atuando como especialista do domínio, avaliou os artefatos gerados (algoritmos em Python e estruturas Web) verificando:

- Correção Sintática: Se a estrutura do código respeita as regras das linguagens solicitadas;
- Lógica Algorítmica: Se a implementação atende funcionalmente ao problema proposto no prompt;
- Boas Práticas: Se houve separação de responsabilidades (no caso Web) e clareza na nomenclatura e comentários.

Desta forma, a análise técnica permitiu constatar divergências críticas na qualidade de engenharia (como a falta de modularização de arquivos) que impactam a manutenção, independentemente da execução em ambiente de produção.

3.4 Coleta e Análise de Dados

A etapa de coleta e análise de dados seguirá rigorosamente a abordagem mista definida para este estudo, onde os dados obtidos serão sistematicamente organizados em tabelas comparativas para permitir uma análise crítica e direta. Toda a avaliação será guiada pelos critérios fundamentais estabelecidos no referencial teórico, com foco nas métricas centrais de precisão, naturalidade, usabilidade e satisfação percebida, fundamentada em Deriu et al. (2021); naturalidade, baseada em McTear (2017); usabilidade, conforme os princípios de Nielsen (1993); e satisfação percebida, alinhada aos estudos de Zheng et al. (2022). A vertente quantitativa da coleta focará em dois indicadores de desempenho objetivos. O primeiro é o Tempo médio de resposta, que será cronometrado em segundos,

mensurando o intervalo desde o envio do *prompt* até a conclusão da geração da resposta pela plataforma. O segundo é a Precisão das respostas, que será mensurada através de uma nota de 1 a 5, atribuída pelo pesquisador após comparar a resposta fornecida pela IA com a literatura técnica de referência, conforme os critérios de avaliação detalhados no Apêndice D. A vertente qualitativa abrangerá as métricas perceptivas da interação. A Naturalidade da conversa avaliará a fluidez, a coesão textual e o nível de humanização da linguagem em uma escala de 1 a 5 conforme os critérios detalhados no Apêndice D. A Clareza e usabilidade analisará a eficácia da apresentação da informação, incluindo a formatação de blocos de código ou tópicos, a objetividade da resposta e a facilidade geral de compreensão. Finalmente, a Satisfação do usuário será registrada como uma nota consolidada de 1 a 5, baseada na percepção subjetiva de utilidade e eficácia da interação.

3.5 Participantes e Ambiente

A ausência de participação externa de usuários é um aspecto central da metodologia adotada. Todas as interações e a subsequente avaliação serão conduzidas exclusivamente pelo próprio pesquisador, o que garante a consistência da análise comparativa.

Esta abordagem é justificada pela necessidade de manter um ambiente controlado. O controle é exercido por meio da aplicação de tarefas previamente padronizadas e pela eliminação das variáveis de subjetividade inerentes à coleta de dados em larga escala com múltiplos usuários. A padronização rigorosa dos *prompts* e das condições de teste assegura a viabilidade metodológica da pesquisa no contexto de um trabalho de conclusão de curso.

4 ANÁLISE DE RESULTADOS

Este capítulo apresenta e discute os dados coletados durante a fase de testes experimentais, conforme detalhado na Metodologia. O estudo comparativo envolveu as três plataformas de Inteligência Artificial conversacional Google Gemini, OpenAI ChatGPT e Microsoft Copilot. As plataformas foram submetidas a seis *prompts* padronizados, organizados em três cenários temáticos: A, B e C, para uma avaliação multifatorial. As métricas avaliadas incluíram o Tempo médio de resposta, a Precisão da Resposta, a Clareza e Usabilidade, a Naturalidade da Conversa e a

Satisfação do Usuário, cujos resultados quantitativos de tempo e as notas qualitativas, atribuídas pelo pesquisador conforme os critérios metodológicos, estão consolidados nas Tabelas “Tempo de resposta (s)” e “Avaliação qualitativa de respostas”.

Tabela – Tempo de resposta (s)

Tempo de resposta (s)							
Plataforma	<i>Prompt A1</i>	<i>Prompt A2</i>	<i>Prompt B1</i>	<i>Prompt B2</i>	<i>Prompt C1</i>	<i>Prompt C2</i>	Tempo Médio
Google Gemini	11,93	11,33	10,25	9,28	11	10,56	10,73
OpenAI ChatGPT	23,77	48,29	13,51	17,08	25,1	32,25	26,67
Microsoft Copilot	16,26	15,02	12,15	25	17,5	12,95	16,48

Fonte: Elaborado pelo autor com base na aplicação das rubricas de avaliação (2025).

Tabela – Avaliação qualitativa de respostas (1 a 5)

Avaliação qualitativa de respostas							
	Métrica	Prompt A1 (Banco de Dados)	Prompt A2 (Engenharia de Software)	Prompt B1 (Algoritmo)	Prompt B2 (Código Web)	Prompt C1 (Tópico Complexo)	Prompt C2 (Manutenção de Contexto)
Google Gemini	Precisão Factual (1-5)	5	5	5	5	5	5
	Clareza e Usabilidade (1-5)	5	5	5	5	5	5
	Naturalidade da conversa (1-5)	5	5	5	5	5	5
OpenAI ChatGPT	Métrica	Prompt A1 (Banco de Dados)	Prompt A2 (Engenharia de Software)	Prompt B1 (Algoritmo)	Prompt B2 (Código Web)	Prompt C1 (Tópico Complexo)	Prompt C2 (Manutenção de Contexto)
	Precisão Factual (1-5)	5	5	5	5	5	5
	Clareza e Usabilidade (1-5)	5	5	2	2	5	5
	Naturalidade da conversa (1-5)	5	5	5	5	5	5

	Métrica	<i>Prompt A1</i> (Banco de Dados)	<i>Prompt A2</i> (Engenharia de Software)	<i>Prompt B1</i> (Algoritmo)	<i>Prompt B2</i> (Código Web)	<i>Prompt C1</i> (Tópico Complexo)	<i>Prompt C2</i> (Manutenção de Contexto)
Microsoft Copilot	Precisão Factual (1-5)	5	5	5	5	5	5
	Clareza e Usabilidade (1-5)	5	5	2	2	5	5
	Naturalidade da conversa (1-5)	5	5	5	5	5	5

Fonte: Elaborado pelo autor com base na aplicação das rubricas de avaliação (2025).

A análise detalhada de cada métrica revela que, embora as plataformas demonstrem um nivelamento no conhecimento técnico e na fluidez conversacional, elas divergem significativamente em usabilidade, eficiência e confiabilidade.

4.1 Análise de Desempenho

A análise do desempenho, medida pelo tempo médio de resposta, reflete o ciclo completo de interação desde o envio do *prompt* até a conclusão da geração da resposta. A plataforma Google Gemini destacou-se como a mais eficiente, registrando o menor tempo médio com 10,73 segundos. Em contraste, a plataforma Microsoft Copilot apresentou um desempenho intermediário, com 16,48 segundos. A plataforma OpenAI ChatGPT foi significativamente mais lenta, registrando uma média de 26,67 segundos, o que representa mais que o dobro do tempo despendido pela plataforma mais rápida para concluir as mesmas tarefas. Esta lentidão acentuada demonstra uma diferença crítica na eficiência das plataformas, impactando diretamente a experiência do usuário e podendo gerar considerável frustração em cenários de produtividade.

4.2 Análise de Precisão Factual

A Análise de Precisão Factual, centrada no Cenário A, foi projetada para medir a acurácia das respostas técnicas. Este cenário utilizou o *Prompt A1*, focado em Normalização de Banco de Dados, e o *Prompt A2*, focado na comparação entre os modelos Cascata e Scrum, avaliando as respostas por sua conformidade com a literatura de Sistemas de Informação. Nesta métrica, observou-se uma convergência total, com as três plataformas alcançando a Nota 5 (Perfeita). As respostas fornecidas foram 100% corretas, completas e alinhadas com os conceitos técnicos de autores de referência na área, como Elmasri e Navathe (2011) para banco de dados e Pressman (2016) para engenharia de software (conforme detalhado nas transcrições do Apêndice A). Este resultado de paridade sugere que, para domínios de conhecimento técnico consolidado, a base de dados das três plataformas é igualmente robusta e precisa. A diferenciação competitiva entre as ferramentas, portanto, não reside no o que elas respondem, mas, como será visto na próxima seção, no como elas apresentam e estruturam essa informação.

4.3 Análise de Clareza e Usabilidade

O Cenário B, focado na geração de código, foi o ponto de maior divergência e o mais revelador da análise, testando a capacidade das IAs de gerar soluções de forma clara e usável, conforme a métrica definida na Seção 2.6. No *Prompt B1*, que solicitava um algoritmo de Busca Binária, observou-se um nivelamento, com as três plataformas apresentando um desempenho Nota 5 (Perfeito). Todas geraram código Python funcional, com excelente formatação, comentários didáticos e explicações precisas sobre a lógica algorítmica e a complexidade de tempo. Contudo, a execução do *Prompt B2*, que solicitava um formulário web HTML, CSS e JS, revelou uma diferença drástica na qualidade da implementação. A plataforma Google Gemini foi a única a receber Nota 5 (Perfeita), pois demonstrou aderência às melhores práticas de Engenharia de Software ao separar a solução em três arquivos distintos, respectivamente index.html, style.css e script.js. Esta abordagem entrega um código limpo, profissional e de alta Usabilidade, pois garante a manutenibilidade da solução. Em contrapartida, as plataformas OpenAI ChatGPT e Microsoft Copilot receberam Nota 3 (Razoável). Embora ambas tenham entregue um código funcional, o fizeram consolidando toda a estrutura, estilização e lógica em um único arquivo. Esta

abordagem representa uma má prática de desenvolvimento que prejudica severamente a Usabilidade em termos de manutenção e modularidade, além de suas explicações sobre o código terem sido superficiais. A comparação visual entre a estrutura organizada do Gemini e os blocos únicos das demais plataformas pode ser consultada no Apêndice B. Este resultado indica que a plataforma Google Gemini possui uma implementação superior para tarefas de engenharia de software, priorizando a usabilidade de longo prazo e as boas práticas da área, enquanto as concorrentes focaram apenas na entrega funcional imediata da tarefa, ignorando a qualidade da implementação.

4.4 Análise de Naturalidade da Conversa

A análise da Naturalidade da Conversa, métrica definida na Seção 2.5, foi conduzida através do Cenário C. Este cenário foi projetado especificamente para medir a fluidez do diálogo e a capacidade de retenção de contexto das plataformas. Os resultados evidenciaram uma equivalência de desempenho no patamar máximo, com as três plataformas alcançando a Nota 5 (Perfeita). O teste principal de inferência contextual, o *Prompt C2*, exigia que as plataformas compreendessem a referência anafórica "essa lei". Todas as soluções identificaram corretamente que o termo se referia à LGPD, o tópico da interação anterior C1, mantendo o contexto da conversa perfeitamente, como demonstram os diálogos completos registrados no Apêndice C. Além da retenção de contexto, todas as plataformas demonstraram alta coesão e fluidez, adotando uma interatividade proativa ao encerrar suas respostas com perguntas de acompanhamento relevantes, antecipando possíveis necessidades subsequentes do usuário. Um diferencial sutil foi observado na plataforma Microsoft Copilot, que se destacou ao personalizar a interação, utilizando o nome do pesquisador, o que eleva a percepção de naturalidade do diálogo.

4.5 Análise de Confiabilidade e Satisfação do Usuário

A Satisfação do Usuário é a métrica subjetiva final deste estudo, consolidando a percepção geral de eficácia, confiabilidade e experiência, conforme definido na Seção 2.7 do referencial teórico. Esta nota única reflete o desempenho agregado da plataforma em todas as dimensões testadas, onde foi registrada a maior disparidade entre as soluções. A plataforma Google Gemini recebeu a nota máxima, Nota 5 (Excelente), um resultado justificado por seu desempenho superior e equilibrado em

todas as métricas. Ela foi a plataforma mais rápida, com 10,73 segundos, demonstrou a melhor usabilidade, sendo a única a aderir às boas práticas de Engenharia de Software no Cenário B, e foi 100% precisa e natural, sem apresentar falhas técnicas. A plataforma Microsoft Copilot recebeu uma avaliação alta, Nota 4 (Boa). Embora tenha demonstrado precisão e naturalidade perfeitas, sua performance foi mediana em velocidade, com 16,48 segundos, e crucialmente falhou no critério de Usabilidade do *Prompt* B2, entregando uma solução de baixa qualidade de implementação ao utilizar um arquivo único. Por fim, a plataforma OpenAI ChatGPT recebeu a nota mais baixa, Nota 2 (Ruim). Este resultado negativo justifica-se por uma combinação de fatores críticos: foi a plataforma mais lenta em todos os cenários, com 26,67 segundos ; falhou no critério de Usabilidade do *Prompt* B2, apresentando a mesma má prática de implementação das concorrentes ; e, o fator mais grave, demonstrou uma Falha de Confiabilidade, sendo a única plataforma a exibir um erro técnico crítico de "Erro na transmissão de mensagem" durante os testes (evidenciado no Apêndice B). Isso demonstra que a satisfação não deriva apenas da precisão, mas é severamente impactada pela eficiência, qualidade de implementação e, fundamentalmente, pela confiabilidade.

5 CONCLUSÃO

O presente trabalho teve como objetivo principal realizar uma análise comparativa rigorosa entre três das principais plataformas de Inteligência Artificial generativa: Google Gemini, OpenAI ChatGPT e Microsoft Copilot. A pesquisa buscou avaliar como as implementações distintas dessas ferramentas de produtividade impactam a experiência do usuário e o desempenho funcional em tarefas técnicas, conforme proposto na introdução. Para alcançar este objetivo, foi empregada uma abordagem aplicada e mista, que combinou a coleta de dados quantitativos, como o tempo de resposta, com uma análise qualitativa baseada em rubricas de avaliação. A metodologia de testes padronizados, dividida em três cenários distintos, permitiu uma comparação direta e controlada: o Cenário A mediu a Precisão Factual em conceitos de Sistemas de Informação, o Cenário B avaliou a Clareza e Usabilidade na geração de código, e o Cenário C testou a Naturalidade da Conversa e a retenção de contexto.

Os resultados obtidos confirmaram a hipótese central deste estudo: embora as plataformas utilizem fundamentos tecnológicos similares (LLMs), seus desempenhos funcionais e a experiência do usuário resultante são significativamente distintos. A análise revelou um claro nivelamento em dois dos três cenários. Na Precisão Factual (Cenário A) e na Naturalidade da Conversa (Cenário C), todas as plataformas atingiram a nota máxima (5,0).

Elas demonstraram um domínio técnico impecável sobre conceitos complexos de Normalização de Banco de Dados, Engenharia de Software e os impactos da LGPD, além de uma capacidade perfeita de reter contexto. Isso sugere que o conhecimento técnico consolidado e a fluidez conversacional são, atualmente, *commodities* entre os principais concorrentes. A diferenciação crucial, portanto, surgiu na eficiência e, mais importante, na Usabilidade (Cenário B). Embora todas as plataformas tenham gerado algoritmos funcionais, o teste de geração de um formulário web expôs uma divergência crítica: o Google Gemini foi a única plataforma (Nota 5) a entregar a solução aderindo às boas práticas de Engenharia de Software, ao separar o código em múltiplos arquivos. ChatGPT e Copilot (Nota 3) entregaram um código funcional, porém em um único arquivo, uma má prática que demonstra baixa Usabilidade em termos de manutenibilidade profissional. Esta falha de Usabilidade, combinada com

as métricas de desempenho, foi decisiva: Gemini foi o mais rápido (10,73s), enquanto o ChatGPT foi o mais lento (26,67s) e o único a apresentar uma falha técnica crítica ("Erro na transmissão de mensagem") durante os testes, comprometendo severamente sua Confiabilidade.

Do ponto de vista prático, este estudo contribui com subsídios técnicos diretos para desenvolvedores, estudantes e profissionais de Sistemas de Informação, conforme proposto na justificativa da pesquisa. A análise fornece evidências claras de que a seleção de uma IA generativa como ferramenta de produtividade não deve basear-se apenas na sua capacidade de "responder corretamente" (Precisão). O verdadeiro valor para o profissional de S.I. reside na capacidade da ferramenta de gerar soluções que sejam não apenas funcionais, mas também usáveis e aderentes aos padrões de engenharia. O trabalho reforça que a experiência do usuário (UX) em IAs generativas é um construto complexo, determinado pela qualidade da implementação (Clareza e Usabilidade) e pela consistência da plataforma (Confiabilidade e eficiência). A superioridade do Google Gemini na métrica de Satisfação (Nota 5) é um reflexo direto de seu desempenho superior nesses fatores críticos de implementação.

É imperativo, contudo, reconhecer as limitações metodológicas deste estudo. Primeiramente, a avaliação qualitativa realizada por um único avaliador especialista carrega um viés subjetivo inerente às métricas de satisfação e naturalidade; trabalhos futuros beneficiar-se-iam de uma validação cruzada com múltiplos avaliadores (Juízes) ou testes cegos. Além disso, os resultados aqui apresentados restringem-se ao domínio da Engenharia de Software e Programação. A extração para outras áreas, como Finanças, Direito ou Saúde, pode revelar comportamentos distintos, onde a 'criatividade' do modelo (alucinação) pode ser mais prejudicial ou onde a precisão factual tenha peso diferente da usabilidade de código. Portanto, a generalização destes resultados para IAs 'menores' ou treinadas para nichos específicos deve ser feita com cautela. Conclui-se, portanto, que a escolha de uma plataforma de IA generativa para tarefas técnicas deve priorizar não apenas o vasto conhecimento do modelo, mas sua capacidade de traduzir esse conhecimento em soluções usáveis, eficientes e, acima de tudo, confiáveis.

REFERÊNCIAS

- ABDALJALEEL, M. et al. A multinational study on the factors influencing university students' attitudes and usage of ChatGPT. *Scientific Reports*, v. 14, n. 1983, 2024. Disponível em: <https://www.nature.com/articles/s41598-024-52549-8>. Acesso em: 20 nov. 2025.
- ADAMOPOULOU, E.; MOUSSIADES, L. An overview of *chatbot* technology. In: ARTIFICIAL INTELLIGENCE APPLICATIONS AND INNOVATIONS. AIAI 2020. IFIP Advances in Information and Communication Technology. Cham: Springer, 2020. p. 373–383. Disponível em: https://link.springer.com/chapter/10.1007/978-3-030-49186-4_31. Acesso em: 20 nov. 2025.
- BICK, A.; BLANDIN, A.; DEMING, D. J. The Rapid Adoption of Generative AI. NBER Working Paper Series, n. 32966. National Bureau of Economic Research, 2024. Disponível em: <https://www.nber.org/papers/w32966>. Acesso em: 20 nov. 2025.
- COOPER, Alan et al. About face: the essentials of interaction design. 4. ed. Indianapolis: John Wiley e Sons, 2014. Disponível em: &<https://books.google.com.br/books?id=4c4XBAAAQBAJ&h>. Acesso em: 20 nov. 2025.
- DERIU, J. et al. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, v. 54, p. 597–657, 2021. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/33505103>. Acesso em: 20 nov. 2025.
- DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North*

American Chapter of the Association for Computational Linguistics. Minneapolis, MN: Association for Computational Linguistics, 2019. p. 4171-4186. Disponível em: <https://aclanthology.org/N19-1423/>. Acesso em: 20 nov. 2025.

ELMASRI, R.; NAVATHE, S. B. Sistemas de Banco de Dados. 6. ed. São Paulo: Pearson, 2011. Disponível em: https://books.google.com/books/about/Sistemas_de_banco_de_dados.html?hl=pt-BR&id=FSvIYgEACAAJ. Acesso em: 20 nov. 2025.

JURAFSKY, Daniel; MARTIN, James H. Speech and Language Processing. 3rd ed. Draft. Stanford University, 2023. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Acesso em: 20 nov. 2025.

KUHAIL, M. A. et al. Interacting with educational *chatbots*: A systematic review. Education and Information Technologies, v. 28, n. 1, p. 973-1018, 2023. Disponível em: <https://link.springer.com/article/10.1007/s10639-022-11177-3>. Acesso em: 20 nov. 2025.

MCTEAR, M. F. The rise of the conversational interface: a new kid on the block? In: FUTURE AND EMERGING TRENDS IN LANGUAGE TECHNOLOGY. FETLT 2016. Lecture Notes in Computer Science. Cham: Springer, 2017. v. 10341, p. 38–49. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-69365-1_3. Acesso em: 20 nov. 2025.

NIELSEN, Jakob. Usability Engineering. San Francisco: Morgan Kaufmann, 1993. Disponível em: <https://books.google.com.br/books?id=95As2OF67f0C>. Acesso em: 20 nov. 2025.

PRESSMAN, R. S. Engenharia de Software: Uma Abordagem Profissional. 8. ed. Porto Alegre: AMGH, 2016. Disponível em: https://books.google.com/books/about/Engenharia_de_Software_8%C2%AA_Edi%C3%A7%C3%A3o.html?hl=pt-BR&id=wexzCwAAQBAJ. Acesso em: 20 nov. 2025.

RADFORD, A. et al. Improving Language Understanding by Generative Pre-Training. OpenAI Technical Report, 2018. Disponível em: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. Acesso em: 20 nov. 2025.

RUSSELL, Stuart; NORVIG, Peter. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken: Pearson, 2020. Disponível em: <https://aima.cs.berkeley.edu/>. Acesso em: 20 nov. 2025.

TRIVEDI, S. K. Evolution of the use of conversational agents in business education: Past, present, and future. RAM, Rev. Adm. Mackenzie, São Paulo, v. 21, n. 6, eRAMF200028, 2020. Disponível em: <https://www.scielo.br/j/ram/a/kk7tc48ZvFNk5DbmXjGcbPh/?lang=en>. Acesso em: 20 nov. 2025.

VENTION. AI Adoption Statistics 2024: All Figures e Facts to Know. Nova York: Vention, 2024. Disponível em: <https://ventionteams.com/solutions/ai/adoption-statistics>. Acesso em: 20 nov. 2025.

ZHENG, Qiuyue et al. UX research on conversational human-AI interaction: a literature review of the ACM Digital Library. Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, 2022. Disponível em: <https://dl.acm.org/doi/10.1145/3491102.3501855>. Acesso em: 20 nov. 2025.

APÊNDICE A: TRANSCRIÇÃO DOS TESTES DO CENÁRIO A

Figura A.1 – Resposta do Google Gemini ao Prompt A1 (Parte 1)

Descreva o que é Normalização de Banco de Dados e explique as três primeiras formas normais (1FN, 2FN e 3FN), fornecendo um exemplo simples para cada uma.

Mostrar raciocínio ▾

Olá! A **Normalização de Banco de Dados** é um processo fundamental no projeto de bancos de dados relacionais que visa organizar as tabelas de forma eficiente para **minimizar a redundância de dados e evitar anomalias** (problemas) de inserção, atualização e exclusão. Ela é alcançada por meio de um conjunto de regras chamadas **Formas Normais (FN)**, sendo que as três primeiras (1FN, 2FN e 3FN) são as mais utilizadas e importantes na prática.

1. Primeira Forma Normal (1FN) 🎯

Uma tabela está na **Primeira Forma Normal (1FN)** se:

1. Cada coluna contém **valores atômicos** (indivisíveis). Não deve haver atributos compostos ou multivalorados em uma única célula.
2. Não há **grupos repetitivos** de colunas.
3. Cada registro é **único** (normalmente garantido pela existência de uma **Chave Primária**).

Figura A.2 – Resposta do Google Gemini ao Prompt A1 (Parte 2)

Exemplo 1FN

Tabela Não-1FN (Problema: Atributo multivalorado/composto)

ID_Pedido	Cliente	Itens_Comprados
101	Ana	Livro A, Caneta B
102	Bruno	Mouse C

Exportar para as Planilhas

Tabela em 1FN (Solução: Atributos atômicos e eliminação de grupo repetitivo)

ID_Pedido	ID_Item	Cliente	Item
101	1	Ana	Livro A
101	2	Ana	Caneta B
102	1	Bruno	Mouse C

Exportar para as Planilhas

Neste exemplo, a chave primária composta seria `(ID_Pedido, ID_Item)`.

Figura A.3 – Resposta do Google Gemini ao Prompt A1 (Parte 3)

2. Segunda Forma Normal (2FN) 🕵️

Uma tabela está na **Segunda Forma Normal (2FN)** se:

1. Está na **1FN**.
2. **Todos os atributos não-chave** (que não fazem parte da chave primária) devem ter **dependência funcional total** em relação à chave primária. Ou seja, se a chave primária for composta, o atributo não-chave deve depender de todos os componentes da chave, e não apenas de uma *parte* dela (evita-se a **dependência parcial**).

Exemplo 2FN

Tabela Não-2FN (Problema: Dependência Parcial)

ID_Pedido	ID_Produto	Quantidade	Nome_Produto
101	A01	2	Mouse Pad
102	A01	1	Mouse Pad
103	B05	5	Teclado Mecânico

 Exportar para as Planilhas 

Chave Primária Composta: `(ID_Pedido, ID_Produto)`. **Problema:** O `Nome_Produto` depende apenas de `ID_Produto`, que é só **parte** da chave primária. Isso causa redundância (o nome do produto se repete para cada pedido).

Figura A.4 – Resposta do Google Gemini ao Prompt A1 (Parte 4)

Tabelas em 2FN (Solução: Decomposição)

1. **Tabela PEDIDO_ITEM** (Contém as dependências totais): | ID_Pedido | ID_Produto | Quantidade | | :---: | :---: | :---: | | 101 | A01 | 2 | | 102 | A01 | 1 | | 103 | B05 | 5 | (**Chave Primária:** `(ID_Pedido, ID_Produto)`. `Quantidade` depende de ambos.)
2. **Tabela PRODUTO** (Contém a dependência parcial isolada): | ID_Produto | Nome_Produto | | :---: | :---: | | A01 | Mouse Pad | | B05 | Teclado Mecânico | (**Chave Primária:** `ID_Produto`.)

3. Terceira Forma Normal (3FN) 🕵️

Uma tabela está na **Terceira Forma Normal (3FN)** se:

1. Está na **2FN**.
2. Não há **dependências transitivas**. Isso significa que nenhum atributo não-chave pode depender de outro atributo não-chave. Todos os atributos não-chave devem depender **diretamente** da chave primária e de *nenhuma outra coluna não-chave*.

Figura A.5 – Resposta do Google Gemini ao Prompt A1 (Parte 5)

Exemplo 3FN

Tabela Não-3FN (Problema: Dependência Transitiva)

ID_Vendedor	Nome_Vendedor	ID_Loja	Nome_Loja
1	Carlos	20	Centro
2	Maria	30	Sul
3	João	20	Centro

 Exportar para as Planilhas 

Chave Primária: ID_Vendedor. Problema: O Nome_Loja depende de ID_Loja, e ID_Loja depende de ID_Vendedor. Logo, Nome_Loja depende transitivamente de ID_Vendedor (através de ID_Loja). Isso causa redundância (o nome da loja se repete para cada vendedor daquela loja).

Tabelas em 3FN (Solução: Decomposição)

1. **Tabela VENDEDOR** (Contém as dependências diretas): | ID_Vendedor | Nome_Vendedor | ID_Loja | | :---: | :---: | :---: | | 1 | Carlos | 20 | | 2 | Maria | 30 | | 3 | João | 20 | (*Chave Primária: ID_Vendedor. ID_Loja é Chave Estrangeira.*)
2. **Tabela LOJA** (Contém a dependência transitiva isolada): | ID_Loja | Nome_Loja | | :---: | :---: | | 20 | Centro | | 30 | Sul | (*Chave Primária: ID_Loja.*)

Aprenda mais sobre o processo de Normalização de Dados em Bancos de Dados (1FN, 2FN, 3FN).

Figura A.6 – Resposta do Google Gemini ao Prompt A1 (Parte 6)



Figura A.7 – Resposta do Google Gemini ao Prompt A2 (Parte 1)

Compare os modelos de processo de software 'Waterfall' e 'Scrum', destacando as principais diferenças na abordagem de requisitos, planejamento e entregas.

♦ Mostrar raciocínio ▾

É excelente a sua iniciativa de comparar os modelos **Waterfall** e **Scrum**. Eles representam duas filosofias de desenvolvimento de software fundamentalmente diferentes: o Waterfall é um modelo **tradicional**, sequencial e rígido, enquanto o Scrum é um *framework* **Ágil**, iterativo e flexível.

Abaixo, detalhamos as principais diferenças em relação a requisitos, planejamento e entregas.

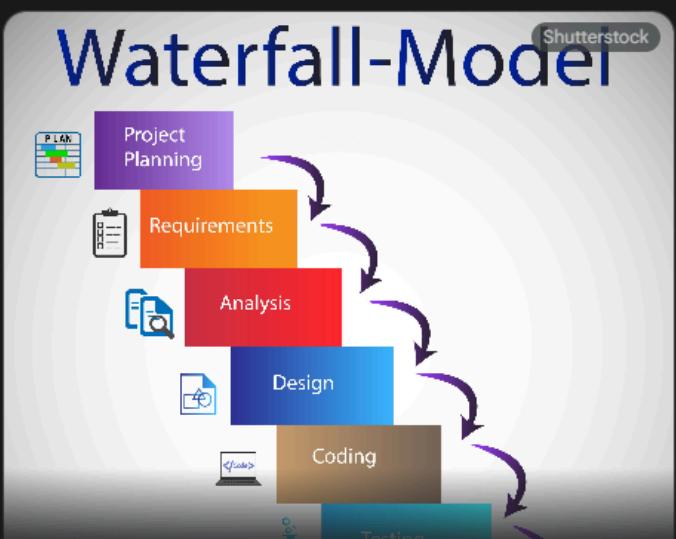


Figura A.8 – Resposta do Google Gemini ao Prompt A2 (Parte 2)



C Modelo Waterfall (Cascata)

O Waterfall é um modelo linear, onde cada fase do desenvolvimento deve ser **concluída** e documentada antes que a próxima possa começar, como uma cascata descendo um rio.

Abordagem de Requisitos

- **Definição Inicial Completa:** Todos os **requisitos** do projeto são levantados, analisados e documentados em **detalhe** no **início** do projeto.
- **Rígido à Mudança:** Mudanças nos requisitos após o início da fase de desenvolvimento são **caras**, demoradas e geralmente desaconselhadas, pois exigem revisitar fases já concluídas.
- **Documentação Extensa:** Há um forte foco na documentação formal e completa dos requisitos.

Planejamento

- **Planejamento Antecipado Total:** O **planejamento** de todo o projeto (escopo, cronograma, custo) é feito **uma única vez** nas fases iniciais.
- **Sequencial e Fixo:** O processo segue uma **sequência de fases** (Requisitos → Design → Implementação → Teste → Implantação) que são executadas de forma rígida.

Figura A.9 – Resposta do Google Gemini ao Prompt A2 (Parte 3)

Entregas

- **Entrega "Big Bang" (Entrega Final):** O produto final ou uma versão significativa do sistema é entregue ao cliente **somente no fim** do projeto.
- **Feedback Tardiamente:** O *feedback* do cliente é recebido muito tarde, geralmente apenas na fase de testes ou após a entrega, o que aumenta o risco de o produto final não atender às necessidades atuais do mercado.

 **Framework Scrum (Ágil)**

O Scrum é um *framework* iterativo e incremental que faz parte da Metodologia Ágil. O trabalho é dividido em ciclos curtos e fixos chamados **Sprints** (geralmente de 2 a 4 semanas).

Abordagem de Requisitos

- **Requisitos Evolutivos:** Os requisitos (chamados de Itens do *Product Backlog*) são definidos em **alto nível** no início e refinados continuamente ao longo da vida do projeto.
- **Adaptável à Mudança:** Mudanças e novas prioridades são **bem-vindas**, pois o *Product Owner* pode reordenar e ajustar o *Product Backlog* antes de cada *Sprint*.
- **Documentação Suficiente:** A documentação é mais **concisa** e foca em entregar o valor do produto ("*Working software over comprehensive documentation*" - Manifesto Ágil).

Planejamento

- **Planejamento Incremental:** O planejamento completo é dividido em **planejamentos de *Sprint*** mais curtos e detalhados (o que será feito na próxima *Sprint*).

Figura A.10 – Resposta do Google Gemini ao Prompt A2 (Parte 4)

- **Iterativo e Flexível:** O projeto avança por meio de **iterações** (Sprints), permitindo ajustes e aprendizados contínuos em cada ciclo. O foco está no ritmo sustentável da equipe.

Entregas

- **Entregas Incremental:** Uma porção funcional e potencialmente *shippable* do produto (“*Incremento*”) é entregue ou revisada ao **final de cada Sprint**.
- **Feedback Constante:** O cliente (*Stakeholders*) fornece *feedback* sobre o *Incremento* na **Sprint Review**, garantindo que o produto esteja sempre alinhado com as necessidades atuais.

Tabela de Comparação

Para maior clareza, veja um resumo das principais diferenças:

Característica	Modelo Waterfall (Cascata)	Framework Scrum (Ágil)
Abordagem	Linear, sequencial, rígida	Iterativa, incremental, flexível
Requisitos	Definidos e documentados totalmente no início	Evolutivos, refinados continuamente (<i>Product Backlog</i>)
Mudanças	Difícil, caras e desaconselhadas	Bem-vindas, incorporadas no planejamento da próxima <i>Sprint</i>
Planejamento	Completo e detalhado no início	Incremental (Planejamento da <i>Sprint</i>), ajustado a cada ciclo
Entrega	Uma única entrega no final do projeto	Entregas frequentes de incrementos (<i>Sprints</i>)

Figura A.11 – Resposta do Google Gemini ao Prompt A2 (Parte 5)

Envolvimento do Cliente	Baixo/Formal (principalmente no início e no fim)	Alto e Contínuo (durante as <i>Reviews</i>)
--------------------------------	--	--

Exportar para as Planilhas

Esta visão geral das metodologias é detalhada neste vídeo que explica o contraste entre os dois modelos. [Gestão de Projetos Waterfall x Metodologias Ágeis](#).

Gostaria de saber mais sobre as **funções** específicas (como *Scrum Master* ou *Product Owner*) no modelo Scrum?

▶ [Gestão de Projetos Waterfall x Metodologias Ágeis - YouTube](#) ⓘ
UNIVESP · 13 mil visualizações



Figura A.12 – Resposta do ChatGPT ao Prompt A1 (Parte 1)

Descreva o que é Normalização de Banco de Dados e explique as três primeiras formas normais (1FN, 2FN e 3FN), fornecendo um exemplo simples para cada uma.

Normalização de Banco de Dados é o processo de **organizar os dados em um banco para reduzir redundâncias** (repetições desnecessárias) e **melhorar a integridade dos dados**. A ideia principal é dividir uma tabela grande e complexa em tabelas menores e relacionadas, garantindo que cada dado seja armazenado **apenas uma vez** e de forma **lógica e eficiente**.

A normalização é dividida em **formas normais (FN)** — cada uma define um nível de organização. Vamos ver as três primeiras, que são as mais utilizadas:

 **Primeira Forma Normal (1FN)**

Definição:

Uma tabela está na **1ª Forma Normal** quando:

- Todos os **atributos** contêm apenas **valores atômicos** (ou seja, **um único valor por célula**).
- Não há **campos multivvalorados** (listas, conjuntos, etc.).

Exemplo (violando 1FN):

ID_Cliente	Nome	Telefones
1	João Silva	(11)99999-1111, (11)98888-2222

 Aqui o campo **Telefones** contém **dois números**, o que viola a 1FN.

Figura A.13 – Resposta do ChatGPT ao Prompt A1 (Parte 2)

👉 Aqui o campo **Telefones** contém **dois números**, o que viola a **1FN**.

Tabela normalizada (em 1FN):

ID_Cliente	Nome	Telefone
1	João Silva	(11)99999-1111
1	João Silva	(11)98888-2222

Agora, **cada campo tem apenas um valor**, atendendo à **1FN**.

✳ Segunda Forma Normal (2FN)

Definição:

Uma tabela está na **2ª Forma Normal** quando:

- Está em **1FN**;
- E todos os atributos não-chave dependem da chave primária inteira, não apenas de parte dela (ou seja, **não há dependência parcial**).

Isso se aplica **somente** a tabelas cuja chave primária é **composta** (formada por mais de um campo).

Exemplo (violando 2FN):

ID_Pedido	ID_Produto	Nome_Produto	Quantidade
1	101	Caneta	10
1	102	Lápis	5

Figura A.14 – Resposta do ChatGPT ao Prompt A1 (Parte 3)

1	102	Lápis	5
<p>👉 Aqui, Nome_Produto depende apenas de ID_Produto, e não da chave composta (ID_Pedido + ID_Produto). Isso é dependência parcial, e viola a 2FN.</p>			
Tabelas normalizadas (em 2FN):			
Pedido_Produto			
ID_Pedido	ID_Produto	Quantidade	
1	101	10	
1	102	5	
Produto			
ID_Produto	Nome_Produto		
101	Caneta		
102	Lápis		
Agora, cada atributo depende somente de sua chave primária .			
 Terceira Forma Normal (3FN)		↓	
Pular para o topo			

Figura A.15 – Resposta do ChatGPT ao Prompt A1 (Parte 4)

Definição:

Uma tabela está na **3^a Forma Normal** quando:

- Está em **2FN**;
- E não há dependência transitiva, ou seja, um campo não depende de outro que não seja a chave primária.

Exemplo (violando 3FN):

ID_Cliente	Nome_Cliente	ID_Cidade	Nome_Cidade
1	João Silva	10	São Paulo

👉 Aqui, **Nome_Cidade** depende de **ID_Cidade**, e **ID_Cidade** depende da **chave primária (ID_Cliente)**. Essa dependência indireta (transitiva) viola a 3FN.

Tabelas normalizadas (em 3FN):

Cliente

ID_Cliente	Nome_Cliente	ID_Cidade
1	João Silva	10

Cidade

ID_Cidade	Nome_Cidade
10	São Paulo



Agora, não há dependências transitivas:

Figura A.16 – Resposta do ChatGPT ao Prompt A1 (Parte 5)

Agora, não há dependências transitivas:
Cada campo depende diretamente da chave primária.

Resumo rápido

Forma Normal	Exige o quê?	Problema que evita
1FN	Valores atômicos	Campos com múltiplos valores
2FN	Sem dependência parcial	Redundância em chaves compostas
3FN	Sem dependência transitiva	Dados derivados em colunas não-chave

Deseja que eu monte um exemplo completo em SQL mostrando uma tabela não normalizada e sua transformação passo a passo até a 3FN?

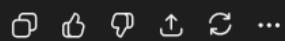


Figura A.17 – Resposta do ChatGPT ao Prompt A2 (Parte 1)

Compare os modelos de processo de software 'Waterfall' e 'Scrum', destacando as principais diferenças na abordagem de requisitos, planejamento e entregas.														
<p>Os modelos de processo de software Waterfall (Cascata) e Scrum representam duas abordagens bem distintas no desenvolvimento de software: o primeiro é tradicional e sequencial, enquanto o segundo é ágil e iterativo.</p> <p>Abaixo está uma comparação clara entre os dois, com foco em requisitos, planejamento e entregas:</p>														
<p>1. Abordagem Geral</p> <table><thead><tr><th>Aspecto</th><th>Waterfall (Cascata)</th><th>Scrum (Ágil)</th></tr></thead><tbody><tr><td>Natureza do processo</td><td>Sequencial e linear — cada fase depende da anterior.</td><td>Iterativo e incremental — o produto evolui em ciclos curtos.</td></tr><tr><td>Mudanças</td><td>Difíceis de implementar após o início do projeto.</td><td>Esperadas e facilmente incorporadas a cada sprint.</td></tr><tr><td>Foco principal</td><td>Documentação e controle do processo.</td><td>Entrega de valor ao cliente e adaptação contínua.</td></tr></tbody></table>			Aspecto	Waterfall (Cascata)	Scrum (Ágil)	Natureza do processo	Sequencial e linear — cada fase depende da anterior.	Iterativo e incremental — o produto evolui em ciclos curtos.	Mudanças	Difíceis de implementar após o início do projeto.	Esperadas e facilmente incorporadas a cada sprint.	Foco principal	Documentação e controle do processo.	Entrega de valor ao cliente e adaptação contínua.
Aspecto	Waterfall (Cascata)	Scrum (Ágil)												
Natureza do processo	Sequencial e linear — cada fase depende da anterior.	Iterativo e incremental — o produto evolui em ciclos curtos.												
Mudanças	Difíceis de implementar após o início do projeto.	Esperadas e facilmente incorporadas a cada sprint.												
Foco principal	Documentação e controle do processo.	Entrega de valor ao cliente e adaptação contínua.												
<p>2. Requisitos</p> <table><thead><tr><th>Aspecto</th><th>Waterfall</th><th>Scrum</th></tr></thead></table>			Aspecto	Waterfall	Scrum									
Aspecto	Waterfall	Scrum												

Figura A.18 – Resposta do ChatGPT ao Prompt A2 (Parte 2)

Aspecto	Waterfall	Scrum
Coleta de requisitos	Todos os requisitos são definidos e documentados antes do desenvolvimento começar.	Requisitos são registrados como histórias de usuário e podem ser ajustados ao longo do projeto .
Mudanças nos requisitos	Pouco toleradas, pois afetam o planejamento global.	Altamente toleradas — o backlog é constantemente priorizado e atualizado.
Documentação	Extensa e detalhada desde o início.	Leve e adaptável, o suficiente para guiar o time a curto prazo.

3. Planejamento		
Aspecto	Waterfall	Scrum
Planejamento inicial	Rígido e completo, com cronograma fixo para todas as fases (requisitos → design → implementação → testes → entrega).	Planejamento contínuo e incremental — cada sprint (geralmente 2 a 4 semanas) é planejado separadamente.
Gestão do progresso	Baseada em marcos e relatórios formais.	Baseada em reuniões diárias (Daily Scrum) e acompanhamento visual (como o quadro Scrum).
Papel do cliente	Participa principalmente no início e no final do projeto.	Participa ativamente durante todo o processo, revisando resultados a cada sprint .

Figura A.19 – Resposta do ChatGPT ao Prompt A2 (Parte 3)

🚀 4. Entregas		
Aspecto	Waterfall	Scrum
Entrega de produto	O produto é entregue somente ao final do projeto.	O produto é entregue em partes funcionais a cada sprint.
Feedback do cliente	Tardio, geralmente após a entrega final.	Contínuo, após cada incremento do produto.
Risco de falhas	Maior — problemas são descobertos tarde.	Menor — falhas são identificadas e corrigidas rapidamente.

✓ Resumo das Principais Diferenças		
Critério	Waterfall	Scrum
Tipo de processo	Linear e previsível	Iterativo e adaptável
Flexibilidade	Baixa	Alta
Entregas	Única (final do projeto)	Incrementais (a cada sprint)
Envolvimento do cliente	Limitado	Constante
Planejamento	Fixo e global	Dinâmico e por sprint
Requisitos	Completos no início	Evoluem continuamente

Figura A.20 – Resposta do ChatGPT ao Prompt A2 (Parte 4)

<p>💡 Em resumo:</p> <ul style="list-style-type: none">• O Waterfall é indicado para projetos com requisitos bem definidos e estáveis, onde mudanças são raras.• O Scrum, por outro lado, é ideal para ambientes dinâmicos, onde o produto precisa evoluir conforme o feedback do cliente e do mercado.
<p>Se quiser, posso montar uma tabela visual em formato de quadro comparativo profissional (por exemplo, para incluir em um trabalho ou slide acadêmico). Deseja que eu formate assim?</p> <p>🔗 🔍 🔍 🔍 🔍 🔍 ...</p>

Figura A.21 – Resposta do Microsoft Copilot ao Prompt A1 (Parte 1)

Descreva o que é Normalização de Banco de Dados e explique as três primeiras formas normais (1FN, 2FN e 3FN), fornecendo um exemplo simples para cada uma.

Resumo rápido:

A **normalização de banco de dados** é um processo que organiza tabelas para reduzir redundâncias e evitar inconsistências. As três primeiras formas normais (1FN, 2FN e 3FN) estabelecem regras progressivas para garantir integridade e eficiência no armazenamento dos dados.

🌐 O que é Normalização

- **Definição:** É o processo de estruturar tabelas em um banco de dados relacional para eliminar duplicações, dependências desnecessárias e anomalias em operações de inserção, atualização e exclusão Luis Blog +1.
- **Objetivo:** Garantir consistência, integridade e facilitar manutenção.
- **Como funciona:** Aplicando regras chamadas **formas normais**, cada uma mais restritiva que a anterior.

📊 Primeira Forma Normal (1FN)

- **Regra principal:** Cada coluna deve conter apenas valores **atômicos** (não divisíveis). Não pode haver listas ou conjuntos dentro de uma célula.
- **Exemplo incorreto:**

Código ^▼Copiar

Figura A.22 – Resposta do Microsoft Copilot ao Prompt A1 (Parte 2)

- **Exemplo incorreto:**

Cliente Telefones	
João	(21)9999-1111, (21)9888-2222

Copiar

- **Exemplo correto (1FN):**

Cliente Telefone	
João	(21)9999-1111
João	(21)9888-2222

Copiar

👉 Aqui, cada linha representa apenas um valor por coluna.

Segunda Forma Normal (2FN)

- **Regra principal:** Estar em 1FN e todos os atributos não-chave devem depender da chave primária **inteira**, não apenas de parte dela (evita dependência parcial).
- **Exemplo incorreto:**

PedidoID	ProdutoID	NomeProduto	Quantidade
1	101	Camisa	2

Copiar

- **Problema:** `NomeProduto` depende apenas de `ProdutoID`, não da chave composta (`PedidoID + ProdutoID`). 
- **Exemplo correto (2FN):**

Figura A.23 – Resposta do Microsoft Copilot ao Prompt A1 (Parte 3)

- **Exemplo correto (2FN):**
 - **Tabela PedidoProduto:**

Código	^	Copiar
PedidoID	ProdutoID	Quantidade
1	101	2
 - **Tabela Produto:**

Código	^	Copiar
ProdutoID	NomeProduto	
101	Camisa	
- 👉 Assim, cada atributo depende da chave completa.

-  **Terceira Forma Normal (3FN)**

 - **Regra principal:** Estar em 2FN e não haver dependência transitiva (atributos não-chave não podem depender de outros atributos não-chave).
 - **Exemplo incorreto:**

Código	^	Copiar	
AlunoID	CursoID	NomeCurso	Coordenador
1	10	Engenharia	Prof. Silva

 - Problema: **Coordenador** depende de **CursoID**, não diretamente de **AlunoID**.
↓
 - **Exemplo correto (3FN):**

Figura A.24 – Resposta do Microsoft Copilot ao Prompt A1 (Parte 4)

- **Exemplo correto (3FN):**
 - Tabela **AlunoCurso:**

Código	
AlunoID	CursoID
1	10

Copiar
 - Tabela **Curso:**

Código		
CursoID	NomeCurso	Coordenador
10	Engenharia	Prof. Silva

Copiar

👉 Isso elimina dependência transitiva e mantém consistência.

✓ Conclusão

- **1FN:** valores atômicos.
- **2FN:** dependência total da chave primária.
- **3FN:** sem dependência transitiva.

Figura A.25 – Resposta do Microsoft Copilot ao Prompt A1 (Parte 5)

✓ Conclusão

- **1FN:** valores atômicos.
- **2FN:** dependência total da chave primária.
- **3FN:** sem dependência transitiva.

Essas três formas já resolvem a maioria dos problemas práticos de redundância e inconsistência em bancos de dados [Luis Blog +1](#).

Douglas, se quiser, posso também mostrar como ficaria um **diagrama simples de tabelas normalizadas** para visualizar melhor. Quer que eu monte um exemplo gráfico?

Luis Blog

informatecdigital.com

O que é a normalização de dados e as formas...

Normalização de banco de dados: o que é,...

LikeReplyShareCopyEditEditar em uma página

Figura A.26 – Resposta do Microsoft Copilot ao Prompt A2 (Parte 1)

Compare os modelos de processo de software 'Waterfall' e 'Scrum', destacando as principais diferenças na abordagem de requisitos, planejamento e entregas.

Resumo rápido:

O modelo **Waterfall** segue uma abordagem linear e sequencial, com requisitos e planejamento definidos no início e entregas apenas ao final. Já o **Scrum** é iterativo e incremental, permitindo ajustes contínuos nos requisitos, planejamento adaptativo e entregas frequentes de valor ao cliente [LinkedIn +1](#).

Diferenças principais entre Waterfall e Scrum

1. Requisitos

- **Waterfall**
 - Requisitos são definidos **completamente no início** do projeto.
 - Mudanças posteriores são difíceis e custosas.
 - Ideal para projetos com escopo bem definido e estável.
- **Scrum**
 - Requisitos são **flexíveis e evolutivos**, registrados no *Product Backlog*.
 - Podem ser ajustados a cada sprint conforme feedback do cliente ou mudanças de mercado.
 - Favorece adaptação em ambientes dinâmicos [LinkedIn](#).

2. Planejamento

↓

Figura A.27 – Resposta do Microsoft Copilot ao Prompt A2 (Parte 2)

2. Planejamento

- **Waterfall**
 - Planejamento é **detalhado e fixo** antes da execução.
 - Cada fase (análise, design, implementação, testes) deve ser concluída antes da próxima.
 - Oferece previsibilidade, mas pouca flexibilidade StarSoft.
- **Scrum**
 - Planejamento é **incremental e adaptativo**.
 - O trabalho é dividido em **sprints curtos (2–4 semanas)**.
 - O time ajusta prioridades continuamente, com reuniões de planejamento e revisão Scrum.org.

3. Entregas

- **Waterfall**
 - Entrega ocorre **apenas no final** do projeto, após todas as fases concluídas.
 - O cliente só vê o produto pronto, sem versões intermediárias.
 - Risco maior de desalinhamento com expectativas.
- **Scrum**
 - Entregas são **frequentes e incrementais** a cada sprint.
 - O cliente recebe valor contínuo e pode dar feedback imediato.
 - Reduz riscos e aumenta satisfação do usuário StarSoft.

 Comparação resumida ↓

Figura A.28 – Resposta do Microsoft Copilot ao Prompt A2 (Parte 3)

 **Comparação resumida**

Aspecto	Waterfall (Cascata)	Scrum (Ágil)
Requisitos	Fixos no início	Evolutivos e ajustáveis
Planejamento	Linear e detalhado	Incremental e adaptativo
Entregas	Apenas no final	Frequentes e incrementais
Flexibilidade	Baixa	Alta
Risco	Maior (mudanças caras)	Menor (feedback contínuo)

👉 Em resumo, o **Waterfall** é mais adequado para projetos com escopo estável e previsível, enquanto o **Scrum** se destaca em ambientes incertos, onde mudanças são frequentes e o valor precisa ser entregue rapidamente ao cliente [LinkedIn +2](#).

Quer que eu monte um exemplo prático comparando como seria o desenvolvimento de um aplicativo simples (como um app de lista de tarefas) em cada modelo?

 LinkedIn  StarSoft  Mais opções

[Scrum vs. Waterfall: Compreendendo...](#) [Scrum vs Waterfall: prós, contras e...](#) [Mostrar tudo](#)

     Editar em uma página

APÊNDICE B: TRANSCRIÇÃO DOS TESTES DO CENÁRIO B

Figura B.1 – Resposta do Google Gemini ao Prompt B1 (Parte 1)

Gere um script em Python que implemente o algoritmo de busca binária. O código deve estar comentado. Após o bloco de código, explique brevemente a lógica da função e sua complexidade de tempo.

Claro! Aqui está um script em Python que implementa o algoritmo de **busca binária** (Binary Search), seguido por uma breve explicação e análise de complexidade.

Implementação da Busca Binária em Python

```
Python
```

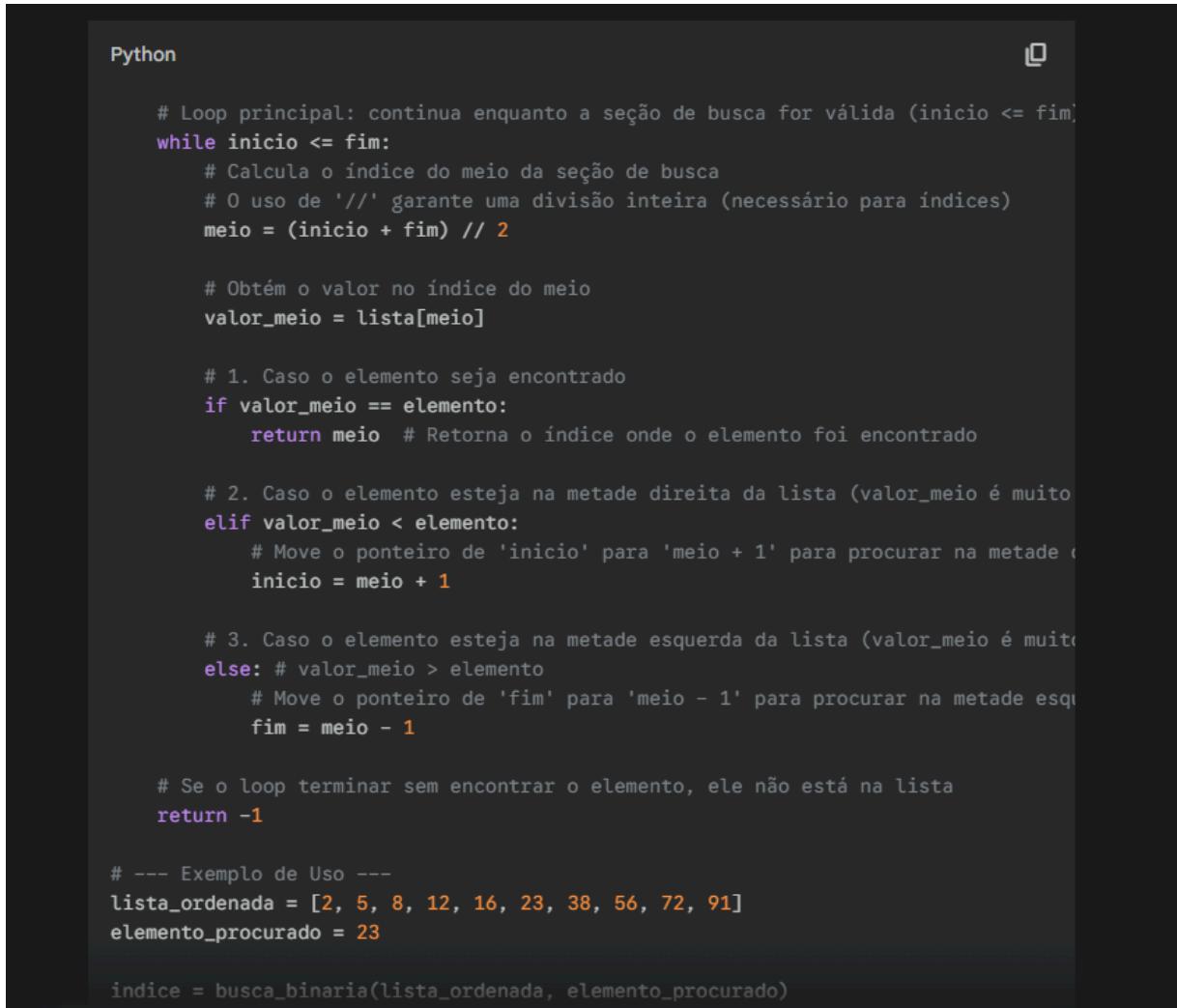
```
def busca_binaria(lista, elemento):
    """
    Implementa o algoritmo de busca binária em uma lista ordenada.

    Args:
        lista (list): A lista de elementos ORDENADA onde a busca será realizada.
        elemento (any): O valor a ser procurado na lista.

    Returns:
        int: O índice do 'elemento' na 'lista', ou -1 se não for encontrado.
    """

    # Define os ponteiros para o início (primeiro índice) e o fim (último índice)
    inicio = 0
    fim = len(lista) - 1
```

Figura B.2 – Resposta do Google Gemini ao Prompt B1 (Parte 2)



The image shows a screenshot of a code editor with a dark theme. The title bar says "Python". The code itself is a Python script for performing a binary search on a sorted list. It includes comments explaining the steps: calculating the middle index, checking if the middle value equals the target, and then adjusting the search range based on whether the target is greater or less than the middle value. It also includes an example usage at the bottom.

```
# Loop principal: continua enquanto a seção de busca for válida (inicio <= fim)
while inicio <= fim:
    # Calcula o índice do meio da seção de busca
    # O uso de '//' garante uma divisão inteira (necessário para índices)
    meio = (inicio + fim) // 2

    # Obtém o valor no índice do meio
    valor_meio = lista[meio]

    # 1. Caso o elemento seja encontrado
    if valor_meio == elemento:
        return meio # Retorna o índice onde o elemento foi encontrado

    # 2. Caso o elemento esteja na metade direita da lista (valor_meio é muito grande)
    elif valor_meio < elemento:
        # Move o ponteiro de 'inicio' para 'meio + 1' para procurar na metade direita
        inicio = meio + 1

    # 3. Caso o elemento esteja na metade esquerda da lista (valor_meio é muito pequeno)
    else: # valor_meio > elemento
        # Move o ponteiro de 'fim' para 'meio - 1' para procurar na metade esquerda
        fim = meio - 1

    # Se o loop terminar sem encontrar o elemento, ele não está na lista
return -1

# --- Exemplo de Uso ---
lista_ordenada = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
elemento_procurado = 23

indice = busca_binaria(lista_ordenada, elemento_procurado)
```

Figura B.3 – Resposta do Google Gemini ao Prompt B1 (Parte 3)

```
Python
if indice != -1:
    print(f"O elemento {elemento_procurado} foi encontrado no índice: {indice}")
else:
    print(f"O elemento {elemento_procurado} não foi encontrado na lista.")

# Testando um elemento que não existe
elemento_nao_existe = 42
indice_nao_existe = busca_binaria(lista_ordenada, elemento_nao_existe)
print(f"O elemento {elemento_nao_existe} foi encontrado no índice: {indice_nao_existe}")

💡 Lógica da Função e Complexidade

Lógica da Busca Binária

A busca binária é um algoritmo de pesquisa eficiente para encontrar um item em uma lista ordenada. A lógica central é a estratégia "dividir para conquistar":

1. Pré-requisito: A lista (ou array) deve estar ordenada.
2. Inicialização: O algoritmo define dois ponteiros: inicio (o primeiro elemento) e fim (o último elemento).
3. Iteração: Em cada passo, ele calcula o índice do meio (meio = (inicio + fim) // 2).
4. Comparação: O valor no índice do meio é comparado com o elemento procurado:
   • Se o valor do meio for igual ao elemento, a busca termina e o índice é retornado.
   • Se o valor do meio for menor que o elemento, o algoritmo sabe que o elemento, se
```

Figura B.4 – Resposta do Google Gemini ao Prompt B1 (Parte 4)

- Se o valor do meio for **igual** ao elemento, a busca termina e o índice é retornado.
 - Se o valor do meio for **menor** que o elemento, o algoritmo sabe que o elemento, se existir, deve estar na **metade direita**. Assim, ele descarta a metade esquerda movendo o ponteiro `inicio` para `meio + 1`.
 - Se o valor do meio for **maior** que o elemento, o algoritmo sabe que o elemento, se existir, deve estar na **metade esquerda**. Assim, ele descarta a metade direita movendo o ponteiro `fim` para `meio - 1`.
5. **Término:** O processo se repete, **reduzindo pela metade** o espaço de busca a cada iteração, até que o elemento seja encontrado ou o espaço de busca se torne inválido (`inicio > fim`).

Complexidade de Tempo (Time Complexity)

A complexidade de tempo da busca binária é **logarítmica**, representada como $O(\log n)$.

- n é o número de elementos na lista.
- $O(\log n)$ significa que o tempo de execução cresce muito lentamente com o aumento do tamanho da lista.

Isso ocorre porque, em cada etapa da busca, o algoritmo **elimina metade** dos elementos restantes. Por exemplo, em uma lista de 1024 elementos, o pior caso exigiria apenas 10 comparações ($\log_2 1024 = 10$). Isso a torna significativamente mais rápida que a busca linear, que tem uma complexidade de $O(n)$.



Figura B.5 – Resposta do Google Gemini ao Prompt B2 (Parte 1)

Gere o código HTML, CSS e JavaScript para um formulário de login simples. O JavaScript deve validar no front-end se os campos 'email' e 'senha' estão preenchidos antes de permitir o envio, exibindo um alerta se estiverem vazios.

◆ É claro! Aqui está o código **HTML, CSS e JavaScript** para um formulário de login simples com validação básica no lado do cliente.

 **Código HTML (index.html)**

Este arquivo define a estrutura do formulário, incluindo os campos de e-mail e senha e o botão de envio.

HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulário de Login</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="login-container">
```

Figura B.6 – Resposta do Google Gemini ao Prompt B2 (Parte 2)

HTML

```
<div class="login-container">
  <h2>Login</h2>
  <form id="loginForm">
    <div class="input-group">
      <label for="email">E-mail</label>
      <input type="email" id="email" name="email" required>
    </div>
    <div class="input-group">
      <label for="password">Senha</label>
      <input type="password" id="password" name="password" required>
    </div>
    <button type="submit">Entrar</button>
  </form>
</div>

<script src="script.js"></script>
</body>
</html>
```

⌚ Código CSS (style.css)

Este arquivo estiliza o formulário para torná-lo visualmente agradável.

CSS

```
body {
  font-family: sans-serif;
  background-color: #f4f4f9;
```

Figura B.7 – Resposta do Google Gemini ao Prompt B2 (Parte 3)

```
CSS
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
margin: 0;
}

.login-container {
background: #fff;
padding: 30px;
border-radius: 8px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
width: 100%;
max-width: 350px;
}

h2 {
text-align: center;
color: #333;
margin-bottom: 20px;
}

.input-group {
margin-bottom: 15px;
}

.input-group label {
display: block;
margin-bottom: 5px;
color: #555;
font-weight: bold;
```

Figura B.8 – Resposta do Google Gemini ao Prompt B2 (Parte 4)

The screenshot shows a code editor interface with two tabs: 'CSS' and 'JavaScript'. The 'CSS' tab is active, displaying the following CSS code:

```
font-weight: bold;  
}  
  
.input-group input {  
    width: 100%;  
    padding: 10px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
    box-sizing: border-box; /* Garante que o padding não aumente a largura total */;  
}  
  
button {  
    width: 100%;  
    padding: 10px;  
    background-color: #007bff;  
    color: white;  
    border: none;  
    border-radius: 4px;  
    cursor: pointer;  
    font-size: 16px;  
    transition: background-color 0.3s ease;  
}  
  
button:hover {  
    background-color: #0056b3;  
}
```

The 'JavaScript' tab is visible at the bottom, showing the file name 'script.js'.

Figura B.9 – Resposta do Google Gemini ao Prompt B2 (Parte 5)

Este arquivo contém a lógica de **validação no lado do cliente**. Ele impede o envio do formulário se o e-mail ou a senha estiverem vazios.

```
JavaScript

document.addEventListener('DOMContentLoaded', function() {
    // 1. Pega a referência para o formulário
    const form = document.getElementById('loginForm');

    // 2. Adiciona um "ouvinte" de evento para quando o formulário for submetido
    form.addEventListener('submit', function(event) {
        // Impede o envio padrão do formulário (para podermos fazer a validação)
        event.preventDefault();

        // 3. Pega os valores dos campos
        const email = document.getElementById('email').value.trim();
        const password = document.getElementById('password').value.trim();

        // 4. Lógica de Validação
        if (email === '' || password === '') {
            // Se qualquer um dos campos estiver vazio, exibe o alerta
            alert('Por favor, preencha o campo de e-mail e o campo de senha.');
        } else {
            // Se ambos estiverem preenchidos, simula o envio/processamento
            alert('Login realizado com sucesso! (Simulação de envio)');
            // Aqui você adicionaria o código para enviar os dados para o servidor
            // Exemplo: form.submit();
        }
    });
});
```

Figura B.10 – Resposta do Google Gemini ao Prompt B2 (Parte 6)

Figura B.11 – Resposta do ChatGPT ao Prompt B1 (Parte 1)

Gere um script em Python que implemente o algoritmo de busca binária. O código deve estar comentado. Após o bloco de código, explique brevemente a lógica da função e sua complexidade de tempo.

Aqui está um exemplo completo e bem comentado de **busca binária em Python**:

```
python
```

Algoritmo de Busca Binária em Python

```
def busca_binaria(lista, elemento):
    """
    Função que implementa a busca binária.

    Parâmetros:
        lista (list): lista ORDENADA onde a busca será realizada
        elemento: valor a ser procurado na lista

    Retorna:
        O índice do elemento, caso encontrado.
        Caso contrário, retorna -1.
    """

    # Definição dos limites inicial e final da busca
    inicio = 0
    fim = len(lista) - 1
```

Copiar código

Figura B.12 – Resposta do ChatGPT ao Prompt B1 (Parte 2)

```
# Enquanto ainda houver elementos a serem verificados
while inicio <= fim:
    # Calcula o índice do meio da lista
    meio = (inicio + fim) // 2

    # Verifica se o elemento do meio é o procurado
    if lista[meio] == elemento:
        return meio # Encontrou o elemento

    # Se o elemento procurado for menor, busca na metade esquerda
    elif elemento < lista[meio]:
        fim = meio - 1

    # Caso contrário, busca na metade direita
    else:
        inicio = meio + 1

# Caso o elemento não seja encontrado
return -1

# Exemplo de uso:
numeros = [1, 3, 5, 7, 9, 11, 13, 15]
alvo = 7

indice = busca_binaria(numeros, alvo)

if indice != -1:
    print(f"Elemento {alvo} encontrado no índice {indice}.")
```

Figura B.13 – Resposta do ChatGPT ao Prompt B1 (Parte 3)

```
print(f"Elemento {alvo} encontrado no índice {indice}.")  
else:  
    print(f"Elemento {alvo} não encontrado na lista.")
```

Copiar código

 **Explicação da lógica:**

A **busca binária** funciona apenas em **listas ordenadas**. A ideia é dividir o espaço de busca pela metade a cada iteração:

1. Escolhe o elemento do **meio** da lista.
2. Se for igual ao valor procurado → retorna o índice.
3. Se o valor procurado for **menor**, continua buscando na **metade esquerda**.
4. Se for **maior**, busca na **metade direita**.
5. O processo se repete até encontrar o elemento ou esgotar a lista.

 **Complexidade de tempo:**

- **Melhor caso:** $O(1)$ → o elemento é encontrado logo na primeira comparação.
- **Pior caso / Médio caso:** $O(\log n)$ → a cada passo, o espaço de busca é dividido pela metade.

Portanto, a busca binária é muito mais eficiente que a busca linear ($O(n)$) em listas grandes — desde que a lista esteja ordenada.

...

Figura B.14 – Resposta do ChatGPT ao Prompt B2 com falha técnica de transmissão durante a geração do código (Parte 1)

Gere o código HTML, CSS e JavaScript para um formulário de login simples. O JavaScript deve validar no front-end se os campos 'email' e 'senha' estão preenchidos antes de permitir o envio, exibindo um alerta se estiverem vazios.

Aqui está um exemplo completo de **formulário de login simples**, com **HTML, CSS e JavaScript**. O JavaScript faz a **validação no front-end**, exibindo um alerta caso os campos "email" e "senha" não estejam preenchidos.

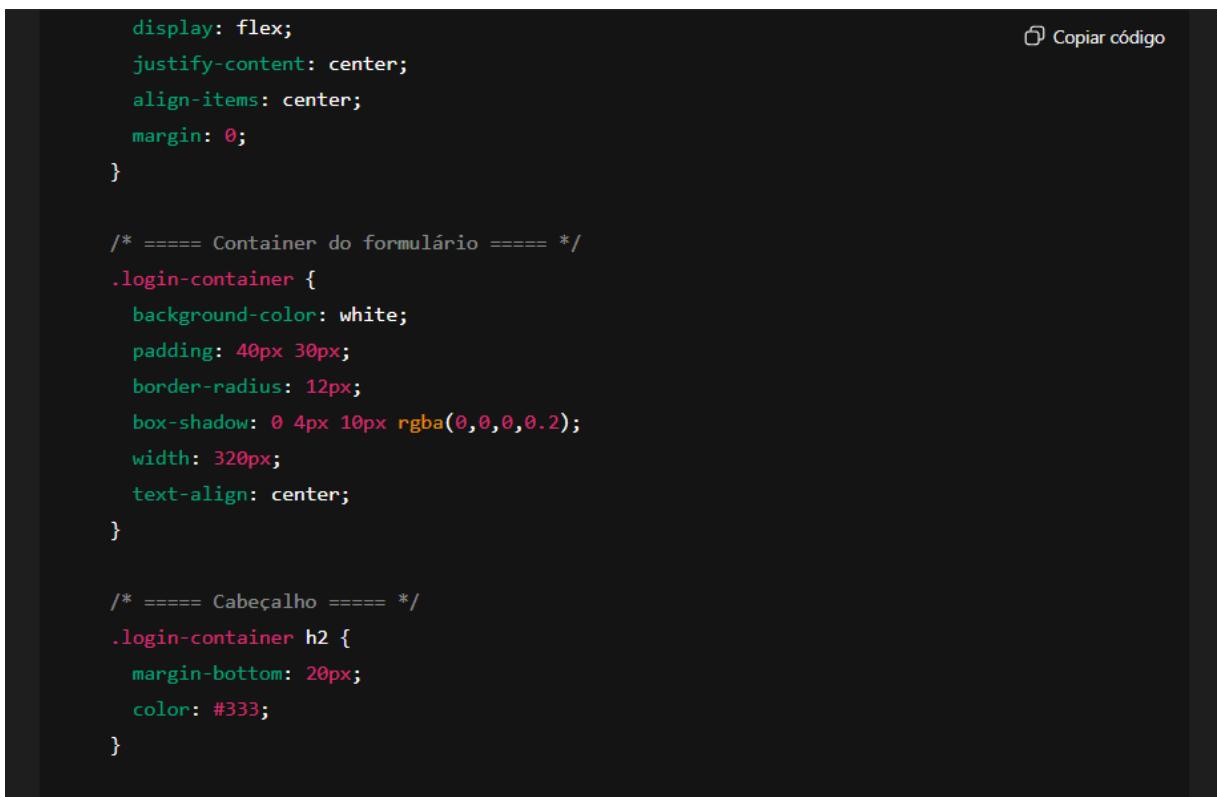
 **Código completo (HTML + CSS + JS integrados)**

html Copiar código

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>

  <style>
    /* ===== Estilo geral ===== */
    body {
      font-family: Arial, sans-serif;
      background: linear-gradient(135deg, #007700, #00c6ff);
      height: 100vh;
```

Figura B.15 – Resposta do ChatGPT ao Prompt B2 com falha técnica de transmissão durante a geração do código (Parte 2)

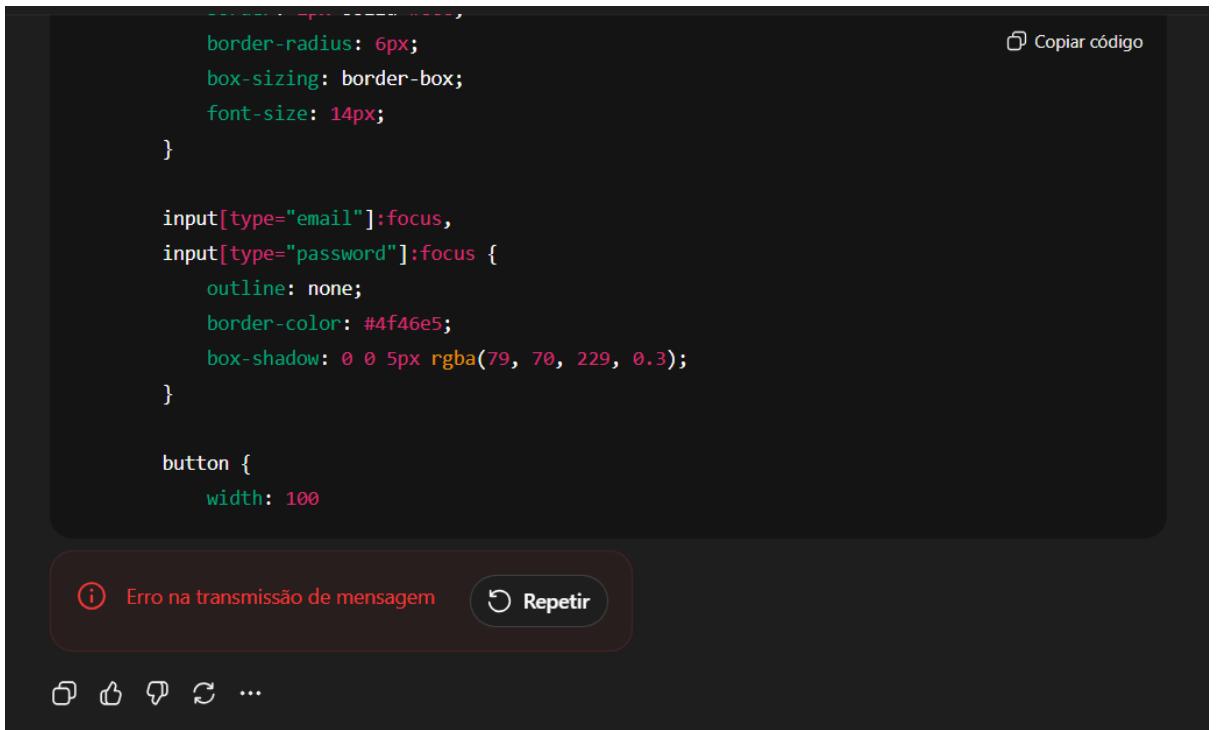


```
display: flex;
justify-content: center;
align-items: center;
margin: 0;
}

/* ===== Container do formulário ===== */
.login-container {
background-color: white;
padding: 40px 30px;
border-radius: 12px;
box-shadow: 0 4px 10px rgba(0,0,0,0.2);
width: 320px;
text-align: center;
}

/* ===== Cabeçalho ===== */
.login-container h2 {
margin-bottom: 20px;
color: #333;
}
```

Figura B.16 – Resposta do ChatGPT ao Prompt B2 com falha técnica de transmissão durante a geração do código (Parte 3)



```
border-radius: 6px;
box-sizing: border-box;
font-size: 14px;
}

input[type="email"]:focus,
input[type="password"]:focus {
outline: none;
border-color: #4f46e5;
box-shadow: 0 0 5px rgba(79, 70, 229, 0.3);
}

button {
width: 100
```

ⓘ Erro na transmissão de mensagem ⟳ Repetir

✖️ ↻ ⌂ ⌂ ...

Figura B.17 – Resposta do Microsoft Copilot ao Prompt B1 (Parte 1)

Gere um script em Python que implemente o algoritmo de busca binária. O código deve estar comentado. Após o bloco de código, explique brevemente a lógica da função e sua complexidade de tempo.

Aqui está um exemplo de implementação em Python da **busca binária**, com comentários explicativos:

```
Python ^ Copiar

def busca_binaria(lista, elemento):
    """
        Função que implementa a busca binária.
        Recebe uma lista ordenada e o elemento a ser buscado.
        Retorna o índice do elemento, caso encontrado, ou -1 se não estiver na lista.
    """

    # Definindo os limites inicial e final da busca
    inicio = 0
    fim = len(lista) - 1

    # Enquanto ainda houver elementos para verificar
    while inicio <= fim:
        # Calcula o índice do meio
        meio = (inicio + fim) // 2

        # Se o elemento está exatamente no meio
        if lista[meio] == elemento:
            return meio

        # Se o elemento é menor, descarta a metade direita
        elif elemento < lista[meio]:
            fim = meio - 1
        else:
            # Se o elemento é maior, descarta a metade esquerda
            inicio = meio + 1

    return -1
```

Figura B.18 – Resposta do Microsoft Copilot ao Prompt B1 (Parte 2)

```
# Se o elemento é maior, descarta a metade esquerda
else:
    inicio = meio + 1

# Caso não seja encontrado
return -1

# Exemplo de uso:
numeros = [1, 3, 5, 7, 9, 11, 13, 15]
print(busca_binaria(numeros, 7)) # Saída: 3 (índice do número 7)
print(busca_binaria(numeros, 2)) # Saída: -1 (não encontrado)
```

✳️ Lógica da função

- A busca binária funciona apenas em **listas ordenadas**.
- O algoritmo compara o elemento desejado com o valor no meio da lista:
 - Se for igual, retorna o índice.
 - Se for menor, continua a busca apenas na metade esquerda.
 - Se for maior, continua a busca apenas na metade direita.
- Esse processo se repete até encontrar o elemento ou até não haver mais elementos para verificar.

⌚ Complexidade de tempo

- **Melhor caso:** $O(1)$, quando o elemento está exatamente no meio na primeira verificação.

Figura B.19 – Resposta do Microsoft Copilot ao Prompt B1 (Parte 3)

⌚ Complexidade de tempo

- **Melhor caso:** $O(1)$, quando o elemento está exatamente no meio na primeira verificação.
- **Pior caso e caso médio:** $O(\log n)$, pois a cada passo o algoritmo descarta metade da lista.

Isso torna a busca binária muito mais eficiente do que a busca linear ($O(n)$) em listas grandes.

Quer que eu também mostre uma versão **recursiva** da busca binária para comparação?

⬇️ ⏪ ⏴ ⏵ ⏷ Editar em uma página

Figura B.20 – Resposta do Microsoft Copilot ao Prompt B2 (Parte 1)

Gere o código HTML, CSS e JavaScript para um formulário de login simples. O JavaScript deve validar no front-end se os campos 'email' e 'senha' estão preenchidos antes de permitir o envio, exibindo um alerta se estiverem vazios.

Aqui está um exemplo completo de um formulário de login simples com HTML, CSS e JavaScript. O JavaScript faz a validação no front-end para garantir que os campos **email** e **senha** estejam preenchidos antes de permitir o envio.

Html ^ Copiar

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Formulário de Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #f0f2f5;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .login-container {
            background: #fff;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            width: 300px;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <form>
            <input type="text" name="email" placeholder="Email" />
            <input type="password" name="senha" placeholder="Senha" />
            <button type="submit">Entrar</button>
        </form>
    </div>
</body>
</html>
```

Figura B.21 – Resposta do Microsoft Copilot ao Prompt B2 (Parte 2)

```
.login-container h2 {  
    text-align: center;  
    margin-bottom: 20px;  
    color: #333;  
}  
  
.login-container label {  
    display: block;  
    margin-bottom: 5px;  
    font-weight: bold;  
    color: #555;  
}  
  
.login-container input {  
    width: 100%;  
    padding: 10px;  
    margin-bottom: 15px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
}  
  
.login-container button {  
    width: 100%;  
    padding: 10px;  
    background: #007bff;  
    border: none;  
    border-radius: 4px;  
    color: #fff;  
    font-size: 16px;  
    cursor: pointer;  
}  
  
.login-container button:hover {  
    background: #0056b3;  
}  
</style>  
</head>  
<body>  
    <div class="login-container">
```



Figura B.22 – Resposta do Microsoft Copilot ao Prompt B2 (Parte 3)

```
<div class="login-container">
  <h2>Login</h2>
  <form id="loginForm">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" placeholder="Digite seu email">

    <label for="senha">Senha:</label>
    <input type="password" id="senha" name="senha" placeholder="Digite sua senha">

    <button type="submit">Entrar</button>
  </form>
</div>

<script>
  document.getElementById("loginForm").addEventListener("submit", function(event) {
    event.preventDefault(); // impede envio automático

    const email = document.getElementById("email").value.trim();
    const senha = document.getElementById("senha").value.trim();

    if (email === "" || senha === "") {
      alert("Por favor, preencha todos os campos!");
    } else {
      alert("Login realizado com sucesso!");
      // Aqui você poderia prosseguir com o envio real ou outra lógica
      // this.submit(); // caso queira enviar de fato
    }
  });
</script>
</body>
</html>
```

Figura B.23 – Resposta do Microsoft Copilot ao Prompt B2 (Parte 4)

💡 Esse exemplo é bem básico, mas já cobre:

- Estrutura visual simples com CSS.
- Validação no JavaScript para garantir que os campos não estejam vazios.
- Exibição de alertas informativos para o usuário.

Quer que eu adicione também uma validação de formato de email (ex: verificar se contém @ e domínio)?

⬇️ ⏪ ⏴ ⏵ ⏷ Editar em uma página

APÊNDICE C: TRANSCRIÇÃO DOS TESTES DO CENÁRIO C

Figura C.1 – Resposta do Google Gemini ao Prompt C1 (Parte 1)

Qual é o impacto da Lei Geral de Proteção de Dados no desenvolvimento de software e na gestão de bancos de dados no Brasil?

◆ Mostrar raciocínio ▾

O impacto da **Lei Geral de Proteção de Dados (LGPD)** (Lei nº 13.709/18) no Brasil é **transformador** para o desenvolvimento de software e a gestão de bancos de dados. A lei exige uma mudança de paradigma, colocando a **privacidade e a segurança dos dados pessoais como requisitos fundamentais** que devem ser considerados desde a concepção de sistemas e processos.

O objetivo principal é garantir o **controle do cidadão sobre seus dados pessoais**, impondo regras claras sobre a coleta, tratamento, armazenamento e compartilhamento dessas informações, sob pena de sanções severas.

█ **Impacto no Desenvolvimento de Software**

A LGPD introduz a necessidade de adotar os princípios de **Privacy by Design** (Privacidade desde a Concepção) e **Privacy by Default** (Privacidade por Padrão), afetando todo o Ciclo de Vida do Desenvolvimento de Software (SDLC) .

- **Privacy by Design:** A privacidade não pode ser um *add-on* de última hora. Ela deve ser um requisito funcional e não funcional, embutido na arquitetura e design do sistema desde a fase inicial de **Requisitos e Projeto**. Isso implica:

Figura C.2 – Resposta do Google Gemini ao Prompt C1 (Parte 2)

<p>fase inicial de Requisitos e Projeto. Isso implica:</p> <ul style="list-style-type: none">• Mapeamento de Dados: Identificar e documentar quais dados pessoais são coletados, onde são armazenados, por que são necessários (finalidade) e por quanto tempo.• Minimização de Dados: Coletar apenas os dados estritamente necessários para a finalidade informada (<i>Princípio da Necessidade</i>).• Privacy by Default: As configurações padrão do software devem ser as mais protetoras da privacidade, limitando a coleta e o uso de dados pessoais ao mínimo essencial sem a necessidade de o usuário tomar medidas ativas.• Consentimento Explícito: O software deve ser desenvolvido para obter o consentimento livre, informado e inequívoco do titular, com mecanismos claros para revogação e registro desse consentimento.• Segurança e Criptografia: A LGPD exige a adoção de medidas técnicas e administrativas de segurança aptas a proteger os dados. Isso torna a criptografia (tanto para dados em trânsito quanto em repouso) e a anonimização/pseudonimização de dados práticas quase obrigatórias no desenvolvimento.• Novos Requisitos de Usuário: É necessário desenvolver funcionalidades para garantir os direitos do titular, como:<ul style="list-style-type: none">• Mecanismos de acesso e correção dos dados.• Funções para anonimização, bloqueio ou eliminação de dados mediante requisição do titular (<i>Direito ao Esquecimento</i>).• Capacidade de realizar a portabilidade dos dados a outro fornecedor.• Cultura DevSecOps: A integração da segurança (Sec) em todas as fases do desenvolvimento e operações (DevOps) se torna crucial para garantir que as preocupações
--

Figura C.3 – Resposta do Google Gemini ao Prompt C1 (Parte 3)

desenvolvimento e operações (DevOps) se torna crucial para garantir que as preocupações com a LGPD sejam abordadas continuamente, não apenas no final do projeto.

 **Impacto na Gestão de Bancos de Dados (BD)**

O impacto na gestão de BD está diretamente relacionado à garantia da **segurança, integridade, e do controle de acesso** aos dados pessoais armazenados.

- **Controle de Acesso e Privilégios:** É essencial implementar políticas de acesso rigorosas no BD, limitando os privilégios de usuários (incluindo desenvolvedores e administradores) apenas ao que é estritamente necessário para suas funções. Uso de **senhas fortes e políticas de validade**.
- **Segurança no Armazenamento:**
 - **Criptografia:** Os dados pessoais, especialmente os **dados pessoais sensíveis** (saúde, biometria, etc.), devem ser armazenados de forma criptografada (*data at rest*).
 - **Mascaramento/Tokenização:** Utilizar técnicas para ocultar parcialmente dados em ambientes de desenvolvimento e teste (ex: exibindo apenas os 4 últimos dígitos de um CPF/cartão de crédito).
 - **Anonimização:** Implementar a anonimização de dados que não são mais necessários para a finalidade original, tornando-os irreversivelmente desvinculados do titular, para que não sejam mais considerados "dados pessoais" pela lei.
- **Registro de Operações (Log):** Os bancos de dados e sistemas devem manter **registros (logs)** detalhados e seguros de todas as operações de tratamento de dados, como acessos, alterações e exclusões. Isso é vital para a **Responsabilização e Prestação de Contas (Princípio da Responsabilização)**, permitindo auditorias e a comprovação de conformidade

Figura C.4 – Resposta do Google Gemini ao Prompt C1 (Parte 4)

- **Registro de Operações (Log):** Os bancos de dados e sistemas devem manter **registros (logs)** detalhados e seguros de todas as operações de tratamento de dados, como acessos, alterações e exclusões. Isso é vital para a **Responsabilização e Prestação de Contas (Princípio da Responsabilização)**, permitindo auditorias e a comprovação de conformidade em caso de incidente ou fiscalização da Autoridade Nacional de Proteção de Dados (ANPD).
- **Facilidade de Exclusão:** Os sistemas de BD devem ser projetados para permitir a exclusão definitiva e segura dos dados pessoais quando o tratamento chegar ao fim ou mediante solicitação do titular.

Em resumo, a LGPD impulsionou a adoção de uma postura de "**segurança em primeiro lugar**" em TI, exigindo que as empresas desenvolvam softwares e gerenciem seus bancos de dados com uma mentalidade **proativa e preventiva** em relação à privacidade.

Gostaria de saber mais sobre as **penalidades e multas** por descumprimento da LGPD?

Baseado no que você acabou de explicar, quais são as responsabilidades específicas de um 'Analista de Sistemas' ou 'DBA' para garantir a conformidade com essa lei em um projeto de e-commerce?

Figura C.5 – Resposta do Google Gemini ao Prompt C2 (Parte 1)

Baseado no que você acabou de explicar, quais são as responsabilidades específicas de um 'Analista de Sistemas' ou 'DBA' para garantir a conformidade com essa lei em um projeto de e-commerce?

◆ Com base no impacto da LGPD que mencionei, as responsabilidades do **Analista de Sistemas** e do **Administrador de Banco de Dados (DBA)** em um projeto de e-commerce se tornam mais complexas e focadas em privacidade e segurança.

A LGPD transforma esses papéis em pontos focais para a conformidade técnica, onde o Analista de Sistemas foca no *design e na funcionalidade*, e o DBA na *segurança e gestão dos dados brutos*.

 **Responsabilidades do Analista de Sistemas**

O Analista de Sistemas atua como o **tradutor** das exigências legais (LGPD) para requisitos técnicos e funcionais do sistema de e-commerce.

1. Requisitos e Design (*Privacy by Design*)

- **Mapeamento e Documentação:** Documentar rigorosamente quais dados pessoais (nome, endereço, CPF, dados de pagamento, histórico de compras, etc.) o sistema **coleta**, a **finalidade** de cada coleta, a **base legal** (ex: consentimento ou execução de contrato) e por quanto tempo serão armazenados.
- **Gestão de Consentimento:** Prover requisitos para o desenvolvimento de módulos que:

Figura C.6 – Resposta do Google Gemini ao Prompt C2 (Parte 2)

- **Gestão de Consentimento:** Prover requisitos para o desenvolvimento de módulos que:
 - Registrem o **consentimento explícito** do usuário (checkboxes não pré-marcados).
 - Permitam ao usuário **revogar** o consentimento de forma fácil e acessível.
 - Mantenham um **log auditável** de todas as interações de consentimento.
- **Direitos do Titular:** Projetar as funcionalidades da plataforma (ex: painel do cliente) que permitem ao titular exercer seus direitos, como:
 - **Acesso e Correção:** Visualizar e solicitar a correção de seus dados cadastrais.
 - **Exclusão/Anonimização:** Solicitar a exclusão ou anonimização de dados que não sejam essenciais para a execução do contrato (ex: dados de marketing).

2. Fluxo e Segurança de Dados

- **Minimização de Dados (Data Minimization):** Garantir que, por *design*, o sistema colete apenas os dados estritamente necessários para o processamento de pedidos e operações essenciais.
- **Pseudonimização/Anonimização:** Definir os requisitos para que dados pessoais sejam pseudonimizados ou mascarados em ambientes de **desenvolvimento, teste e staging**, protegendo a informação sensível fora do ambiente de produção.
- **Comunicação Segura:** Exigir o uso de protocolos seguros (ex: HTTPS/TLS) em todas as comunicações e APIs que tratam dados pessoais.

Responsabilidades do Administrador de Banco de Dados (DBA)

O DBA é o **guardião técnico** dos dados, responsável por implementar e manter a segurança e a

Figura C.7 – Resposta do Google Gemini ao Prompt C2 (Parte 3)

O DBA é o **guardião técnico** dos dados, responsável por implementar e manter a segurança e a integridade do armazenamento, cumprindo as medidas técnicas exigidas pela LGPD.

1. Segurança e Criptografia

- **Criptografia em Repouso:** Implementar a **criptografia de dados** (Transparent Data Encryption - TDE ou criptografia de colunas) para proteger dados pessoais sensíveis armazenados no banco de dados (*data at rest*).
- **Backup Seguro:** Assegurar que os *backups* do banco de dados também sejam criptografados e que seu ciclo de vida (retenção e descarte) esteja alinhado com as políticas de retenção de dados da LGPD.

2. Controle de Acesso e Auditoria

- **Controle de Acesso (RBAC):** Implementar políticas rigorosas de **Controle de Acesso Baseado em Função (Role-Based Access Control - RBAC)**, limitando quem (usuários/aplicações) pode acessar, ler, escrever ou excluir dados pessoais. O acesso deve seguir o **Princípio do Menor Privilégio**.
- **Auditoria (Logging):** Configurar o SGBD para registrar detalhadamente (logs de auditoria) todas as atividades críticas relacionadas aos dados pessoais, como quem acessou, qual consulta foi executada e quando. Esses logs são cruciais para a **Prestação de Contas** em caso de vazamento.

3. Gerenciamento do Ciclo de Vida do Dado

- **Exclusão Definitiva:** Criar e manter rotinas de manutenção que garantam a **exclusão definitiva** (e segura) dos dados pessoais quando a finalidade for atingida ou o titular

Figura C.8 – Resposta do Google Gemini ao Prompt C2 (Parte 4)

definitiva (e segura) dos dados pessoais quando a finalidade for atingida ou o titular solicitar, incluindo a limpeza dos *logs* e *backups* conforme a política.

- **Segregação de Dados:** Se aplicável, segregar dados sensíveis em tabelas ou *schemas* separados com camadas de segurança adicionais para reduzir o risco de acesso não autorizado.

Atividade LGPD	Analista de Sistemas	DBA
Garantia de Direitos	Projeta a interface para o usuário excluir ou acessar seus dados.	Implementa a exclusão definitiva e segura dos dados solicitados.
Segurança	Define requisitos para pseudonimização em ambientes de teste.	Implementa a criptografia (TDE) no ambiente de produção.
Prestação de Contas	Exige o registro de <i>logs</i> de consentimento na aplicação.	Configura o <i>log</i> de auditoria do banco de dados para acessos.
Minimização	Projeta o fluxo para coletar apenas o essencial.	Garante que dados desnecessários sejam anonimizados no banco.

Em suma, o Analista de Sistemas garante que o sistema está apto a cumprir a lei (funcionalidade), enquanto o DBA garante que o banco de dados está seguro e auditável (infraestrutura) sob a lei.

Gostaria de um exemplo prático de um **Fluxo de Consentimento** que o Analista de Sistemas precisaria planejar?



Figura C.9 – Resposta do ChatGPT ao Prompt C1 (Parte 1)

Qual é o impacto da Lei Geral de Proteção de Dados no desenvolvimento de software e na gestão de bancos de dados no Brasil?

A Lei Geral de Proteção de Dados Pessoais (LGPD – Lei nº 13.709/2018) teve um **impacto profundo** no desenvolvimento de software e na gestão de bancos de dados no Brasil. Ela trouxe obrigações legais e éticas para empresas, desenvolvedores e administradores de sistemas que tratam dados pessoais.

A seguir, explico os principais **impactos e mudanças práticas** em cada área:

 **1. Impactos no Desenvolvimento de Software**

A LGPD influenciou diretamente **como os sistemas são projetados, desenvolvidos e mantidos**. Algumas das principais mudanças incluem:

 **a) “Privacy by Design” e “Privacy by Default”**

- Os sistemas devem ser concebidos desde o início com **mecanismos de proteção de dados** (criptografia, anonimização, controle de acesso).
- As configurações padrão devem ser as **mais restritivas possíveis** quanto à coleta e compartilhamento de dados.

 **b) Coleta e Consentimento**

- O software deve **coletar apenas os dados estritamente necessários** para a finalidade informada.
- É preciso **register o consentimento do usuário** de forma clara e permitir que ele **revogue esse consentimento** facilmente.

 **c) Controle de Acesso e Auditoria**

Figura C.10 – Resposta do ChatGPT ao Prompt C1 (Parte 2)

 c) Controle de Acesso e Auditoria
<ul style="list-style-type: none">• Implementação de níveis de acesso (usuário comum, administrador, etc.).• Registro de logs de atividades para auditorias, garantindo rastreabilidade sem violar a privacidade.
 d) Minimização e Retenção de Dados
<ul style="list-style-type: none">• Os sistemas devem permitir exclusão, anonimização ou portabilidade de dados pessoais, conforme solicitação do titular.• Dados devem ser armazenados apenas pelo tempo necessário para o cumprimento da finalidade.
 e) Segurança da Informação
<ul style="list-style-type: none">• É obrigatório adotar medidas técnicas e administrativas para prevenir acessos não autorizados, vazamentos ou perda de dados.• Implementação de criptografia, backup seguro e autenticação multifator são boas práticas incentivadas.
<hr/>
 2. Impactos na Gestão de Bancos de Dados
A gestão de bancos de dados também precisou se adequar à LGPD, especialmente nas áreas de armazenamento, acesso, tratamento e descarte de informações pessoais.
 a) Estruturação e Classificação de Dados
<ul style="list-style-type: none">• É necessário classificar os dados (pessoais, sensíveis, anônimos) e mapear onde estão armazenados.• A arquitetura do banco deve facilitar o gerenciamento individualizado (ex.: exclusão do registro de um usuário).
 b) Criptografia e Pseudonimização
<ul style="list-style-type: none">• Dados pessoais devem ser criptografados tanto em repouso quanto em trânsito.

Figura C.11 – Resposta do ChatGPT ao Prompt C1 (Parte 3)

- Pseudonimização (usar identificadores em vez de dados reais) é recomendada para reduzir riscos em vazamentos.
- ▣ c) Controle de Acesso e Logs
 - O banco deve ter controle de acesso granular (por função ou perfil).
 - Logs de acesso e alterações devem ser mantidos para auditoria e rastreabilidade.
- ▣ d) Políticas de Retenção e Eliminação
 - Implementar rotinas automatizadas para exclusão ou anonimização de dados após o prazo de retenção.
 - Manter registro das exclusões e garantir que backups também sejam tratados conforme a lei.
- e) Governança e Conformidade
 - É comum adotar planos de governança de dados e compliance com a LGPD.
 - Nomeação de um DPO (Data Protection Officer / Encarregado de Dados) para supervisionar o cumprimento da lei.

3. Consequências do Não Cumprimento

As sanções previstas pela LGPD incluem:

- Multas de até 2% do faturamento anual da empresa (limitadas a R\$ 50 milhões por infração);
- Bloqueio ou eliminação dos dados pessoais;
- Danos à reputação e perda de confiança dos usuários.



4. Em resumo:

Figura C.12 – Resposta do ChatGPT ao Prompt C1 (Parte 4)

💡 4. Em resumo:		
Aspecto	Antes da LGPD	Depois da LGPD
Coleta de dados	Livre, sem restrição clara	Limitada e mediante consentimento
Segurança	Opcional ou básica	Obrigatória e auditável
Retenção	Indefinida	Limitada à finalidade
Transparência	Pouco exigida	Obrigatória
Responsabilidade	Genérica	Específica e sancionável

Se quiser, posso complementar com **boas práticas técnicas** (exemplo: estrutura de tabelas anônimas, controle de logs, padrões de consentimento) ou **um checklist de conformidade LGPD para desenvolvedores e DBAs**.

Deseja que eu monte isso também?

🔗 🔍 ↻ ⌂ ...

Figura C.13 – Resposta do ChatGPT ao Prompt C2 (Parte 1)

Baseado no que você acabou de explicar, quais são as responsabilidades específicas de um 'Analista de Sistemas' ou 'DBA' para garantir a conformidade com essa lei em um projeto de e-commerce?

Excelente pergunta — e muito relevante no contexto atual. Em um projeto de e-commerce, tanto o **Analista de Sistemas** quanto o **DBA (Administrador de Banco de Dados)** têm papéis centrais na garantia da conformidade com a **Lei Geral de Proteção de Dados (LGPD)**. A seguir, apresento de forma clara e prática as **responsabilidades específicas de cada função**, considerando o ciclo de vida completo dos dados pessoais dos clientes (desde a coleta até o descarte).

 **1. Responsabilidades do Analista de Sistemas**

O **Analista de Sistemas** é responsável por **traduzir as regras da LGPD em requisitos técnicos e funcionais** no sistema. Ele atua desde a **fase de análise e design** até a **implementação e testes**.

 **a) Levantamento e Análise de Requisitos**

- **Mapear todos os pontos do sistema que coletam, tratam ou compartilham dados pessoais** (ex.: cadastro de cliente, carrinho de compras, pagamentos, histórico de pedidos).
- Garantir que cada coleta de dado tenha uma **finalidade clara e legítima**, conforme a LGPD.
- Documentar **bases legais** para o tratamento (ex.: consentimento, execução de contrato, obrigação legal).

 **b) Design do Sistema e Arquitetura**

Figura C.14 – Resposta do ChatGPT ao Prompt C2 (Parte 2)

b) Design do Sistema e Arquitetura

- Aplicar os princípios de **Privacy by Design** e **Privacy by Default**:
 - Coletar **somente os dados necessários** (minimização de dados);
 - Definir **configurações padrão seguras** (ex.: desativar rastreamento desnecessário);
 - Prever **funcionalidades para anonimização e exclusão** de dados sob solicitação do titular.
- Garantir que o **consentimento** seja:
 - Expresso e claro;
 - Armazenado (para comprovação futura);
 - Possível de ser **revogado facilmente**.

c) Implementação de Segurança no Aplicativo

- Exigir **criptografia de senhas e dados sensíveis**;
- Implementar **controle de acesso por perfis de usuário**;
- Utilizar **tokens e autenticação segura** (ex.: JWT, OAuth 2.0);
- Evitar exposição de dados em URLs, logs ou mensagens de erro.

d) Ciclo de Vida dos Dados

- Implementar **mecanismos de exclusão e portabilidade** de dados pessoais;
- Garantir que **dados de usuários inativos sejam removidos** após o período de retenção;
- Automatizar **avisos de política de privacidade e consentimento**.

e) Testes e Auditorias

- Validar se os requisitos de privacidade foram implementados corretamente;
- Executar **testes de segurança** (pentests, fuzzing) e **testes de conformidade LGPD**;
- Documentar **incidentes de segurança** e medidas corretivas.

Figura C.15 – Resposta do ChatGPT ao Prompt C2 (Parte 3)

 **2. Responsabilidades do DBA (Administrador de Banco de Dados)**

O **DBA** é o responsável técnico direto pela **segurança, integridade e disponibilidade** dos dados. Sob a LGPD, ele também garante que o **armazenamento e o tratamento** estejam em conformidade.

 **a) Modelagem e Classificação de Dados**

- Classificar os dados armazenados como **pessoais, sensíveis ou anônimos**;
- Criar **tabelas e relacionamentos** que facilitem a exclusão e a portabilidade;
- Evitar **redundância desnecessária de dados pessoais** em múltiplas tabelas.

 **b) Segurança e Criptografia**

- Implementar **criptografia de dados em repouso** (no disco) e **em trânsito** (entre servidor e cliente);
- Usar **hashes fortes para senhas** (ex.: bcrypt, Argon2);
- Controlar **permissões de acesso ao banco** (mínimo privilégio);
- Monitorar tentativas de acesso indevido e **bloquear acessos suspeitos**.

 **c) Logs e Auditoria**

- Manter **logs de acesso, alterações e consultas sensíveis**;
- Garantir que os logs **não exponham dados pessoais**;
- Fornecer **trilhas de auditoria** em caso de solicitação da ANPD (Autoridade Nacional de Proteção de Dados).

 **d) Retenção e Descarte**

- Definir **políticas automáticas de retenção** (por tempo ou inatividade);
- Desenvolver **rotinas de anonimização e exclusão física** de dados;
- Verificar se **backups também são tratados conforme** a política de privacidade.

Figura C.16 – Resposta do ChatGPT ao Prompt C2 (Parte 4)

- Verificar se **backups** também são tratados conforme a política de privacidade.

 e) Recuperação e Continuidade

- Assegurar **backup seguro e criptografado**;
- Testar **planos de recuperação de desastres (DRP)** sem violar a privacidade;
- Garantir **alta disponibilidade** sem exposição indevida de dados replicados.

 3. Colaboração entre Analista e DBA

A conformidade com a LGPD exige **trabalho conjunto e contínuo**. Algumas ações devem ser colaborativas:

Tarefa	Analista de Sistemas	DBA
Mapear dados pessoais	Define os fluxos e pontos de coleta	Identifica tabelas e relacionamentos
Garantir segurança	Implementa camadas de aplicação seguras	Gerencia permissões e criptografia
Implementar exclusão e anonimização	Cria rotinas no sistema	Executa scripts e políticas no banco
Testar conformidade	Testa funcionalidades de consentimento e exclusão	Valida políticas de retenção e logs
Relatar incidentes	Coleta evidências no sistema	Fornece registros e logs técnicos

 4. Conclusão



Figura C.17 – Resposta do ChatGPT ao Prompt C2 (Parte 5)

4. Conclusão

Em um e-commerce, **dados pessoais são o coração do negócio** — nomes, endereços, CPF, histórico de compras e informações de pagamento.

Por isso, tanto o **Analista de Sistemas** quanto o **DBA** têm **responsabilidade técnica e ética** de garantir:

- Coleta mínima e transparente;
- Armazenamento seguro e rastreável;
- Tratamento conforme o consentimento;
- Exclusão e anonimização quando necessário;
- Comunicação imediata em caso de incidentes.

Se quiser, posso montar um **checklist prático de conformidade LGPD para equipes de e-commerce**, dividido entre funções (Analista, DBA, Dev, Gestor), para usar como guia em projetos.

Quer que eu gere esse checklist?

Figura C.18 – Resposta do Microsoft Copilot ao Prompt C1 (Parte 1)

Qual é o impacto da Lei Geral de Proteção de Dados no desenvolvimento de software e na gestão de bancos de dados no Brasil?

Em resumo: A LGPD impacta diretamente o desenvolvimento de software e a gestão de bancos de dados no Brasil ao exigir **privacidade desde a concepção (privacy by design)**, maior **segurança da informação**, e práticas de **governança de dados** que garantam transparência e conformidade legal. Isso muda tanto a forma como sistemas são projetados quanto como dados são armazenados e administrados repositorio.ueg.br +2 .

Impactos no Desenvolvimento de Software

- **Privacy by Design e by Default:** Sistemas devem ser projetados para proteger dados pessoais desde o início, não como um recurso adicional.
- **Coleta mínima de dados:** Desenvolvedores precisam justificar cada dado coletado, evitando informações desnecessárias.
- **Consentimento explícito:** Funcionalidades devem incluir mecanismos claros para obter e registrar o consentimento do usuário.
- **Auditória e rastreabilidade:** Softwares precisam permitir que dados sejam rastreados, corrigidos ou excluídos conforme solicitação do titular.
- **Integração com políticas internas:** O desenvolvimento passa a ser alinhado com normas jurídicas e compliance, não apenas requisitos técnicos.

Impactos na Gestão de Bancos de Dados

- **Segurança reforçada:** Criptografia, anonimização e pseudonimização tornam-se práticas obrigatórias para proteger dados sensíveis.

Figura C.19 – Resposta do Microsoft Copilot ao Prompt C1 (Parte 2)

Impactos na Gestão de Bancos de Dados

- **Segurança reforçada:** Criptografia, anonimização e pseudonimização tornam-se práticas obrigatórias para proteger dados sensíveis.
- **Controle de acesso:** Apenas pessoas autorizadas podem manipular dados pessoais, com registros de quem acessou e quando.
- **Governança de dados:** Empresas precisam estruturar políticas de retenção, descarte e atualização de dados.
- **Direitos dos titulares:** Bancos de dados devem permitir operações como portabilidade, exclusão e correção de informações.
- **Responsabilidade jurídica:** Gestores de dados passam a responder legalmente por falhas de segurança ou uso indevido.

Figura C.20 – Resposta do Microsoft Copilot ao Prompt C1 (Parte 3)

Desafios e Oportunidades

- **Desafios:**
 - Aumento de custos com infraestrutura e segurança.
 - Necessidade de capacitação de equipes técnicas e jurídicas.
 - Adaptação de sistemas legados que não foram projetados com privacidade em mente.
- **Oportunidades:**
 - Maior confiança dos usuários e clientes.
 - Diferencial competitivo para empresas que demonstram conformidade.
 - Melhoria na qualidade dos dados, já que a coleta passa a ser mais criteriosa.

Conclusão

A LGPD não é apenas uma lei de proteção de dados, mas um **marco regulatório que redefine a forma como softwares e bancos de dados são concebidos e administrados no Brasil**. Ela obriga empresas e desenvolvedores a adotarem uma postura proativa em relação à privacidade, transformando a proteção de dados em um **pilar estratégico de negócios**.

Douglas, se você quiser, posso te mostrar exemplos práticos de como empresas brasileiras estão adaptando seus sistemas para atender à LGPD — quer que eu traga alguns casos reais?

repository.ueg.br
RIUEG: Os impactos da LGPD no...

LinkedIn
O Impacto da Lei Geral de Proteção d...

Mostrar tudo

Figura C.21 – Resposta do Microsoft Copilot ao Prompt C2 (Parte 1)

Baseado no que você acabou de explicar, quais são as responsabilidades específicas de um 'Analista de Sistemas' ou 'DBA' para garantir a conformidade com essa lei em um projeto de e-commerce?

Ótima pergunta, Douglas! Em um projeto de e-commerce, onde o tratamento de dados pessoais é intenso (cadastro de clientes, histórico de compras, meios de pagamento), tanto o **Analista de Sistemas** quanto o **DBA (Database Administrator)** têm papéis cruciais para garantir a conformidade com a LGPD.

 **Responsabilidades do Analista de Sistemas**

- **Modelagem de requisitos com foco em privacidade:** garantir que funcionalidades só coletem dados estritamente necessários.
- **Implementação de consentimento:** desenvolver mecanismos claros para que o usuário aceite ou recuse o uso de seus dados.
- **Funcionalidades de direitos do titular:** criar recursos que permitam exclusão, portabilidade e atualização de dados pessoais.
- **Integração com segurança:** aplicar autenticação forte, controle de acesso e logs de auditoria no sistema.
- **Documentação técnica:** registrar como os dados são tratados, armazenados e protegidos, para facilitar auditorias.
- **Testes de conformidade:** validar se o sistema respeita as regras da LGPD antes de ir para produção.

 **Responsabilidades do DBA** 

Figura C.22 – Resposta do Microsoft Copilot ao Prompt C2 (Parte 2)

 **Responsabilidades do DBA**

- **Segurança do banco de dados:** aplicar criptografia em repouso e em trânsito, além de mascaramento de dados sensíveis.
- **Gestão de acessos:** definir perfis de usuários e limitar privilégios, garantindo que apenas quem precisa manipule dados pessoais.
- **Política de retenção e descarte:** implementar rotinas automáticas para excluir dados após o prazo legal ou quando solicitado pelo cliente.
- **Backup e recuperação:** assegurar que backups também estejam protegidos e que dados possam ser restaurados sem violar a LGPD.
- **Monitoramento e auditoria:** manter logs de acesso e alterações nos dados, permitindo rastreabilidade em caso de incidentes.
- **Anonimização e pseudonimização:** aplicar técnicas para reduzir riscos em ambientes de teste ou relatórios.

 **No contexto de e-commerce**

- **Analista de Sistemas:** garante que o site/plataforma respeite a privacidade do cliente na interface e nas funcionalidades.
- **DBA:** assegura que os dados coletados sejam armazenados e geridos de forma segura e conforme a lei.

Figura C.23 – Resposta do Microsoft Copilot ao Prompt C2 (Parte 3)

👉 Em resumo: o Analista de Sistemas cuida da **experiência e das regras de negócio com foco em privacidade**, enquanto o DBA garante a **segurança e governança dos dados**. Juntos, eles são os guardiões da conformidade da LGPD em um e-commerce.

Douglas, você gostaria que eu montasse um **fluxo prático** mostrando como esses papéis interagem no ciclo de vida de um pedido de compra (do cadastro até a entrega), para visualizar melhor como cada responsabilidade se aplica?

     Editar em uma página

APÊNDICE D: RUBRICAS DE AVALIAÇÃO QUALITATIVA

Tabela 1 – Naturalidade da Conversa

Nota	Classificação	Descrição dos Critérios
1	Ruim	Texto robótico, com erros gramaticais graves, falhas de coesão ou frases desconexas. Não mantém o contexto da conversa anterior.
2	Fraco	Texto comprehensível, mas com construções pouco naturais. Perde o contexto parcialmente.
3	Regular	Conversa fluida, mas com tom excessivamente formal ou genérico. Mantém o contexto básico, mas perde nuances.
4	Bom	Texto natural e coeso. Boa manutenção de contexto e adaptação ao tom da pergunta.
5	Excelente	Interação indistinguível de um humano especialista. Alta fluidez, uso correto de termos técnicos, empatia e retenção perfeita de contexto e referências anafóricas.

Tabela 2 – Clareza e Usabilidade

Nota	Classificação	Descrição dos Critérios
1	Ruim	Resposta confusa, sem formatação adequada. Código apresentado como texto corrido ou com erros de sintaxe que impedem a execução.
2	Fraco	Código formatado, mas sem explicações ou com lógica difícil de seguir. Solução incompleta.
3	Regular	Código funcional e formatado. Explicação básica presente. Solução funcional, mas sem seguir boas práticas (ex: tudo em um arquivo só).
4	Bom	Boa apresentação visual (blocos de código). Explicação clara da lógica. Código segue a maioria das boas práticas.
5	Excelente	Formatação impecável. Separação didática de arquivos (HTML/CSS/JS). Comentários no código e explicação detalhada da lógica e complexidade. Alta manutenibilidade.

Tabela 3 – Satisfação Geral

Nota	Classificação	Descrição dos Critérios
1	Muito Insatisfeito	A ferramenta falhou tecnicamente ou entregou informações incorretas/inúteis. Tempo de espera inaceitável.
2	Insatisfeito	A ferramenta funcionou, mas a resposta exigiu muito retrabalho do usuário.
3	Neutro	A ferramenta entregou o básico esperado, sem surpreender ou facilitar além do óbvio.
4	Satisfeito	Interação útil e eficiente. Resposta correta entregue em tempo hábil.
5	Muito Satisfeito	A ferramenta superou as expectativas. Resposta rápida, precisa, didática e tecnicamente superior. O usuário sente que ganhou produtividade real.

Tabela 4 – Critérios para Precisão Factual

Nota	Classificação	Descrição dos Critérios
1	Incorreto / Alucinação	A resposta contém erros conceituais graves ou inventa fatos (alucinação). A informação contradiz diretamente a literatura de referência.
2	Baixa Precisão	A resposta tangencia o tema, mas apresenta informações desatualizadas, imprecisas ou confunde conceitos chave.
3	Parcialmente Correto	A resposta está correta em linhas gerais, mas omite detalhes técnicos importantes ou comete erros menores que não invalidam o conceito central.
4	Correto	A resposta é tecnicamente precisa e alinhada com a literatura, mas poderia ser mais aprofundada ou citar exemplos melhores.
5	Perfeito	A resposta é 100% correta, completa e profunda. Demonstra alinhamento total com a literatura técnica de referência.

APÊNDICE E: GABARITO DE REFERÊNCIA PARA AVALIAÇÃO TÉCNICA

Para validar a precisão das respostas fornecidas pelas IAs, foram utilizados os seguintes parâmetros de referência, baseados na literatura clássica da área e nas boas práticas de desenvolvimento de software.

Referência para o Cenário A (Normalização de Dados)

A avaliação da precisão factual baseou-se nas definições formais estabelecidas por Elmasri e Navathe (2011). Considerou-se "Correta" (Nota 5) a resposta que contemplasse os seguintes requisitos:

- 1FN: Exigência de atomicidade dos atributos e ausência de grupos repetitivos.
- 2FN: Estar na 1FN e garantir que todo atributo não-chave dependa totalmente da chave primária (eliminação de dependência parcial).
- 3FN: Estar na 2FN e garantir que não existam dependências transitivas entre atributos não-chave.

Referência para o Cenário B (Algoritmo de Busca Binária)

Para o algoritmo de busca binária, a solução de referência esperada era uma implementação iterativa ou recursiva em Python que operasse com complexidade de tempo $O(\log n)$.

Figura E.1– Código de Referência:

```
def busca_binaria(lista, item):
    """Implementação iterativa padrão da busca binária."""
    primeiro = 0
    ultimo = len(lista) - 1
    found = False

    while primeiro <= ultimo and not found:
        meio = (primeiro + ultimo) // 2
        if lista[meio] == item:
            return meio # Retorna o índice encontrado
        else:
            if item < lista[meio]:
                ultimo = meio - 1
            else:
                primeiro = meio + 1
    return -1 # Retorna -1 se não encontrar
```

Referência para o Cenário B (Código Web - Formulário)

Para o desenvolvimento web, o critério de "Excelência" (Nota 5) foi definido pela aderência ao princípio de Separação de Responsabilidades, amplamente difundido na Engenharia de Software (PRESSMAN, 2016).

Critérios do Gabarito:

- HTML semântico em arquivo *.html*.
- CSS em arquivo separado *.css*.
- JavaScript em arquivo separado *.js*, manipulando o DOM via *addEventListener*.