Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Gabriel Antonio dos Santos

Kassio Claudio Souza do Nascimento

Pedro Henrique Posmão Pereira

ESTUDO COMPARATIVO DE DIFERENTES ARTEFATOS DE CÓDIGO FRONT-END GERADOS POR MODELOS DE LINGUAGEM DE GRANDE ESCALA (LLMS)

Americana, SP

2025

Gabriel Antonio Dos Santos
Kassio Claudio Souza do Nascimento
Pedro Henrique Posmão Pereira

ESTUDO COMPARATIVO DE DIFERENTES ARTEFATOS DE CÓDIGO FRONT-END GERADOS POR MODELOS DE LINGUAGEM DE GRANDE ESCALA (LLMS)

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na área de concentração em Desenvolvimento de sistemas.

Orientador: Prof. Me. Rafael Rodrigo Martinati

Este trabalho corresponde à versão final do Trabalho de Conclusão de Curso apresentado por Gabriel Antonio dos Santos, Kassio Claudio Souza do Nascimento e Pedro Henrique Posmão Pereira e orientado pelo Prof. Me. Rafael Rodrigo Martinati

Americana, SP
2025

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana Ministro Ralph Biasi- CEETEPS Dados Internacionais de Catalogação-na-fonte

SANTOS, Gabriel Antonio dos

Estudo comparativo de diferentes artefatos de código front-end gerados por modelos de linguagem de grande escala (LLMS). / Gabriel Antonio dos Santos, Kassio Claudio Souza do Nascimento, Pedro Henrique Posmão Pereira – Americana, 2025.

153f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana Ministro Ralph Biasi – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Rafael Rodrigo Martinati

1. Análise de dados 2. Inovação tecnológica 3. Inteligência artificial. I. SANTOS, Gabriel Antonio dos, II. NASCIMENTO, Kassio Claudio Souza do, III. PEREIRA, Pedro Henrique Posmão IV. MARTINATI, Rafael Rodrigo V. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana Ministro Ralph Biasi

CDU: 681516
13,8375
007.52

Elaborada pelo autor por meio de sistema automático gerador de ficha catalográfica da Fatec de Americana Ministro Ralph Biasi.

Gabriel Antonio dos Santos
Kassio Claudio Souza do Nascimento
Pedro Henrique Posmão Pereira

Estudo Comparativo de Diferentes Artefatos de Código Front-End gerados por Modelos de Linguagem de Grande Escala (LLMs)

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana Ministro Ralph Biasi.
Área de concentração: Análise e Desenvolvimento de Sistemas.

Americana, 28 de junho de 2025.

Banca Examinadora:

Rafael Rodrigo Martinati
Mestre
Fatec Americana "Ministro Ralph Biasi"

Jonas Bodê
Mestre
Fatec Americana "Ministro Ralph Biasi"

Lucas Serafim Parizotto
Especialista
Fatec Americana "Ministro Ralph Biasi"

DEDICATÓRIA

Aos seis semestres de crescimento acadêmico e as amizades feitas pelo caminho.

AGRADECIMENTOS

Somos gratos a Deus, por nossa existência, e às nossas famílias pelo apoio incondicional que nunca nos foi negado.

Deixamos um agradecimento especial ao nosso orientador por seu constante incentivo e motivação, pela dedicação do seu escasso tempo ao nosso projeto de pesquisa.

Também queremos agradecer à Fatec Americana Ministro Ralph Biasi, por nos fornecer toda a estrutura necessária para a realização deste trabalho.

RESUMO

Esta monografia faz um estudo comparativo entre a capacidade de diferentes Modelos de Linguagem de Grande Escala (LLMs) na geração de código em desenvolvimento front-end, com foco nas ferramentas mais utilizadas, conhecidas e parametrizadas dentro do campo: HTML, CSS e JavaScript.

O estudo faz uma contextualização geral sobre a importância da produção e pesquisa contínua das ferramentas de inteligência artificial, por meio de uma abordagem empírica, aplicada e comparativa, além de ressaltar aspectos importantes sobre o impacto desta tecnologia em campos atrelados ao desenvolvimento, como educação e mercado de trabalho.

Foram conduzidos experimentos com quatro modelos amplamente utilizados: GPT, Claude, Gemini e Grok — identificados nesta pesquisa como Modelos A, B, C e D, respectivamente. A escolha metodológica baseou-se na observação direta dos resultados, na aplicação de tarefas práticas em diferentes níveis de complexidade e na comparação sistemática entre os modelos em contextos controlados.

Nessa etapa, os objetivos da metodologia incluíram a avaliação técnica, a identificação de limitações e a definição de métricas quantitativas, além de insights práticos para aplicação profissional. Foram selecionados artefatos de desenvolvimento comumente utilizados na prática, como formulários simples, componentes interativos e um editor de código ao vivo, a fim de testar desde habilidades básicas até capacidades mais sofisticadas dos modelos.

Os resultados indicam que os LLMs demonstram proficiência na resolução de tarefas de baixa e média complexidade, com geração de código funcional e aderente às instruções fornecidas. O Modelo B destacou-se com melhor desempenho em tarefas intermediárias, oferecendo soluções mais completas, ainda que mais extensas. Contudo, todos os modelos apresentaram dificuldades consideráveis na tarefa de maior complexidade, revelando limitações em manipulações avançadas do DOM e na criação de interatividade dinâmica sem auxílio de bibliotecas externas.

Por fim, a análise crítica dos dados aponta que os LLMs têm elevado potencial como ferramentas de assistência ao desenvolvimento, promovendo ganhos em produtividade e acessibilidade. No entanto, persistem desafios importantes, como a necessidade de supervisão humana contínua, a presença de vieses e a limitada capacidade em resolver problemas altamente complexos. Conclui-se, portanto, que os LLMs podem ser integrados de forma vantajosa aos fluxos de trabalho, desde que utilizados com critérios técnicos e éticos rigorosos.

Palavras-chave: Inteligência Artificial Generativa; Desenvolvimento Front-end; LLMs.

ABSTRACT

This monograph presents a comparative study on the capabilities of different Large Language Models (LLMs) in generating front-end development code, focusing on the most widely used, well-known, and parameterized tools in the field: HTML, CSS, and JavaScript.

The study provides a general contextualization of the importance of continuous research and development of artificial intelligence tools through an empirical, applied, and comparative approach. It also highlights the impact of this technology in related areas such as education and the job market.

Experiments were conducted using four widely known models: GPT, Claude, Gemini, and Grok — referred to in this research as Models A, B, C, and D, respectively. The methodological choice was based on direct observation of results, the application of practical tasks at varying levels of complexity, and systematic comparison between models in controlled environments.

At this stage, the methodological objectives included technical evaluation, identification of limitations, and the definition of quantitative metrics, as well as practical insights for professional application. Common development artifacts were selected, such as simple forms, interactive components, and a live code editor, to test both basic and more advanced model capabilities.

The results indicate that LLMs show proficiency in handling low and medium-complexity tasks, generating functional code aligned with the given instructions. Model B stood out in intermediate tasks, providing more complete—though lengthier—solutions. However, all models showed significant difficulties in the most complex task, revealing limitations in advanced DOM manipulation and in creating dynamic interactivity without external libraries.

Finally, a critical analysis of the data suggests that LLMs have strong potential as development support tools, promoting gains in productivity and accessibility. Nonetheless, key challenges remain, including the need for continuous human oversight, the presence of biases, and limited problem-solving ability in highly complex scenarios. It is concluded that LLMs can be advantageously integrated into development workflows, provided they are used with rigorous technical and ethical standards.

Keywords: *Generative Artificial Intelligence; Front-end Development; LLMs.*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

A Inteligência Artificial Generalizada (IAG) se consolida como um dos pilares da tecnologia nos últimos anos, impactando profundamente diversos campos da sociedade. Entre os avanços mais expressivos estão os Modelos de Linguagem de Grande Escala (LLMs – Large Language Models), que possibilitam a geração automatizada de arquivos e códigos por meio da cognição e criação de linguagem computadorizada. No contexto do desenvolvimento web, esses modelos vêm sendo utilizados de maneira crescente como assistentes de codificação, contribuindo para a aceleração de tarefas, redução de erros e melhoria na experiência dos programadores.

Porém, tais avanços trazem questionamentos técnicos e acadêmicos sobre a real viabilidade do uso dessas tecnologias no setor de desenvolvimento de sistemas web, especialmente em relação à qualidade do código gerado, à aderência a padrões de projeto e à conformidade com boas práticas da engenharia de software. Além disso, surgem preocupações quanto aos limites dessas ferramentas frente a desafios mais complexos, como a criação de interfaces interativas ou a inclusão de componentes avançados.

Portanto, os crescentes avanços no macrocampo da inteligência artificial (AI) trazem consigo um referencial prospectivo de possibilidades e paradigmas, em especial os modelos de linguagem generativos, que, por conta de sua natureza tecnicamente acessível, por trabalharem em um alto nível de linguagem, possibilitam a automatização de tarefas de codificação repetitivas de forma dinâmica.

1.2 OBJETIVO GERAL

O objetivo geral desta pesquisa é avaliar criticamente o desempenho de diferentes LLMs na geração de código para aplicações web, considerando cenários de complexidade variada. A proposta busca não apenas verificar a funcionalidade dos códigos gerados, mas também analisar aspectos como legibilidade, estruturação lógica, adequação a padrões técnicos e aplicabilidade prática no contexto profissional.

1.3 OBJETIVOS ESPECÍFICOS

Foram empregados os seguintes objetivos específicos: expor uma contextualização desde os fundamentos da IA, bem como a importância de seu estudo, abordando pontos centrais como: origem da tecnologia, sua história, e o envolvimento dos modelos de grande linguagem com o setor de desenvolvimento web. Para mais, a definição de LLMs, os desafios inerentes da utilização da ferramenta para finalidade de codificação, perspectivas futuras de uso, padrões de design e outros conceitos importantes para a condução deste estudo. Por fim, realizar uma experimentação comparativa entre IAs, que resulta em uma coleta de dados para a formatação.

1.4 DEFINIÇÃO DA ESTRUTURA GERAL DESTE ESTUDO

Inicia-se da ideia de que uma pesquisa aprofundada sobre a utilização da IA na codificação voltada para a web, pode gerar resultados úteis tanto para a comunidade de desenvolvedores quanto para o público geral. Fundamentado nessa perspectiva, surge o interesse de conduzir um estudo comparativo entre diferentes modelos de linguagem mais amplamente disponibilizados, a fim de gerar resultados fidedignos que derivam de uma comparação sólida.

Foi descrito inicialmente a relevância da temática e posteriormente conceitos importantes para o entendimento desta análise comparativa.

Deste modo, para garantir a condução deste estudo de forma alinhada com princípios científicos bem fundamentados, foi utilizada uma abordagem empírica, que consistiu principalmente na experimentação e análise prática entre determinados modelos de linguagem de grande escala, com foco nos resultados obtidos por meio de uma coleta de dados rigorosamente controlada a partir de uma metodologia científica empírica, aplicada e comparativa.

O trabalho foi estruturado em seis capítulos, sendo que o primeiro apresenta a introdução; o segundo foca em uma contextualização mais abrangente a respeito da relevância de produções afins; o terceiro apresenta o referencial teórico; o quarto explicita a metodologia; o quinto aborda os resultados e o sexto corresponde à conclusão.

Ao final, espera-se oferecer contribuições relevantes tanto para o público geral quanto para o meio acadêmico e profissionais da área, especialmente no uso eficiente e fundamentado da Inteligência Artificial no desenvolvimento web.

2 IMPORTÂNCIA DO TEMA

A revolução tecnológica das últimas décadas transformou radicalmente diversos setores da sociedade. Desde a concepção do teste de Turing na década de 1950, a IA avançou significativamente, com o surgimento de redes neurais e o aprendizado profundo (deep learning), que impulsionaram o reconhecimento de voz, a visão computacional e o processamento de linguagem natural.

O estudo contínuo da inteligência artificial é apontado como essencial desde sua concepção matemática, justamente por conta de sua natureza dinâmica e, por vezes, imprevisível. "Most of the programs which we can put into the machine will result in its doing something that we cannot make sense (if at all), or which we regard as completely random behaviour" (TURING, 1950, p.456). Isso reforça a necessidade de investigações contínuas sobre a forma como sistemas inteligentes operam, aprendem e produzem.

Além disso, como destaca o cientista da computação Michael Irwin Jordan, o estudo da inteligência artificial é de extrema importância para além do propósito de replicar a inteligência humana. Ele defende que é necessário ampliar a compreensão do termo para incluir o Machine Learning (ML), que apesar de ser um dos pilares fundamentais da Inteligência Artificial, não representa a totalidade de sua compreensão. Jordan ressalta que o foco deve estar em aprimorar a capacidade humana e em criar ambientes mais seguros e interessantes através de uma "teia de computação, dados e entidades físicas". A cada dia os modelos de linguagem se tornam mais presentes e impactantes nos mais diversos setores da sociedade, como: educação, mercado de trabalho e ferramentas de desenvolvimento, causando um impacto frontal na vida das pessoas.

2.1 PANORAMA ACERCA DOS IMPACTOS DA IA

2.1.1 PERCEPÇÕES NA EDUCAÇÃO

No contexto educacional, a IA tem se mostrado uma ferramenta poderosa, oferecendo aplicações que vão desde a personalização do aprendizado até a automação de tarefas administrativas. Isso inclui o uso de sistemas de tutoria inteligente, assistentes virtuais, chatbots educacionais, plataformas de aprendizado adaptativo, análise de dados educacionais e ferramentas de avaliação automatizada (COSTA JÚNIOR et al., 2024). A capacidade de resolver problemas, pensar criticamente, colaborar eficazmente e adaptar-se a novas situações são habilidades essenciais para os indivíduos no século XXI (COSTA JÚNIOR et al., 2024), e a IA pode desempenhar um papel crucial no desenvolvimento dessas competências, oferecendo um aprendizado personalizado e eficiente (COSTA JÚNIOR et al., 2024). Essas transformações no desenvolvimento de competências acadêmicas fornecem uma base para inferir a viabilidade e o potencial da IA como ferramenta de codificação aplicada no desenvolvimento web.

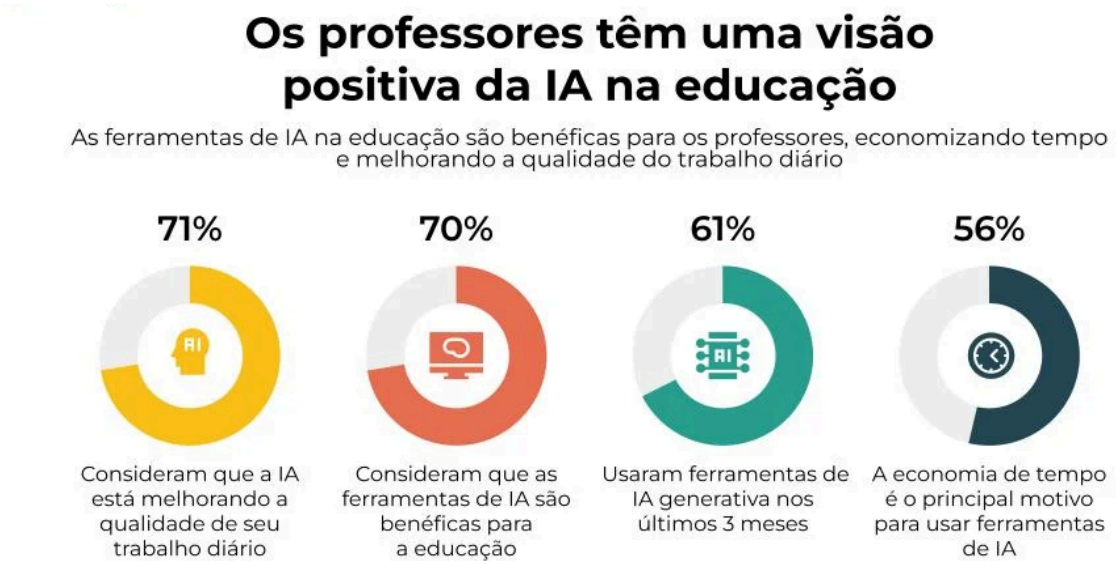
Ferramentas como Code.org e Scratch, que utilizam IA, tornam o ensino de programação mais acessível e eficiente, personalizando o aprendizado para crianças e jovens e adaptando o conteúdo e o ritmo ao progresso individual do aluno (RESNICK et al., 2009). Essas plataformas incentivam a resolução de problemas e o pensamento lógico, que são habilidades cruciais para qualquer profissional técnico, incluindo desenvolvedores web. Essa capacidade da IA de auxiliar no aprendizado da programação e no desenvolvimento do raciocínio crítico (COSTA JÚNIOR et al., 2024) sugere uma transição lógica para a assistência na criação de código em contextos de desenvolvimento web, ao permitir que a IA análise problemas de código, sugira soluções e gere trechos de código com base em requisitos.

Além disso, a IA melhora significativamente a aprendizagem ao fornecer respostas imediatas e personalizadas, adaptando o conteúdo às necessidades individuais dos alunos (COSTA JÚNIOR et al., 2024). Plataformas de aprendizado adaptativo, que utilizam algoritmos de IA, ajustam o nível de dificuldade dos exercícios e identificam áreas de prática (SHUTE; RAHIMI, 2017). Essa personalização pode impulsionar o engajamento e a eficácia da aprendizagem,

oferecendo um ensino mais significativo (COSTA JÚNIOR et al., 2023, p. 1). Essa mesma capacidade de análise e adaptação pode ser aplicada para otimizar o processo de codificação, identificando gargalos, sugerindo melhorias de desempenho e gerando código mais eficiente.

De acordo com uma pesquisa realizada pela empresa Slidesgo, que entrevistou mais de 750 professores e educadores sobre suas percepções da IA na educação, 70,7% dos professores entrevistados consideram as ferramentas de IA benéficas para o setor educacional. Além disso, a empresa também analisou quais são os benefícios da IA na educação conforme as imagens abaixo mostram.

Figura 1 - Visão dos Educadores



Fonte: (Slidesgo)

Figura 2 - Benefícios IA na Educação

Fonte: (Slidesgo)

2.1.2 INOVAÇÃO NO MERCADO DE TRABALHO

As habilidades tecnológicas e digitais são de extrema importância no cenário global e para o mercado de trabalho atual, impulsionado pela transformação digital (BRYNJOLFSSON; MCAFEE, 2014). As competências demandadas incluem, mas não se limitam a, programação, análise de dados, uso de ferramentas digitais e compreensão de sistemas automatizados e de inteligência artificial (IA). A programação é reconhecida como uma habilidade fundamental no mundo digital de hoje. A IA, ao apoiar o ensino dessas habilidades, demonstra intrinsecamente sua capacidade de interagir e processar lógica de codificação.

A automação e a IA estão transformando as profissões, criando novas oportunidades e eliminando tarefas repetitivas e manuais (MANYIKA et al., 2017). Profissionais com habilidades técnicas avançadas são altamente demandados, pois as empresas buscam indivíduos capazes de desenvolver, implementar e gerenciar tecnologias avançadas. No contexto do desenvolvimento web, isso implica que a IA possui o potencial de automatizar porções do processo de codificação, como a geração de código boilerplate, refatoração, e até mesmo testes automatizados. Tal

automação permite que os desenvolvedores concentrem seus esforços em tarefas mais complexas, inovadoras e criativas.

2.1.3 AVANÇOS NO DESENVOLVIMENTO

A inteligência artificial (IA) tem um impacto crescente no desenvolvimento web, a alta acessibilidade dos modelos generativos capazes de gerar códigos, permite que desenvolvedores iniciantes sejam capazes de elaborar sistemas simples com relativa facilidade.

O principal efeito deste impacto reside na automação e eficiência na geração de artefatos de código. Grandes Modelos de Linguagem (LLMs), como os utilizados no projeto “Code2API” desenvolvido pelo professor Zhipeng Gao e seus colegas, demonstram uma capacidade notória para transformar snippets de código de plataformas como Stack Overflow em APIs reutilizáveis.

A automação engloba a identificação de parâmetros de retorno, a geração de novos métodos dentro de uma classe, e a capacidade de se adaptar a diferentes linguagens de programação, como Java e Python. Essa abordagem, cria um novo campo de estudo, baseado na capacidade responsiva de cada modelo realizar suas tarefas de acordo com o comando fornecido, este campo é conhecido como "engenharia de prompts", e em muitos casos elimina a necessidade de regras manuais complexas, tornando o processo de criação de APIs 26 vezes mais rápido do que o feito por humanos, além de extremamente instintivo.

Isso o é de fundamental importância, pois muitos snippets de código na web não são diretamente reutilizáveis ou compiláveis. A IA também é capaz de auxiliar na padronização da reutilização de código e na melhoria da qualidade do código, servindo como uma ferramenta de design de API.

Além disso, a IA demonstrou potencial significativo para a depuração de código, como demonstrado em uma ferramenta anterior desenvolvida pelo pesquisador Zhipeng Xue, um framework chamado “SelfPiCo”, capaz de utilizar LLMs para guiar a execução de código parcial ou incompleto — comum em projetos de software encontrados em repositórios da internet— preenchendo dinamicamente definições ausentes e detectando erros em tempo de execução, como erros de tipo.

Enquanto ferramentas tradicionais de geração de testes unitários exigem código completo e compilável, ferramentas como o “SelfPiCo” já podem operar em código parcial e injetar valores em pontos aleatórios da execução, esse funcionamento permite a descoberta de bugs dependentes de estado e problemas de condição.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 INTELIGÊNCIA ARTIFICIAL

3.1.1 FUNDAMENTOS DA INTELIGÊNCIA ARTIFICIAL

A inteligência Artificial (IA) é um ramo da ciência da computação voltado à realização de tarefas que comumente exige inteligência humana. Dentre as áreas mais exploradas dentro da IA, estão: o Machine Learning (aprendizagem de máquina), o Processamento de Linguagem Natural (NLP) e as Redes Neurais Artificiais, sendo estas últimas responsáveis por simular o funcionamento do cérebro humano com a utilização de estruturas computacionais.

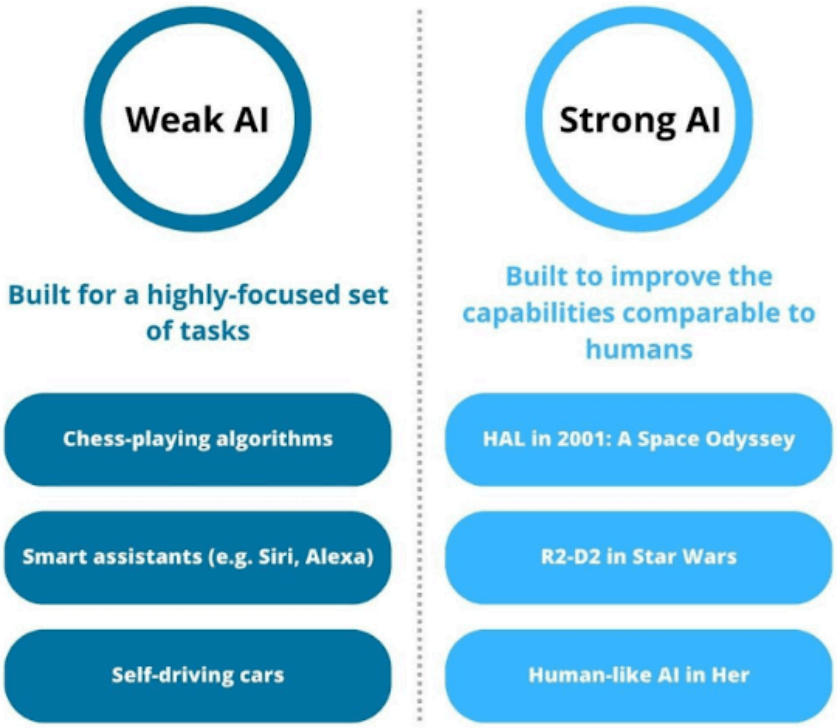
Com o advento dos modelos generativos de conteúdo, a IA passou a ser amplamente discutida, embora seus fundamentos sejam conhecidos há muitas décadas. Ainda na década de 30, o matemático Alan Turing propôs a "Máquina de Turing", um modelo de computação teórico que fundamentou a base dos computadores modernos. Sua visão sobre a solução de problemas através de um sistema baseado em regras e processamento de dados abriu caminho para uma nova perspectiva, onde tomadas de decisão poderiam ser feitas através de lógica computacional. Em 1950 Alan Turing apresentou ao mundo o Teste de Turing, no intuito de prover uma definição operacional satisfatória de inteligência. O teste de Turing consiste em interrogatório, no qual a máquina é aprovada se o interrogador humano não puder fazer diferenciação entre as respostas geradas por IA e as respostas humanas.

Neste teste, critérios como: capacidade de se comunicar naturalmente, razão (armazenar e acessar informação recebida), raciocínio automatizado (chegar a

novas conclusões partindo de informações armazenadas) e aprendizado de máquina são avaliados (RUSSELL; NORVIG, 2016). O teste de Turing continua sendo uma referência importante no estudo da IA moderna, principalmente nos sistemas generativos de linguagem natural. Após, a conferência de Dartmouth em 1956 que o termo Inteligência artificial, cunhado pelo cientista da computação John Mccarthy, passou a ser amplamente usado.

Desde então, a IA se desenvolveu como uma área de estudo próprio e com uma variedade de subcampos, que se estendem desde abordagens educacionais até arte e profundas modificações nas relações de trabalho. Para a definição do campo de estudo deste trabalho, é importante distinguir dois conceitos fundamentais dentro da IA: a IA fraca e a IA forte. A IA fraca refere-se a sistemas projetados para tarefas específicas, como sugerir códigos ou corrigir erros em editores. Já a IA forte envolve a produção de sistemas com consciência e raciocínio comparáveis aos humanos — algo que ainda pertence ao campo especulativo. No contexto atual do desenvolvimento de aplicações web, estamos diante de aplicações de IA fraca, com funcionalidades mais eficazes e especializadas.

Figura 3 - Diferenças entre IA Fraca e IA Forte



Fonte: (Intersog)

3.1.2 EVOLUÇÃO HISTÓRICA DA IA APLICADA À PROGRAMAÇÃO

Os primeiros experimentos com IA ocorreram nas décadas de 1950 e 1960, com sistemas rudimentares voltados à soluções de problemas lógicos e matemáticos. Com a evolução do hardware e o aumento da capacidade de armazenamento e processamento de dados, novos algoritmos foram sendo desenvolvidos, viabilizando aplicações práticas em ambientes produtivos.

Na década de 2000, o crescimento da web impulsionou o surgimento de ferramentas de codificação baseadas em templates e frameworks. Contudo, os avanços mais expressivos ocorreram a partir de 2017 com o desenvolvimento dos Transformers e, em especial, com os modelos generativos baseados em redes neurais profundas, como o GPT (*Generative Pre-trained Transformer*), desenvolvido pela OpenAI.

Ferramentas como GitHub Copilot, Amazon CodeWhisperer e ChatGPT exemplificam a aplicação direta da IA à programação, atuando como assistentes capazes de interpretar o contexto do código e sugerir linhas completas de forma coerente (LEE; QIUFAN, 2021). Essas ferramentas integram-se a ambientes de desenvolvimento integrados (IDEs) e têm sido adotadas amplamente no setor de desenvolvimento web.

Segundo Cuadernos de Educación y Desarrollo (2023), tais sistemas estão redefinindo as competências exigidas dos profissionais da área, exigindo domínio não apenas de linguagens de programação, mas também da capacidade de formular instruções em linguagem natural com clareza e precisão.

3.1.3 PARADIGMAS NA CONSTRUÇÃO DE SISTEMAS COMPUTACIONAIS

O domínio da Inteligência Artificial (IA) é caracterizado por ser uma coleção de padrões, métodos e avanços que, sozinhas ou juntas, solucionam problemas complexos (SICHMAN, 2021, p. 39). Essas tecnologias incluem pesquisa, lógica e representação de conhecimento, mecanismos de decisão, percepção, planejamento, processamento de linguagem natural, tratamento de incertezas e aprendizado de máquina (SICHMAN, 2021, p. 39). Para tal, a IA pode utilizar paradigmas diferentes,

sendo os principais o simbólico, e conexionista, o evolutivo e o probabilístico (SICHMAN, 2021, p. 39).

No paradigma simbólico, o processo inicia-se com a identificação do conhecimento do domínio, seguido pela sua apresentação em uma linguagem formal e pela implementação de um mecanismo de inferência para utilizar esse aprendizado (SICHMAN, 2021, p. 39). O paradigma conexionista, por sua vez, inspira-se no funcionamento do cérebro, usando uma rede de elementos simples, conhecidos como neurônios artificiais, que são capazes de aprender e generalizar a partir de exemplos (SICHMAN, 2021, p. 39). Nesse modelo, o raciocínio é tratado como o aprendizado direto de uma função entrada-saída, por meio de uma técnica matemática de aproximação de funções via regressão não linear (SICHMAN, 2021, p. 39).

Já o paradigma evolutivo emprega um método probabilístico de busca de soluções e otimização de problemas (SICHMAN, 2021, p. 39). Nesse contexto, as soluções são representadas como "indivíduos", e técnicas inspiradas na teoria da evolução – como hereditariedade, mutação, seleção natural e recombinação (ou crossing over) – são aplicadas para selecionar os "indivíduos" mais adaptados para as gerações seguintes, ou seja, aqueles que maximizam uma função objetivo (fitness function) (SICHMAN, 2021, p. 39). Finalmente, o paradigma probabilístico usa modelos para representar o conceito estatístico de independência condicional, com base em relacionamentos causais no domínio (SICHMAN, 2021, p. 40). A inferência nesse paradigma consiste em calcular a distribuição condicional de probabilidades, e em alguns casos de topologia específica, existem algoritmos bastante eficientes para essa tarefa (SICHMAN, 2021, p. 40).

Tabela 1 - Os Quatro Paradigmas da Inteligência Artificial

Paradigma	Descrição Geral	Exemplo de Aplicação	Técnicas Associadas

Simbólico	Baseia-se na representação explícita do conhecimento por meio de regras lógicas e linguagens formais. Usa inferência para resolver problemas.	Sistemas especialistas, diagnóstico médico (MYCIN)	Lógica formal, sistemas especialistas, Prolog
Conexionista	Inspirado no funcionamento do cérebro. Utiliza redes de neurônios artificiais conectados entre si. Aprende a partir de exemplos (entrada/saída).	Reconhecimento de imagens, voz, e escrita manuscrita	Redes neurais artificiais, deep learning
Evolutivo	Baseado em processos de seleção natural. Usa algoritmos genéticos para a otimização de soluções, gerando, cruzando e mutando candidatos.	Otimização de rotas, design de antenas, ajuste de hiperparâmetros em redes neurais	Algoritmos genéticos, programação evolutiva
Probabilístico	Usa modelos estatísticos e de inferência para lidar com incertezas e relações causais. Avalia distribuições de probabilidade condicional.	Previsão de risco, diagnósticos médicos com incerteza, reconhecimento de fala	Redes Bayesianas, inferência estatística, Markov

Fonte: Autoria própria

SICHMAN, Jaime Simão. Inteligência artificial e sociedade: avanços e riscos. *Estudos Avançados*, São Paulo, v. 35, n. 101, p. 37–49, 2021. – CERTO

3.1.4 AUTONOMIA EM SISTEMAS DE INTELIGÊNCIA ARTIFICIAL

O conceito de agente inteligente é crucial para a Inteligência Artificial, sendo definido por Wooldridge (1997 apud JENNINGS, 1999, p.1 apud SICHMAN, 2021, p. 40) como "um sistema computacional encapsulado que está situado em algum ambiente, e que é capaz de ação autônoma e flexível naquele ambiente, a fim de cumprir seus objetivos". Posteriormente, Boissier e Sichman (2004, p.5 apud SICHMAN, 2021, p. 40) ampliaram essa definição, descrevendo um agente como uma "entidade real ou virtual, que é autônoma, pró-ativa, reativa e social, sendo capaz de realizar atividade organizada de modo a atingir seus objetivos, eventualmente interagindo com usuários". Em ambas as definições, o conceito de autonomia é centralizado, sendo essencial para a reflexão sobre os possíveis efeitos, tanto positivos quanto negativos, da interação desses sistemas com os seres humanos (SICHMAN, 2021, p. 40).

A literatura de IA apresenta diversas definições para o termo autonomia, sendo quase todas elas relacionais e associadas a pelo menos quatro significados distintos, conforme inicialmente discutido por Sichman (1995, p.50 apud SICHMAN, 2021, p. 40):

Autonomia em relação ao design: um agente é considerado autônomo se possui sua própria existência, independente da existência de outros agentes, uma abordagem proposta por Demazeau e Müller (1990 apud SICHMAN, 2021, p. 40) e que, anos mais tarde, se materializou na "computação baseada em serviços" (SICHMAN, 2021, p. 40).

Autonomia em relação ao ambiente: um agente autônomo deve ser capaz de operar em ambientes dinâmicos e incertos, onde a percepção pode ser imperfeita e os efeitos de suas ações nem sempre são previsíveis, mesmo que essas ações não sejam controladas pelo próprio agente (SICHMAN, 2021, p. 41). Essa definição é frequentemente encontrada em trabalhos de robótica móvel, como os elencados por Nilsson (1994 apud SICHMAN, 2021, p. 41), e está muito próxima da noção de autopoiese citada por Bourgine (1995 apud SICHMAN, 2021, p. 41).

Autonomia em relação aos próprios objetivos: um agente autônomo é aquele que pode alcançar seus propósitos por conta própria, sem uma necessidade a priori

de cooperar com outros agentes (SICHMAN, 2021, p. 41). Caso decida cooperar, essa escolha se deve a uma possível melhoria em sua atuação (SICHMAN, 2021, p. 41). Essa noção de autonomia foi utilizada por Demazeau e Müller (1990 apud SICHMAN, 2021, p. 41) para classificar comportamentos de agentes de negócio e, posteriormente, refinada por Ferber (1995 apud SICHMAN, 2021, p. 42) para propor a noção de situações de interação entre agentes.

Autonomia em relação às motivações: um agente autônomo possui o livre arbítrio de escolha para interagir socialmente, decidindo cooperar ou não, ou adotar objetivos de outros agentes, com base em seu próprio estado mental (CASTELFRANCHI, 1990 apud SICHMAN, 2021, p. 41). Este significado implica que um agente não é necessariamente benevolente (SICHMAN, 2021, p. 41).

A definição de autonomia em relação às motivações é a que mais gera discussões no contexto das aplicações atuais e futuras da IA, pois implica que um agente não é necessariamente benevolente (SICHMAN, 2021, p. 41). Virginia Dignum (2019, p.18 apud SICHMAN, 2021, p. 41) reitera o caráter multifacetado da autonomia, enfatizando que "autonomia não é uma propriedade intrínseca de um sistema, mas sim o resultado da sua interação com a tarefa, contexto e ambiente", e que ela "deve ser projetado no sistema". Isso significa que nenhum sistema é autônomo em todas as situações ou para todas as tarefas (SICHMAN, 2021, p. 41). A delegação de autonomia, portanto, varia consideravelmente; por exemplo, enquanto a autonomia de um aspirador de pó robótico para decidir qual local limpar primeiro pode ser aceitável, a autonomia de um agente de reserva de viagens para comprar passagens sem confirmação prévia do usuário talvez não fosse adequada (SICHMAN, 2021, p. 41).

A discussão sobre graus de autonomia diferentes que podem ser atribuídos a agentes artificiais é proposta por Falcone e Castelfranchi (2000 apud SICHMAN, 2021, p. 42), fundamentada em métricas de confiança baseadas no histórico de interações anteriores. Essa dinâmica é semelhante à observada na sociedade humana: um docente pode detalhar procedimentos para um orientado inicialmente, mas delegar mais autonomia de planejamento após interações bem-sucedidas (SICHMAN, 2021, p. 42). Um exemplo de autonomia de planejamento em interações

entre agentes inteligentes autônomos pode ser encontrado em Maia e Sichman (2020 apud SICHMAN, 2021, p. 42).

No contexto das interações entre agentes inteligentes e humanos, um grande desafio é incorporar esses graus de autonomia nos chamados Sistemas Sociotécnicos (SST) (SICHMAN, 2021, p. 42). O termo SST foi cunhado por Eric Trist, Ken Bamforth e Fred Emery durante a Segunda Guerra Mundial, a partir de estudos com trabalhadores em minas de carvão inglesas no Instituto Tavistock em Londres (TRIST et al., 2013 apud SICHMAN, 2021, p. 42). A abordagem sociotécnica, segundo Appelbaum (1997 apud SICHMAN, 2021, p. 42), parte da premissa de que as organizações são compostas por elementos sociais e técnicos que trabalham conjuntamente para realizar tarefas organizacionais, gerando tanto produtos físicos quanto resultados sociais/psicológicos (SICHMAN, 2021, p. 42). O foco é permitir que ambos os elementos gerem resultados positivos, ao contrário de métodos convencionais onde as pessoas se adaptam aos elementos técnicos (SICHMAN, 2021, p. 42).

Sistemas sociotécnicos já estão presentes em nossas vidas há pelo menos duas décadas, como evidenciado por experiências com call-centers e serviços bancários (SICHMAN, 2021, p. 42). Atualmente, na maioria dos casos, os elementos técnicos fornecem subsídios para que humanos possam tomar decisões, e existem instâncias para recursos que podem alterar decisões equivocadas, inclusive aplicando sanções aos envolvidos para aprimorar resultados futuros (SICHMAN, 2021, p. 42). Contudo, a inserção da tecnologia de IA nesses sistemas pode alterar essa prática, permitindo que os próprios elementos técnicos tomem algumas decisões (SICHMAN, 2021, p. 42). Essa mudança de paradigma não é inerentemente boa ou ruim, mas exige que tais sistemas incorporem outras propriedades inerentes à interação humana (SICHMAN, 2021, p. 42).

Tabela 2 - Autonomias em IA

Tipo de Autonomia	Descrição	Exemplo de Aplicação
Em relação ao design	O agente tem existência própria, independente de outros agentes. Ex.:	Serviços autônomos em sistemas distribuídos

	Computação baseada em serviços.	
Em relação ao ambiente	O agente atua em ambientes dinâmicos e incertos, com percepção imperfeita e efeitos não totalmente previsíveis.	Robótica móvel, como drones e veículos autônomos
Em relação aos próprios objetivos	O agente é capaz de atingir seus objetivos de forma independente, podendo optar por cooperar com outros agentes.	Agentes de negociação ou sistemas de recomendação inteligentes
Em relação às motivações	O agente decide com base em seu estado mental, podendo escolher adotar ou não objetivos alheios, refletindo sua motivação interna.	Agentes com tomada de decisão ética ou interações sociais complexas

Fonte: Autoria Própria

3.1.5 LARGE LANGUAGE MODELS

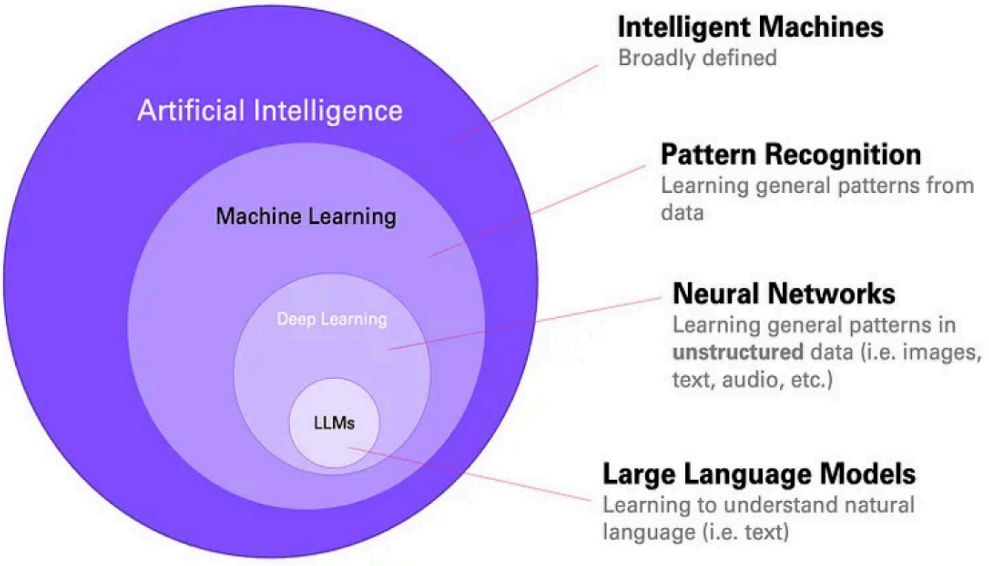
Os Modelos de Linguagem de Grande Escala (LLMs, do inglês *Large Language Models*) figuram entre os avanços mais relevantes da inteligência artificial na contemporaneidade e são fundamentados em redes neurais profundas. Com isso, esses modelos são projetados para processar, interpretar e gerar linguagem natural com um grau elevado de similaridade à produção textual humana (Gallegos et al., 2023).

Ademais, o adjetivo “grande” refere-se à enorme quantidade de parâmetros utilizados em sua estrutura, que pode variar de centenas de milhões a bilhões, como ocorre nos modelos da série GPT-2, cujas versões apresentam entre 124 milhões e 1.558 milhões de parâmetros (Raschka, 2025). Dessa forma, essa escala de

parametrização, aliada ao treinamento sobre vastos dados textuais, permite que os LLMs capturem padrões sutis e complexidades linguísticas com notável eficácia.

Além disso, a arquitetura fundamental que sustenta esses modelos é o Transformer, cuja eficiência reside na segmentação do texto em tokens — unidades mínimas que representam palavras ou subpalavras — e na posterior conversão desses tokens em vetores numéricos denominados *embeddings*, que codificam seu significado em contexto (Alamaar et al., 2024). Segundo o engenheiro de pesquisa Sebastian Raschka no livro *Build a Large Language Mode*l, a aplicação de mecanismos de atenção no Transformer confere ao modelo a função de atribuir pesos distintos aos tokens, conforme sua relevância no contexto global da sequência, otimizando o processo da informação.

Figura 4 - Explicação LLMs



Fonte: (ultralytics)

O treinamento de um LLM geralmente ocorre em duas fases distintas: inicialmente, há uma etapa de pré-treinamento em larga escala com dados não rotulados, seguida por uma fase de ajuste fino (*fine-tuning*) voltada para domínios ou tarefas específicas (Alamaar et al., 2024).Certamente, esse procedimento tem sido aplicado com sucesso em modelos como o GPT-2, que exemplifica o potencial dos LLMs em tarefas de geração textual (Raschka, 2025). De forma geral, os LLMs

podem ser classificados em modelos de representação, como o BERT, voltados para compreensão de linguagem, e modelos generativos, como os da família GPT, orientados à produção de texto (Alamaar et al., 2024).

Portanto, o impacto dos LLMs tem sido expressivo e significativo no campo da Processamento de Linguagem Natural (PLN), permitindo avanços em áreas como tradução automática, sumarização, análise de sentimentos e geração de conteúdo textual (Gallegos et al., 2023). Inclusive, um dos recursos estruturantes dessas aplicações é a utilização de *text embeddings*, que representam o texto em forma numérica, possibilitando operações como a busca semântica baseada em significado contextual (Alamaar et al., 2024). Além do mais, práticas como a engenharia de *prompts* e a integração com ferramentas externas por meio de abordagens como a Geração Aumentada por Recuperação (RAG) vêm expandindo ainda mais as capacidades desses modelos.

Contudo, a disseminação e o uso crescente dos LLMs também suscitam preocupações de ordem ética e ambiental. Logo, um dos principais desafios éticos está relacionado aos vieses sociais incorporados durante o treinamento, uma vez que esses modelos podem reproduzir e até acentuar estereótipos e preconceitos presentes nos dados de origem. Tais tendências, especialmente os de natureza representacional, são particularmente problemáticos e eficazes por perpetuar visões depreciativas ou hierarquizantes sobre determinados grupos sociais (Gallegos et al., 2023). Sob a perspectiva ambiental, destaca-se o alto custo energético atrelado tanto à fase de treinamento quanto à de inferência dos modelos, o que ocasiona implicações significativas para a sustentabilidade na área da computação (Husom et al., 2024).

3.2 DESENVOLVIMENTO WEB

3.2.1 DEFINIÇÃO

O desenvolvimento web, em sua essência, está associado à criação de aplicações que operam no ambiente da internet. Tais aplicações, em geral,

apresentam uma arquitetura dividida em duas camadas principais: o lado do cliente (frontend) e o lado do servidor (backend). Tal como, a camada cliente é executada diretamente no navegador do usuário, enquanto a camada servidor opera em um servidor web, encarregado de processar requisições, gerar conteúdo em HTML e enviá-lo de volta ao navegador, que realiza a renderização da interface (Kaluza et al, 2019).

Diante da crescente complexidade dessas aplicações e da demanda por maior eficiência, tornou-se prática comum a adoção de frameworks no desenvolvimento web. Além disso, esses frameworks, ou estruturas de aplicações web, consistem em conjuntos de ferramentas compostos por APIs, bibliotecas e extensões que oferecem uma base sólida para a construção de sistemas, facilitando o processo e promovendo ganhos substanciais em produtividade (Swacha, J; Kulpa, A, 2023).

Dessa forma, entre os benefícios mais relevantes do uso de frameworks, destacam-se a redução do tempo de desenvolvimento e a padronização do código, que contribui para sua clareza e manutenção. Além disso, tais ferramentas impõem arquiteturas consistentes, o que favorece a organização do projeto e a colaboração entre equipes sendo possível identificar frameworks voltados para as diferentes camadas da aplicação, abrangendo desde soluções para backend até ferramentas específicas para frontend, bem como abordagens full-stack que contemplam ambas as partes (Salas-Zárate et al, 2014).

Ademais, a adoção de boas práticas é outro elemento fundamental no contexto do desenvolvimento web. Com o intuito de, fazer referência a métodos consolidados e documentados, que se mostram eficazes para alcançar resultados desejados de forma consistente.Isto é, elas envolvem desde a validação de formulários — garantindo a integridade e o formato adequado dos dados fornecidos pelos usuários — até o uso de recursos modernos como HTML5, que permite a integração de mídias ricas sem a necessidade de plugins. Outras práticas incluem o emprego de mapeamento objeto-relacional (ORM) para simplificar o acesso a bancos de dados, a implementação de medidas de segurança contra vulnerabilidades, a utilização de chamadas assíncronas via AJAX, a execução de

testes automatizados e o uso de templates para otimizar o desenvolvimento da interface (Salas-Zárate et al, 2014).

Mais recentemente, observou-se a incorporação de modelos de linguagem de larga escala (LLMs) no cotidiano do desenvolvimento web. Tais modelos, utilizados por meio de chatbots inteligentes, têm se mostrado valiosos ao auxiliar programadores na geração de trechos de código, na identificação e correção de erros e na simplificação de tarefas recorrentes. Com isso, ferramentas baseadas em LLMs são empregadas para automatizar a escrita de estruturas HTML e CSS, scripts em JavaScript e até mesmo a integração entre serviços de backend e bases de dados (Smutny et al, 2024).

3.2.2 PADRÕES DE DESIGN

Um padrão de design é amplamente compreendido como uma solução comprovada para problemas recorrentes no desenvolvimento de software. Segundo Stoyan Stefanov em seu livro Javascript Patterns, padrões foram desenvolvidos e refinados ao longo do tempo, baseando-se em práticas que visam melhorar a reutilização de componentes e a flexibilidade dos sistemas. Assim, os padrões de design ajudam a resolver problemas específicos de forma eficiente, refletindo um processo contínuo de adaptação e aprimoramento para garantir soluções mais robustas e reutilizáveis (Stefanov, 2010).

Bem como, a adoção e aplicação de padrões de design no desenvolvimento de software oferecem uma série de benefícios importantes. Por exemplo, um dos principais benefícios é a possibilidade de utilizar um kit de ferramentas com soluções testadas e comprovadas, que facilitam a comunicação entre as equipes de desenvolvimento e aprimoram a clareza no processo de criação do software. Nesse sentido, os padrões oferecem uma linguagem comum e um vocabulário compartilhado entre os desenvolvedores, tornando o processo de documentação e discussão do projeto mais eficiente (Stefanov, 2010). Além disso, esses padrões contribuem para o aprimoramento das habilidades de resolução de problemas, pois eles são baseados em princípios sólidos de design orientado a objetos, ajudando os

desenvolvedores a criar sistemas mais estruturados e organizados (Gamma et al, 1997).

Dessa forma, os padrões de design são tradicionalmente classificados em três grandes categorias: Padrões Criacionais, Padrões Estruturais e Padrões Comportamentais. Os Padrões Criacionais lidam com a criação de objetos, buscando garantir maior flexibilidade e reuso durante a geração de instâncias de classes. Já os Padrões Estruturais descrevem como combinar objetos e classes em estruturas maiores, visando construir novas funcionalidades sem comprometer a coesão do sistema. Por fim, os Padrões Comportamentais se concentram na comunicação e atribuição de responsabilidades entre os objetos, buscando uma melhor distribuição das responsabilidades dentro do sistema (Shvets, 2018; Gamma et al, 1997).

Figura 5 - Tipos de Padrões de Design

Fonte: (Gran)

Também, é importante notar que muitos desses padrões estão diretamente associados aos princípios do design orientado a objetos, como os princípios SOLID (Responsabilidade Única, Aberto-Fechado, Não Se Repita, entre outros), que são aplicados através desses padrões de forma estruturada e eficaz.Assim, os padrões em essência proporcionam maneiras concretas de aplicar esses princípios e melhorar a qualidade do código, evitando a duplicação e tornando os sistemas mais modulares e fáceis de entender (Martin, 2008).

No entanto, é necessário ter cautela ao aplicar padrões de design, pois, embora tragam benefícios significativos, sua utilização excessiva pode resultar em uma complexidade desnecessária. Com isso, a introdução de camadas adicionais de abstração pode dificultar o entendimento e a manutenção do código, além de impactar negativamente o desempenho do sistema. (Hallie et al, 2021).

3.3 APLICAÇÕES ATUAIS DA IA NO DESENVOLVIMENTO WEB

A Inteligência Artificial, especialmente em suas formas, generativa e interativa, tem transformado profundamente o desenvolvimento de aplicações web. Processos antes manuais agora podem ser automatizados, promovendo maior produtividade, acessibilidade e agilidade. Essa mudança permite que desenvolvedores concentrem-se em tarefas mais criativas, ao mesmo tempo em que reduz barreiras de entrada para iniciantes (CUADERNOS DE EDUCACIÓN Y DESARROLLO, 2023).

Entre as principais aplicações da IA nesse contexto estão a geração automatizada de estruturas HTML e estilizações CSS com base em linguagem natural, bem como a criação de scripts JavaScript para interações dinâmicas. Essas funções facilitam a construção de interfaces e tornam o processo de desenvolvimento mais intuitivo, aproximando-o de uma abordagem centrada na linguagem humana (CUADERNOS DE EDUCACIÓN Y DESARROLLO, 2023).

Outra contribuição relevante é a automatização de testes que simulam fluxos de usuários, detectando falhas e garantindo a qualidade do sistema. Além disso, assistentes de IA produzem documentação técnica e comentários de código, o que facilita o entendimento e a manutenção dos projetos, contribuindo para a sustentabilidade do desenvolvimento (CUADERNOS DE EDUCACIÓN Y DESARROLLO, 2023).

Ferramentas inteligentes vêm sendo integradas a ambientes como Visual Studio Code e Android Studio, oferecendo sugestões em tempo real e atuando como tutores personalizados. Isso beneficia especialmente programadores em fase de aprendizado, adaptando-se ao seu ritmo e promovendo uma experiência de

desenvolvimento mais acessível e eficiente (CUADERNOS DE EDUCACIÓN Y DESARROLLO, 2023).

Além de otimizar a produtividade, essas tecnologias ampliam a capacidade criativa dos desenvolvedores. McLuhan (1969) descreve essas transformações como uma “extensão da inteligência humana”, reforçando o papel da IA como parceira na construção digital. Assim, mais do que automação, a IA representa um novo estágio de interação entre humanos e máquinas na criação tecnológica.

3.4 DESAFIOS PARA A IMPLEMENTAÇÃO DA IA COMO FERRAMENTA DE CODIFICAÇÃO NO DESENVOLVIMENTO WEB

Apesar do grande potencial da IA, sua implementação eficaz, seja na sociedade ou como ferramenta de codificação, enfrenta desafios substanciais (COSTA JÚNIOR et al., 2024). As fontes destacam barreiras que se aplicariam igualmente à sua utilização no desenvolvimento web:

Barreiras Tecnológicas e de Infraestrutura: A carência de infraestrutura tecnológica adequada, como equipamentos modernos e conexões de internet de alta velocidade, pode impedir a implementação eficaz de ferramentas baseadas em IA (LUCKIN et al., 2016). Para o desenvolvimento web, isso se traduziria na complexidade de integrar sistemas de IA com plataformas de desenvolvimento existentes e na necessidade de suporte técnico contínuo.

Custos Financeiros: Os altos custos de implementação e manutenção de tecnologias de IA são significativos (MEANS et al., 2013). Isso engloba não apenas a aquisição de hardware e software especializados, mas também a necessidade constante de atualização para acompanhar os rápidos avanços na área, o que pode ser um impedimento para pequenas e médias empresas.

Formação e Adaptação dos Profissionais: Muitos educadores não estão familiarizados com as tecnologias de IA e carecem das habilidades necessárias para utilizá-las de forma eficaz (COSTA JÚNIOR et al., 2024). De forma análoga, desenvolvedores web precisariam de formação contínua e capacitação para integrar

e utilizar as ferramentas de IA de forma produtiva em seus fluxos de trabalho, garantindo que não se sintam sobrecarregados ou resistentes à adoção.

Questões Éticas e de Privacidade: A coleta e o processamento de grandes volumes de dados, incluindo dados de código e projetos, levantam preocupações significativas sobre privacidade e segurança (WILLIAMSON, 2017). É crucial que sejam implementadas políticas rigorosas de proteção de dados para garantir o uso ético das informações. Além disso, a transparência dos algoritmos de IA é fundamental para assegurar que não haja viés ou discriminação (O'NEIL, 2016), exigindo auditoria regular para identificar e mitigar qualquer forma de viés. A implementação desigual dessas ferramentas também pode exacerbar as disparidades existentes no mercado.

3.5 PERSPECTIVAS FUTURAS E IMPACTOS NO SETOR DE DESENVOLVIMENTO WEB

Sempre que uma tecnologia altamente produtiva surge, ela vem acompanhada de profundas mudanças sociais e quebra de paradigma. Um paralelo claro pode ser traçado entre o advento da automação industrial e o crescimento contínuo das tecnologias de Inteligência artificial; funções antes desempenhadas exclusivamente por humanos, agora, podem ser desempenhadas por modelos de linguagem treinados. O pesquisador lusitano Pedro Domingos em seu trabalho de divulgação científica "O Algoritmo Mestre" argumenta que — conflitos dessa ordem não são de homem versus Máquina e sim homem com máquina versus homem sem máquina — implicando diretamente nos desafios de ordem adaptativa da mão de obra e não em sua substituição de maneira evidente. Porém, ao contrário da automação industrial, o avanço da IA não se restringe apenas a trabalhos braçais antes desempenhados por mãos humanas, mas apresenta uma variedade mais ampla de atividades: artísticas, administrativas e até mesmo campos específicos do desenvolvimento web, como a codificação front-end.

Diante de um salto tecnológico tão grande, é natural que tensões semelhantes à era pré-automatização também surjam no setor de desenvolvimento, tais como o desemprego estrutural. No entanto, a inteligência artificial apresenta um caráter

específico: desenvolvimento humano. Segundo Azambuja e Silva (2024), A inteligência artificial é uma inteligência humana, ficamos presos a uma falsa dicotomia e permanecemos incapazes de pensar criativamente as infinitas possibilidades da articulação e interconexão que estas inteligências complexas nos permitem.

A Inteligência artificial, produto das faculdades humanas, tem potencial de expandir as possibilidades de interação e aprendizado dentro de todos os setores de trabalho, uma ferramenta potencializadora no campo do desenvolvimento web e não uma substituta definitiva de seus desenvolvedores. O campo de codificação front-end é caracterizado por uma grande dinamicidade, uma constante evolução de frameworks, padrões de design e bibliotecas, nesse contexto a Inteligência artificial surge como uma ferramenta promissora na otimização de processos, diminuição na complexidade de codificação e representa um alto potencial para diminuir a barreira de entrada de novos profissionais.

4 METODOLOGIA

4.1 ABORDAGEM EMPÍRICA, APLICADA E COMPARATIVA

A presente pesquisa adota uma abordagem empírica, aplicada e comparativa para investigar o desempenho de modelos LLM na geração de código para desenvolvimento web. Esta tríade metodológica foi escolhida por sua adequação ao objeto de estudo e aos objetivos da pesquisa.

A abordagem empírica fundamenta-se na observação direta dos resultados produzidos pelos modelos, permitindo conclusões baseadas em evidências concretas e não apenas em pressupostos teóricos. Segundo Wohlin et al. (2012), estudos empíricos em engenharia de software são essenciais para avaliar novas tecnologias e ferramentas em contextos práticos. Esta abordagem é particularmente relevante no campo emergente da geração automática de código, onde a eficácia das soluções precisa ser verificada através de experimentos controlados.

A natureza aplicada da pesquisa manifesta-se na seleção de tarefas de programação web com relevância prática e aplicabilidade direta no desenvolvimento de software contemporâneo. Como destacam Kitchenham e Charters (2007), pesquisas aplicadas em engenharia de software devem abordar problemas reais enfrentados por desenvolvedores e organizações. Os artefatos selecionados para este estudo representam componentes comumente utilizados em aplicações web modernas, variando em complexidade e requisitos técnicos.

Por fim, a abordagem comparativa permite contrastar o desempenho de diferentes modelos LLM sob condições idênticas, identificando padrões, tendências e limitações específicas. De acordo com Juristo e Moreno (2013), estudos comparativos são fundamentais para estabelecer benchmarks e avaliar o progresso em tecnologias emergentes. Esta abordagem possibilita não apenas determinar qual modelo apresenta melhor desempenho em determinadas tarefas, mas também compreender os fatores que influenciam esse desempenho.

A integração dessas três abordagens metodológicas cria um framework robusto para a avaliação sistemática dos modelos LLM na geração de código,

permitindo conclusões fundamentadas e contribuições significativas para o campo de estudo.

4.2 OBJETIVOS DA METODOLOGIA

A metodologia foi desenvolvida com os seguintes objetivos específicos:

1. Avaliar a capacidade técnica dos modelos LLM na geração de código funcional para desenvolvimento web front-end, utilizando exclusivamente HTML, CSS e JavaScript.
2. Comparar o desempenho de diferentes modelos LLM (A, B, C e D) na geração de artefatos com níveis crescentes de complexidade, desde tarefas simples até implementações avançadas.
3. Identificar limitações e potencialidades dos modelos LLM atuais no contexto de desenvolvimento web, especialmente em relação à manipulação do DOM e implementação de interatividade.
4. Estabelecer métricas quantitativas para avaliação sistemática do código gerado, permitindo comparações objetivas entre os modelos e identificação de padrões de desempenho.
5. Fornecer insights práticos sobre a aplicabilidade dos modelos LLM como ferramentas de assistência ao desenvolvimento web, destacando cenários onde seu uso é mais promissor e áreas que ainda requerem supervisão humana significativa.

Estes objetivos foram concebidos para garantir que a metodologia não apenas produzisse resultados academicamente relevantes, mas também oferecesse contribuições práticas para desenvolvedores e organizações interessadas em incorporar LLMs em seus fluxos de trabalho de desenvolvimento.

4.3 MATERIAIS E MÉTODOS

Para a realização dos experimentos, foram cuidadosamente selecionados um conjunto de materiais e métodos alinhados com os desafios práticos encontrados no

desenvolvimento web no contexto real. Os modelos de linguagem selecionados são amplamente reconhecidos no mercado e tanto as ferramentas de análise e validação de código quanto os materiais de fundamentação teórica de boas práticas no desenvolvimento, são referências técnicas de altíssima confiabilidade.

4.3.1 MODELOS AVALIADOS

O experimento envolveu quatro modelos de linguagem de grande escala, GPT, Claude, Gemini e Grok. Os modelos foram denominados A, B, C e D respectivamente por questões de neutralidade na avaliação. Todos os modelos possuem capacidade de geração de código.

Tabela 3 - Comparação Técnica de Modelos de Linguagem Utilizados

Característica	Modelo A	Modelo B	Modelo C	Modelo D
Denominação	GPT	Claude	Gemini	Grok
Arquitetura	Transformer	Transformer	Transformer	Transformer
Parâmetros	200 bilhões (estimativa)	175 bilhões (estimativa)	500 bilhões (estimativa)	300 bilhões (estimativa)
Data de Lançamento	Mai/2024(4o)	Mai/2025(4.0)	Mai/2025(2.5)	Fev/2025(Grok 3)
Treinamento em Código	Extensivo	Especialista	Extensivo	Moderado
Versão Utilizada	GPT-4o	4.0 Sonnet	2.5 Flash	Grok 3

Fonte: Autoria Própria

4.3.2 FERRAMENTAS DE VALIDAÇÃO E ANÁLISE DE CÓDIGO

- ESLint / Prettier: Ferramentas open-source consideradas essenciais para codificação JavaScript limpa e bem formatada.

- SonarQube: Ferramenta de análise estática de código que ajuda a identificar problemas de qualidade e segurança em software, mantida pela empresa SonarSource.
- W3C Validator

Conjunto de ferramentas do World Wide Web Consortium (W3C) para verificar a conformidade de páginas web com padrões como HTML, XHTML e CSS. O W3C é responsável pelo desenvolvimento dos padrões web.

4.3.3 DEFINIÇÃO DOS ARTEFATOS

Os artefatos foram definidos com complexidade crescente para avaliar diferentes aspectos da capacidade de geração de código dos modelos. A tabela abaixo detalha os prompts utilizados para solicitar a geração dos artefatos, incluindo sua finalidade e nível de dificuldade:

Tabela 4 - Descrição de Prompts Utilizados

Prompt	Finalidade	Nível de Dificuldade
"Vamos criar um formulário de contato simples, com validação, em HTML, CSS e Javascript. Faça todo o código em apenas um arquivo HTML"	Avaliar capacidade básica de estruturação HTML, estilização CSS e validação com JavaScript	Baixo
"Vamos criar um filtro Dinâmico de Lista/Produtos em HTML, CSS e Javascript. Este artefato deve filtrar itens de uma lista em tempo real e mostrá-los como em um dashboard. Faça todo o código em apenas um arquivo HTML"	Avaliar capacidade de manipulação do DOM e implementação de lógica de filtragem dinâmica	Intermediário
"Vamos criar um artefato capaz de realizar o upload de	Avaliar capacidade de interação com APIs do navegador e manipulação	Intermediário

arquivos(imagens) com pré-visualização e validação. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML"	de arquivos	
"Vamos criar um artefato, um editor de código com sintaxe destacada e execução em tempo real. O artefato deve ser criado em HTML, CSS e Javascript. Não utilize código externo. Crie em apenas um arquivo HTML."	Avaliar capacidade de implementar funcionalidades avançadas que envolvem execução segura de código e manipulação complexa do DOM	Alto

Fonte: Autoria Própria

4.3.4 CRITÉRIOS DE AVALIAÇÃO

Para avaliar sistematicamente o desempenho dos modelos, foram definidas métricas quantitativas baseadas em trabalhos anteriores sobre avaliação de código gerado por LLMs. O Índice de Desempenho Geral (IDG) foi desenvolvido como uma métrica composta para facilitar a comparação entre os modelos.

As métricas utilizadas foram:

Tabela 5 - Métricas e Descrição dos Critérios de Avaliação

Critério	Escala	Descrição	Peso no IDG
Funcionalidade	0-10	Avalia se o código executa corretamente a função solicitada. Segundo Tufano et al. (2021), a funcionalidade é o aspecto mais fundamental na avaliação de código gerado automaticamente, pois determina se o artefato cumpre seu	25%

		propósito básico.	
Completude	0-10	Mede o grau em que o código implementa todos os requisitos especificados no prompt. Chen et al. (2023) destacam que a completude é essencial para determinar a capacidade do modelo de compreender e atender a todos os aspectos de uma especificação que ignoram a maioria dos requisitos.	25%
Qualidade do Código	0-10	Avalia a organização, legibilidade e aderência a boas práticas de programação. Conforme Allamanis et al. (2018), a qualidade do código afeta diretamente sua manutenibilidade e compreensibilidade a longo prazo.	25%
Complexidade da Solução	0-10	Analisa a sofisticação e robustez da implementação. Peng et al. (2022) argumentam que soluções mais complexas e bem elaboradas tendem a ser mais escaláveis e adaptáveis a diferentes contextos.	25%
Taxa de Sucesso	Sim/Não	Indicador binário que registra se o modelo conseguiu entregar um artefato minimamente funcional que atende aos requisitos essenciais do prompt.	N/A (indicador auxiliar)

Fonte: Autoria Própria

O Índice de Desempenho Geral (IDG) é calculado como a média aritmética das quatro primeiras métricas:

IDG = (Funcionalidade + Completude + Qualidade do Código + Complexidade da Solução) / 4

Esta abordagem de avaliação multidimensional é consistente com as recomendações de Finkelstein et al. (2022), que propõem a utilização de métricas compostas para avaliar artefatos de software gerados por IA, considerando tanto aspectos funcionais quanto qualitativos.

A escala de classificação do IDG foi definida como:

- 0-2.5: Insatisfatório
- 2.6-5.0: Regular
- 5.1-7.5: Bom
- 7.6-10: Excelente

4.4 COLETA DE DADOS

A coleta de dados foi realizada através de um processo estruturado em múltiplas etapas, garantindo a consistência e comparabilidade dos resultados:

4.4.1 PREPARAÇÃO DOS PROMPTS

Para cada artefato, foi elaborado um prompt específico com a descrição dos requisitos funcionais e técnicos necessários para a tarefa. A formulação dos prompts seguiu critérios de clareza e concisão, com o objetivo de evitar ambiguidades que pudessem favorecer ou prejudicar o desempenho de determinados modelos. Essa abordagem buscou garantir condições equitativas de avaliação, assegurando que todos os modelos recebessem instruções equivalentes e compreensíveis.

4.4.2 GERAÇÃO DOS ARTEFATOS

Para garantir condições experimentais consistentes durante a geração dos artefatos, cada um dos quatro modelos analisados (Modelos A, B, C e D) foi

submetido aos mesmos quatro prompts, totalizando 16 artefatos gerados. A fim de assegurar a comparabilidade dos resultados, foram seguidos rigorosamente os mesmos procedimentos em todos os casos.

Primeiramente, empregaram-se parâmetros de geração idênticos para todos os modelos. Além disso, não foram fornecidas informações contextuais adicionais além do próprio prompt, o que contribuiu para isolar o efeito do modelo sobre o conteúdo gerado.Outro aspecto fundamental foi a coleta exclusivamente da primeira resposta produzida por cada modelo, sem permitir iterações, revisões ou refinamentos posteriores, o que reforça a imparcialidade do processo.

Por fim, registrou-se integralmente o código gerado por cada modelo em resposta a cada prompt, assegurando a rastreabilidade e a integridade dos dados experimentais. Esses procedimentos padronizados garantiram a equidade nas condições de geração e permitiram uma análise comparativa válida entre os diferentes modelos.

4.4.3 AVALIAÇÃO DOS ARTEFATOS

O primeiro critério considerado foi a verificação de funcionalidade, na qual cada artefato foi testado em ambiente controlado para determinar se executava corretamente as funções solicitadas no prompt original. Em seguida, realizou-se a análise de completude, que consistiu na comparação entre os requisitos especificados nos prompts e as funcionalidades efetivamente implementadas nos códigos gerados.

Também foi avaliada a qualidade do código, com base em aspectos como organização estrutural, legibilidade, presença de comentários explicativos e aderência às boas práticas de programação comumente aceitas no desenvolvimento web. Outro aspecto considerado foi a complexidade da solução apresentada, incluindo sua sofisticação técnica, robustez e potencial de escalabilidade.

Com base nesses elementos, determinou-se a taxa de sucesso de cada artefato por meio de uma classificação binária (Sim/Não), indicando se ele atendia minimamente aos requisitos funcionais. Todos os resultados das avaliações foram

registrados em formulários padronizados, o que possibilitou o cálculo do Índice de Desempenho Geral (IDG) e permitiu uma comparação sistemática entre os diferentes modelos analisados.

4.4.4 DOCUMENTAÇÃO E ANÁLISE

Todos os artefatos gerados foram documentados integralmente, com o código original preservado sem qualquer modificação, assegurando a integridade dos dados para análise posterior. A partir desses registros, foi realizado o cálculo do Índice de Desempenho Geral (IDG) para cada artefato e modelo, permitindo uma avaliação quantitativa padronizada. Além disso, comparou-se o desempenho dos diferentes modelos em relação a cada artefato, a fim de identificar variações na eficácia das respostas geradas.

Essa metodologia de coleta e análise de dados foi cuidadosamente projetada para maximizar a objetividade e a reprodutibilidade do estudo, possibilitando a formulação de conclusões robustas e fundamentadas sobre o desempenho de modelos de linguagem de grande escala (LLMs) na geração de código voltado ao desenvolvimento web.

5 RESULTADOS

5.1 RESULTADOS ARTEFATO 1: FORMULÁRIO DE CONTATO COM VALIDAÇÃO

Modelo A

Tabela 6 - Resultado Modelo A, Primeiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um formulário de contato simples, com validação, em HTML, CSS e Javascript. Faça todo o código em apenas um arquivo HTML.”*
Resposta	*“Claro! Abaixo está um exemplo completo de um formulário de contato simples com validação usando apenas HTML, CSS e JavaScript, tudo em um único arquivo HTML:”*
Código Gerado	*Anexo A*
Avaliação	*8.0 - Excelente*

Fonte: Autoria Própria

O Modelo A entregou uma solução exemplar para formulário de contato, demonstrando excelente compreensão dos requisitos funcionais e implementando todas as funcionalidades essenciais com alta qualidade técnica.

Modelo B

Tabela 7 - Resultado Modelo B, Primeiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um formulário de contato simples, com validação, em HTML, CSS e Javascript. Faça todo o código em apenas um arquivo HTML.”*

Resposta	*“Criei um formulário de contato completo e moderno com as seguintes funcionalidades:”*
Código Gerado	*Anexo B*
Avaliação	*10.0 - Excelente*

Fonte: Autoria Própria

O Modelo B demonstrou superioridade absoluta ao entregar uma solução de nível profissional com arquitetura orientada a objetos, validação em tempo real, animações sofisticadas e funcionalidades avançadas que superam significativamente os requisitos básicos.

Modelo C

Tabela 8 - Resultado Modelo C, Primeiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um formulário de contato simples, com validação, em HTML, CSS e Javascript. Faça todo o código em apenas um arquivo HTML.”*
Resposta	*“Com certeza! Abaixo está o código completo para um formulário de contato simples com validação, tudo em um único arquivo HTML.”*
Código Gerado	*Anexo C*
Avaliação	*7.0 - Bom*

Fonte: Autoria Própria

O Modelo C ofereceu uma implementação sólida e bem executada dos requisitos fundamentais, mantendo simplicidade sem comprometer a funcionalidade.

Modelo D

Tabela 9 - Resultado Modelo D, Primeiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um formulário de contato simples, com validação, em HTML, CSS e Javascript. Faça todo o código em apenas um arquivo HTML.”*
Resposta	*“Vou criar um formulário de contato simples com validação de campos (nome, email e mensagem) usando HTML, CSS e JavaScript em um único arquivo. O formulário terá validação para campos obrigatórios e formato de email, com mensagens de erro claras e um design limpo.”*
Código Gerado	*Anexo D*
Avaliação	*6.5 - Bom*

Fonte: Autoria Própria

O Modelo D, embora funcional, apresentou uma abordagem mais minimalista que atende aos requisitos básicos mas não explora o potencial completo da funcionalidade solicitada.

Para a primeira tarefa, a criação de um formulário de contato com validação, todos os quatro modelos (A, B, C e D) foram capazes de gerar soluções funcionais e satisfatórias. O código produzido por cada modelo atendeu aos requisitos básicos de campos (nome, e-mail, assunto, mensagem) e implementou mecanismos de validação em JavaScript, conforme solicitado. Este resultado sugere que tarefas de desenvolvimento web front-end bem definidas e de baixa complexidade são consistentemente tratadas pelos LLMs avaliados.

5.2 RESULTADOS ARTEFATO 2: FILTRO DINÂMICO DE PRODUTOS

Modelo A

Tabela 10 - Resultado Modelo A, Segundo Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um filtro Dinâmico de Lista/Produtos em HTML, CSS e Javascript. Esse artefato deve filtrar itens de uma lista em tempo real e mostrá-los como em um dashboard. Faça todo o código em apenas um arquivo HTML.”*
Resposta	*“Claro! Aqui está um exemplo de um filtro dinâmico de produtos estilo dashboard, feito com HTML, CSS e JavaScript em um único arquivo. Ele permite digitar no campo de busca e filtrar itens em tempo real:”*
Código Gerado	*Anexo E*
Avaliação	*8.0 - Excelente*

Fonte: Autoria Própria

O Modelo A entregou uma solução exemplar para filtro dinâmico de produtos, demonstrando excelente compreensão dos requisitos funcionais e implementando todas as funcionalidades essenciais com alta qualidade técnica.

Modelo B

Tabela 11 - Resultado Modelo B, Segundo Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um filtro Dinâmico de Lista/Produtos em HTML, CSS e Javascript. Esse artefato deve filtrar itens de uma lista em tempo real e mostrá-los como em um dashboard. Faça todo o código em apenas um arquivo HTML.”*

Resposta	*“Vou criar um filtro dinâmico de produtos com visual moderno e funcionalidades completas em tempo real.”*
Código Gerado	*Anexo F*
Avaliação	*8.5 - Excelente*

Fonte: Autoria Própria

O Modelo B demonstrou capacidade superior ao criar uma solução robusta e visualmente atrativa, com múltiplos filtros funcionando harmoniosamente, estatísticas em tempo real e excelente experiência do usuário. A qualidade do código é alta, com boas práticas de organização e otimização.

Modelo C

Tabela 12 - Resultado Modelo C, Segundo Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um filtro Dinâmico de Lista/Produtos em HTML, CSS e Javascript. Esse artefato deve filtrar itens de uma lista em tempo real e mostrá-los como em um dashboard. Faça todo o código em apenas um arquivo HTML.”*
Resposta	*“Ótimo! Vamos criar um filtro dinâmico de lista/produtos com HTML, CSS e JavaScript, tudo em um único arquivo HTML para facilitar a visualização e o uso.”*
Código Gerado	*Anexo G*
Avaliação	*6.25 - Bom*

Fonte: Autoria Própria

O Modelo C apresenta uma solução intermediária com melhor estruturação e design mais elaborado, incluindo busca por múltiplos campos e melhor experiência visual. Embora seja limitado em funcionalidades avançadas, oferece uma base sólida e bem implementada.

Modelo D

Tabela 13 - Resultado Modelo D, Segundo Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um filtro Dinâmico de Lista/Produtos em HTML, CSS e Javascript. Esse artefato deve filtrar itens de uma lista em tempo real e mostrá-los como em um dashboard. Faça todo o código em apenas um arquivo HTML.”*
Resposta	*“Vou criar um filtro dinâmico de lista/produtos em um único arquivo HTML que inclui HTML, CSS e JavaScript para filtrar itens em tempo real e exibi-los em um formato de dashboard.”*
Código Gerado	*Anexo H*
Avaliação	*5.75 - Bom*

Fonte: Autoria Própria

O Modelo D oferece apenas o mínimo necessário, com funcionalidades muito básicas e design simplificado. Embora funcione, carece de recursos importantes para um filtro dinâmico robusto, resultando em uma solução que atende apenas parcialmente às expectativas de um sistema de filtragem de produtos profissional.

5.3 RESULTADOS ARTEFATO 3: FERRAMENTA DE UPLOAD E PRÉ-VISUALIZAÇÃO DE IMAGENS

Modelo A

Tabela 14 - Resultado Modelo A, Terceiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato capaz de realizar o upload de*

	arquivos(imagens) com pré-visualização e validação. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML"
Resposta	*"Claro! Abaixo está um arquivo HTML completo que implementa um artefato de upload de imagens com pré-visualização e validação. Ele é feito com HTML, CSS e JavaScript puros, tudo em um único arquivo."*
Código Gerado	*Anexo I*
Avaliação	*8.0 - Excelente*

Fonte: Autoria Própria

Implementou uma solução robusta e bem estruturada com validações adequadas, interface limpa e responsiva, e código bem organizado. Atende perfeitamente aos requisitos básicos com qualidade superior, mas sem os recursos avançados do Modelo B.

Modelo B

Tabela 15 - Resultado Modelo B, Terceiro Artefato

Interação	**Conteúdo**
Solicitação	*"Vamos criar um artefato capaz de realizar o upload de arquivos(imagens) com pré-visualização e validação. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML"*
Resposta	*"Vou criar um sistema completo de upload de imagens com pré-visualização e validação em um único arquivo HTML."*
Código Gerado	*Anexo J*
Avaliação	*10.0 - Excelente*

Fonte: Autoria Própria

Entregou uma solução profissional e completa com arquitetura orientada a objetos, interface moderna com gradientes e animações, drag & drop, upload múltiplo, sistema de estatísticas em tempo real, barra de progresso e experiência de usuário premium. Código exemplar que seria adequado para produção.

Modelo C

Tabela 16 - Resultado Modelo C, Terceiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato capaz de realizar o upload de arquivos(imagens) com pré-visualização e validação. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML”*
Resposta	*“Ok! Vamos criar um artefato em um único arquivo HTML que inclui HTML, CSS e JavaScript para upload de imagens com pré-visualização e validação.”*
Código Gerado	*Anexo K*
Avaliação	*7.0 - Bom*

Fonte: Autoria Própria

Ofereceu uma implementação funcional e organizada com drag & drop e validações corretas. Código limpo e bem comentado, porém com interface mais simples e limitada a um arquivo por vez. Cumpre o objetivo básico sem inovações.

Modelo D

Tabela 17 - Resultado Modelo D, Terceiro Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato capaz de realizar o upload de arquivos(imagens) com pré-visualização e validação. O*

	artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML”
Resposta	*“Vamos criar um artefato para upload de imagens com pré-visualização e validação usando HTML, CSS e JavaScript em um único arquivo. O código incluirá:”*
Código Gerado	*Anexo L*
Avaliação	*6.0 - Bom*

Fonte: Autoria Própria

Entregou apenas o mínimo necessário com interface extremamente básica, sem drag & drop ou recursos modernos. Embora funcional, demonstra pouca sofisticação técnica e experiência de usuário muito limitada.

5.4 RESULTADOS ARTEFATO 4: EDITOR DE CÓDIGO AO VIVO

Modelo A

Tabela 18 - Resultado Modelo A, Quarto Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato, um editor de código com sintaxe destacada e execução em tempo real. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML. Não utilize bibliotecas ou fontes externas.”*
Resposta	*“Claro! Vou criar um arquivo HTML único que contenha:* ● *Um editor de código com destaque básico de sintaxe (para JavaScript).* ● *Uma área onde o código é executado em tempo real e o resultado exibido.*

	● *Tudo feito só com HTML, CSS e JS puro, sem bibliotecas externas.”*
Código Gerado	*Anexo M*
Avaliação	*4.25 - Regular*

Fonte: Autoria Própria

O Modelo A criou um editor funcional mas muito limitado. A qualidade do código é boa, mas a funcionalidade não atende completamente ao requisito de um editor de código ao vivo para desenvolvimento web.

Modelo B

Tabela 19 - Resultado Modelo B, Quarto Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato, um editor de código com sintaxe destacada e execução em tempo real. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML. Não utilize bibliotecas ou fontes externas.”*
Resposta	*“Vou criar um editor de código HTML/JS com visualização em tempo real para você!”*
Código Gerado	*Anexo N*
Avaliação	*5.0 - Regular*

Fonte: Autoria Própria

O Modelo B, apesar da excelente qualidade estrutural, falha no requisito fundamental. A boa organização do código não compensa a falta de funcionalidade real.

Modelo C

Tabela 20 - Resultado Modelo C, Quarto Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato, um editor de código com sintaxe destacada e execução em tempo real. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML. Não utilize bibliotecas ou fontes externas.”*
Resposta	*“Com certeza! Vamos criar um editor de código com destaque de sintaxe e execução em tempo real, tudo em um único arquivo HTML, utilizando apenas HTML, CSS e JavaScript puros.”*
Código Gerado	*Anexo O*
Avaliação	*4.25 - Regular*

Fonte: Autoria Própria

O Modelo C oferece uma abordagem mais simples mas ainda falha no requisito fundamental. A simplicidade não compensa a falta de funcionalidade real e a menor sofisticação comparada aos outros modelos.

Modelo D

Tabela 21 - Resultado Modelo D, Quarto Artefato

Interação	**Conteúdo**
Solicitação	*“Vamos criar um artefato, um editor de código com sintaxe destacada e execução em tempo real. O artefato deve ser criado em HTML, CSS e Javascript. Crie em apenas um arquivo HTML. Não utilize bibliotecas ou fontes externas.”*
Resposta	*“Vou criar um editor de código simples com destaque de sintaxe para JavaScript e execução em tempo real, tudo em*

	um único arquivo HTML.”
Código Gerado	*Anexo P*
Avaliação	*4.0 - Regular*

Fonte: Autoria Própria

O Modelo D falhou em entregar um editor de código ao vivo funcional. Embora o código seja estruturalmente organizado, a solução fundamental está incorreta para o problema proposto.

5.5 ANÁLISE COMPARATIVA

Para facilitar a comparação entre os modelos, foi realizada uma análise quantitativa utilizando as métricas definidas na metodologia. As tabelas a seguir apresentam o desempenho detalhado de cada modelo em cada artefato.

5.5.1 TABELAS COMPARATIVAS POR ARTEFATO

Artefato 1: Formulário de Contato com Validação

Tabela 22 - Análise Comparativa dos Resultados do Artefato 1

Métrica	**Modelo A**	**Modelo B**	**Modelo C**	**Modelo D**
Funcionalidade (0 - 10)	9.0	10.0	8	8
Completude (0 - 10)	8.0	10.0	7	6
Qualidade do Código (0 - 10)	8.0	10.0	7	7
Complexidade da Solução (0 - 10)	7.0	10.0	6	5
Taxa de Sucesso	Sim	Sim	Sim	Sim

Índice de Desempenho Geral (IDG)	8.0	10.0	7	6.5

Fonte: Autoria Própria

Artefato 2: Filtro Dinâmico de Produtos

Tabela 23 - Análise Comparativa dos Resultados do Artefato 2

Métrica	**Modelo A**	**Modelo B**	**Modelo C**	**Modelo D**
Funcionalidade (0 - 10)	9.0	9.0	7.0	7.0
Completude (0 - 10)	8.0	9.0	6.0	5.0
Qualidade do Código (0 - 10)	8.0	8.0	7.0	6.0
Complexidade da Solução (0 - 10)	7.0	8.0	5.0	5.0
Taxa de Sucesso	Sim	Sim	Sim	Sim
Índice de Desempenho Geral (IDG)	8.0	8.5	6.25	5.75

Fonte: Autoria Própria

Artefato 3: Upload de Imagens com Pré-visualização

Tabela 24 - Análise Comparativa dos Resultados do Artefato 3

Métrica	**Modelo A**	**Modelo B**	**Modelo C**	**Modelo D**
Funcionalidade (0 - 10)	9.0	10.0	8.0	6.0
Completude (0 - 10)	8.0	10.0	7.0	5.0
Qualidade do Código (0 - 10)	8.0	10.0	7.0	6.0

Complexidade da Solução (0 - 10)	7.0	10.0	6.0	5.0
Taxa de Sucesso	Sim	Sim	Sim	Sim
Índice de Desempenho Geral (IDG)	8.0	10.0	7.0	6.0

Fonte: Autoria Própria

Artefato 4: Editor de Código ao Vivo

Tabela 25 - Análise Comparativa dos Resultados do Artefato 4

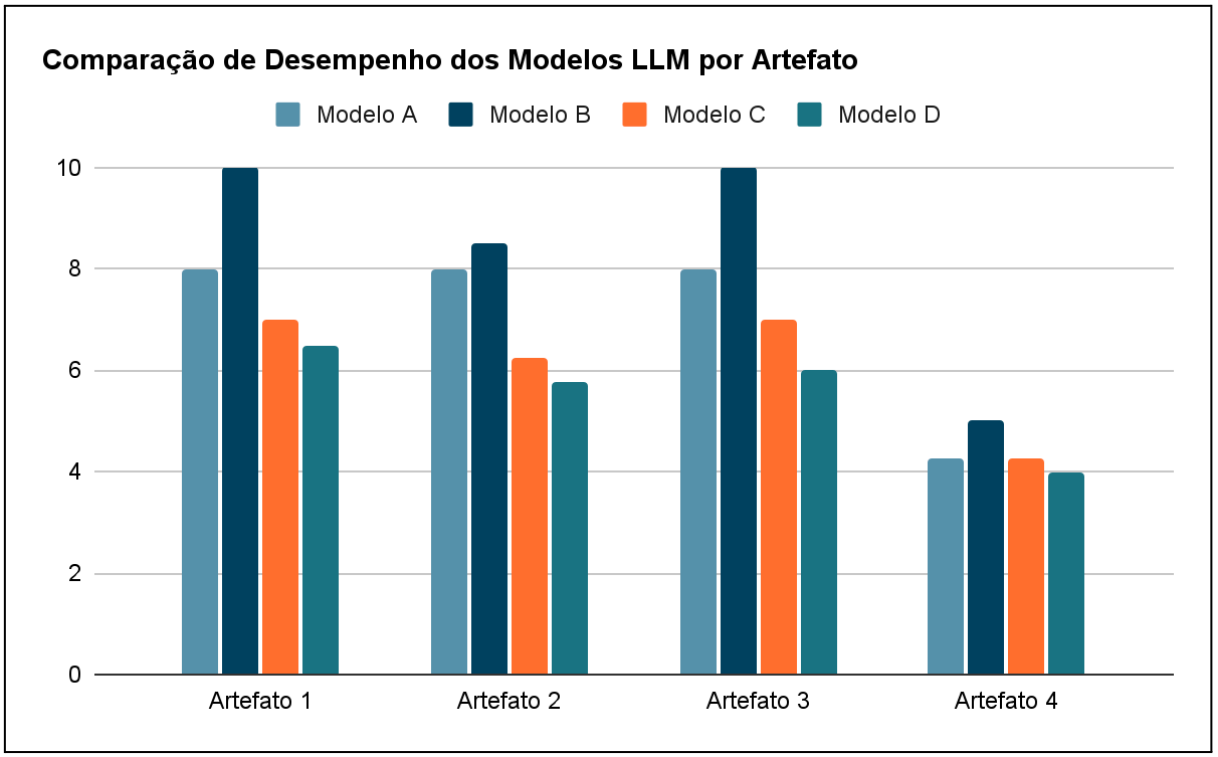
Métrica	**Modelo A**	**Modelo B**	**Modelo C**	**Modelo D**
Funcionalidade (0 - 10)	2.0	2.0	2.0	2.0
Completude (0 - 10)	3.0	4.0	5.0	5.0
Qualidade do Código (0 - 10)	7.0	7.0	6.0	5.0
Complexidade da Solução (0 - 10)	5.0	7.0	4.0	4.0
Taxa de Sucesso	Não	Não	Não	Não
Índice de Desempenho Geral (IDG)	4.25	5.0	4.25	4.0

Fonte: Autoria Própria

5.5.2 VISUALIZAÇÃO GRÁFICA DO DESEMPENHO

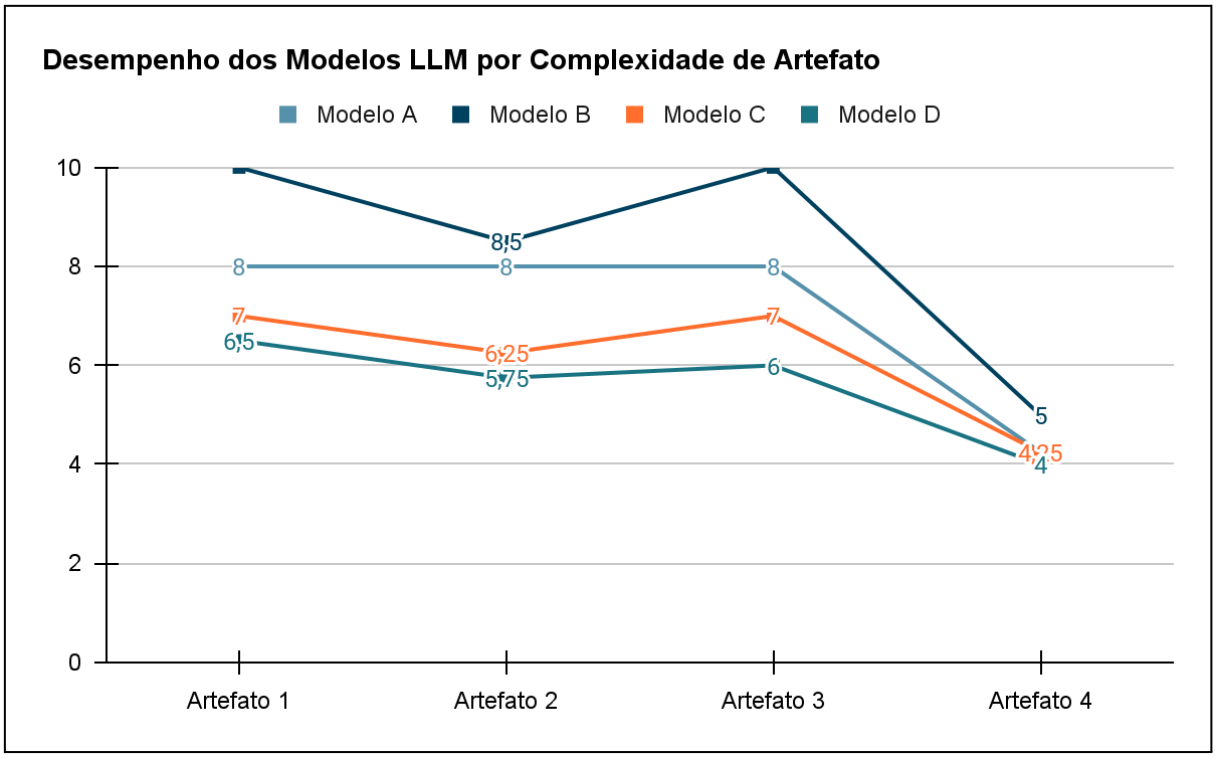
Para melhor visualização das tendências de desempenho, foram gerados dois gráficos comparativos:

Figura 6 - Comparação de Desempenho dos Modelos LLM por Artefato



Fonte: Autoria Própria

Figura 7 - Desempenho dos Modelos LLM por Complexidade de Artefato



Fonte: Autoria Própria

5.6 DISCUSSÃO GERAL DOS RESULTADOS

Os experimentos indicam que os LLMs avaliados demonstram proficiência na geração de código para tarefas de desenvolvimento web front-end de baixa e média complexidade, como formulários e componentes interativos simples. A capacidade de seguir instruções precisas e gerar código funcional para esses cenários é evidente na maioria dos modelos.

O Modelo B apresentou um desempenho consistentemente superior nas tarefas de complexidade intermediária, oferecendo soluções mais completas e bem estruturadas, embora mais verbosas. Isso sugere uma capacidade diferenciada de interpretação de requisitos e geração de código robusto, mas levanta a questão do equilíbrio entre completude e concisão.

A dificuldade generalizada em completar a tarefa de alta complexidade (editor de código ao vivo) aponta para limitações atuais significativas no manejo de interações avançadas do DOM e na execução de código dinâmico, especialmente

quando restritos a implementações puras de HTML, CSS e JavaScript sem bibliotecas auxiliares.

A natureza do prompt, intencionalmente menos detalhada para avaliar a capacidade intrínseca dos modelos, pode ter influenciado esse resultado, indicando que tarefas muito complexas podem exigir engenharia de prompt mais sofisticada ou abordagens iterativas.

6 CONCLUSÃO

Esta pesquisa teve como objetivo central avaliar criticamente o desempenho de diferentes Modelos de Linguagem de Grande Escala (LLMs) na geração de código para aplicações web, com enfoque no front-end, por meio de uma abordagem empírica, aplicada e comparativa. A investigação buscou compreender a aplicabilidade prática, os limites técnicos e as potencialidades dessas ferramentas no contexto profissional de desenvolvimento web, considerando critérios como funcionalidade, completude, qualidade de código e complexidade da solução.

A fundamentação teórica demonstrou que os LLMs, enquanto tecnologias derivadas da inteligência artificial generativa, representam um avanço significativo no campo do Processamento de Linguagem Natural, com impacto direto em tarefas de codificação e suporte técnico. Verificou-se que essas ferramentas estão inseridas em um contexto de transformação digital que abrange, simultaneamente, aspectos sociais, econômicos, educacionais e éticos, reforçando sua complexidade como objeto de estudo.

A análise experimental confirmou que todos os modelos avaliados foram capazes de gerar soluções funcionais para tarefas de complexidade baixa e média, com destaque para os modelos A (GPT) e B (Claude), que apresentaram desempenho superior nas métricas avaliadas. Em contrapartida, modelos como C (Gemini) e D (Grok) demonstraram limitações na completude e qualidade de código em cenários mais complexos. Tal variação reforça a importância de uma abordagem crítica e criteriosa na adoção dessas tecnologias em fluxos reais de desenvolvimento.

Dentre os principais achados, destaca-se que os LLMs têm elevado potencial como assistentes de codificação, oferecendo ganhos de produtividade, acessibilidade e padronização. Entretanto, também foi possível identificar desafios significativos, como a necessidade de supervisão humana constante, a gestão de vieses algorítmicos e a limitação na implementação de funcionalidades mais sofisticadas ou específicas. Isso sugere que, embora promissoras, essas ferramentas ainda não substituem integralmente o papel do desenvolvedor humano — mas o ampliam e o transformam.

Portanto, conclui-se que a utilização de LLMs no desenvolvimento web é viável e vantajosa, desde que ocorra de forma consciente, supervisionada e integrada a práticas consolidadas da engenharia de software. O estudo reforça a ideia de que o valor dessas ferramentas não reside apenas em sua capacidade técnica, mas também em sua articulação com o conhecimento humano, abrindo novas possibilidades de colaboração entre profissionais e sistemas inteligentes.

Como encaminhamento para futuras pesquisas, recomenda-se a ampliação da amostra de modelos testados, a inclusão de tarefas de backend e a análise longitudinal do desempenho dos LLMs em ambientes reais de produção. Além disso, seria relevante aprofundar investigações sobre a ética algorítmica e o impacto socioeconômico da automatização da codificação no mercado de trabalho técnico e educacional.

REFERÊNCIAS

ALLAMANIS, M.; BARR, E. T.; DEVANBU, P.; SUTTON, C. *A survey of machine learning for big code and naturalness*. *ACM Computing Surveys*, v. 51, n. 4, p. 1–37, 2018.

AZAMBUJA, C. C. de; SILVA, G. F. da. *Novos desafios para a educação na Era da Inteligência Artificial*. *Unisinos Journal of Philosophy*, v. 25, n. 1, p. 1–16, 2024. Disponível em: https://www.scielo.br/j/fsu. Acesso em: 15 maio 2025.

BRYNJOLFSSON, Erik; MCAFEE, Andrew. *The second machine age: Work, progress, and prosperity in a time of brilliant technologies*. 1. ed. W. W. Norton & Company, 2014.

CHEN, Xiaojin; LIU, Di; LE, Quoc. *Evaluating large language models on code generation: Challenges and opportunities*. In: *Proceedings of the 45th International Conference on Software Engineering (ICSE '23)*. 2023. p. 1123–1134.

COSTA JÚNIOR, J. F.; LIMA, Uilliane Faustino de; LEME, Mário Domingos; MORAES, Leonardo Silva; COSTA, Jonas Bezerra da; BARROS, Diogo Magalhães de; SOUSA, Maria Aparecida de Moura Amorim; OLIVEIRA, Luis Carlos Ferreira de. *A inteligência artificial como ferramenta de apoio no ensino superior*. *Rebena - Revista Brasileira de Ensino e Aprendizagem*, v. 6, p. 246–269, 2023. Disponível em: https://rebena.emnuvens.com.br/revista/article/view/111. Acesso em: 08 mai. 2025.

CUADERNOS DE EDUCACIÓN Y DESARROLLO. O impacto da Inteligência Artificial no desenvolvimento das competências acadêmicas. Disponível em: https://cuedespyd.hypotheses.org/. Acesso em: 12 jun. 2025.

DOMINGOS, Pedro. *O algoritmo mestre: como a busca pelo algoritmo de machine learning definitivo recriará nosso mundo*. Tradução de Aldir José Coelho Corrêa da Silva. São Paulo: Novatec, 2017.

FINKELSTEIN, Adam; HARMAN, Mark; JIA, Yanjun. *Automated software engineering using large language models: Promises, challenges, and evaluation frameworks*. *IEEE Transactions on Software Engineering*, v. 48, n. 10, p. 3589–3612, 2022.

GALLEGOS, I. O.; ROSSI, R. A.; BARROW, J.; TANJIM, M.; KIM, S.; DERNO(NC)OURT, F.; YU, T.; ZHANG, R.; AHMED, N. K. *Bias and Fairness in Large Language Models: A Survey*. 2023.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1997.

HALLIE, Lydia; OSMANI, Addy. *Learning Patterns: Patterns for building powerful web apps with vanilla JavaScript and React*. 2021.

HUSOM, Erik; GOKNIL, Arda; SHAR, L. K.; SEN, Sagar. *The Price of Prompting: Profiling Energy Use in Large Language Models Inference*. 2024.

JORDAN, Michael I. *Artificial Intelligence—The Revolution Hasn't Happened Yet*. *Harvard Data Science Review*, v. 1, n. 1, 2019. Disponível em: https://doi.org/10.1162/99608f92.f06c6e61. Acesso em: 07 jun. 2025.

JURISTO, Natalia; MORENO, Ana M. *Basics of Software Engineering Experimentation*. Berlin: Springer, 2013.

KALUZA, Marin; KALANJ, Marijana; VUKELIC, Bernard. *A Comparison of Back-End Frameworks for Web Application Development*. 2019.

LEE, Kai-Fu; QIUFAN, Chen. *AI 2041: Ten Visions for Our Future*. New York: Currency, 2021.

LUCKIN, Rose et al. *Intelligence Unleashed: An Argument for AI in Education*. Pearson Education, 2016.

MAI, Yubo; GAO, Zhipeng; HU, Xing; BAO, Lingfeng; LIU, Yu; SUN, JianLing. *Are Human Rules Necessary? Generating Reusable APIs with CoT Reasoning and In-Context Learning*. *Proceedings of the ACM on Software Engineering*, v. 1, n. FSE, Art. 104, jul. 2024. Disponível em: https://doi.org/10.1145/3660811. Acesso em: 9 jun. 2025.

MANYIKA, James; CHUI, Michael; MUNTHA, Susan; BIELSKI, Boris; MADHANI, Jean-Paul; JAIDEE, Caroline; GJORÇE, Matthias; HEMEL, Danielle; MURRAY,

Ashish; DING, Diana; GARR, Eric. *Jobs Lost, Jobs Gained: What the Future of Work Will Mean for Jobs, Skills, and Wages*. McKinsey Global Institute, 2017.

MARTIN, Robert C. *Clean Code*. 2008.

MCLUHAN, Marshall. *Os meios de comunicação como extensões do homem*. São Paulo: Cultrix, 1969.

MEANS, Barbara et al. *The Effectiveness of Online and Blended Learning: A Meta-Analysis of the Empirical Literature*. U.S. Department of Education, 2013.

PENG, Z.; ZHAO, S.; LIU, C.; LIN, X.; XIE, B. *Deep learning for code generation: A systematic review and future directions*. *ACM Computing Surveys*, v. 55, n. 2, p. 1–41, 2022.

RASCHKA, Sebastian. *Build a Large Language Model (From Scratch)*. 2024.

RUSSELL, Stuart J.; NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 3. ed. Upper Saddle River: Prentice Hall, 2010.

RESNICK, Mitchel; MALONEY, John; MONROY-HERNÁNDEZ, Andrés; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen; MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay; SILVERMAN, Brian; KAFKA, Yasmin. *Scratch: Programming for all*. *Communications of the ACM*, v. 52, n. 11, p. 60–67, 2009.

SALAS-ZÁRATE, María del Pilar; ALOR-HERNÁNDEZ, Giner; VALENCIA-GARCÍA, Rafael; RODRÍGUEZ-MAZAHUA, Lisbeth; RODRÍGUEZ-GONZÁLEZ, Alejandro; LÓPEZ-CUADRADO, José Luis. *Analyzing Best Practices on Web Development Frameworks*. 2014.

SHUTE, V. J.; RAHIMI, S. Review of computer-based assessment for learning in elementary and secondary education. *Journal of Computer Assisted Learning*, v. 33, p. 1–19, 2017. Disponível em: https://onlinelibrary.wiley.com/doi/10.1111/jcal.12172. Acesso em: 04 mai. 2025.

SICHMAN, Jaime Simão. Inteligência artificial e sociedade: avanços e riscos. *Estudos Avançados*, v. 35, n. 101, p. 37–49, 2021.

SLIDESGO SCHOOL. Pesquisa sobre IA na educação: insights exclusivos da Slidesgo sobre ferramentas de IA para educação. *Slidesgo*, 2025. Disponível em: https://slidesgo.com/pt/slidesgo-school/novidades/pesquisa-sobre-ia-na-educacao. Acesso em: 14 abr. 2025.

SMUTNY, Pavel; BJOKO, Michal. Comparative Analysis of Chatbots Using Large Language Models for Web Development Tasks. 2024.

STEFANOV, Stoyan. *JavaScript Patterns*. 2010.

SHVETS, Alexander. *Dive Into Design Patterns*. 2018.

SWACHA, J.; KULPA, A. Evaluation of popularity and multiaspectual comparison of widely used web development frameworks. *Computer Science Review*, v. 37, 2023.

TUFANO, Marco; WATSON, Catherine; BAVOTA, Gian; DI PENTA, Massimiliano; WHITE, Martin; POSHYVANYK, Denys. *On learning meaningful code changes via neural machine translation*. In: *Proceedings of the 43rd International Conference on Software Engineering (ICSE '21)*, p. 954–965, 2021.

TURING, A. M. Computing Machinery and Intelligence. *Mind*, v. 59, n. 236, p. 433–460, 1950. Disponível em: https://doi.org/10.1093/mind/LIX.236.433. Acesso em: 01 jun. 2025.

VINA, Abirami. From code to conversation: how does an LLM work? *Ultralytics*, 18 nov. 2024. Disponível em: https://www.ultralytics.com/pt/blog/from-code-to-conversation-how-does-an-llm-work. Acesso em: 12 jun. 2025.

WILLIAMSON, Ben. *Big Data in Education: The Digital Future of Learning, Policy and Practice*. London: SAGE Publications, 2017.

WOHLIN, Claes; RUNESON, Per; HÖST, Martin; OHLSSON, Magnus C.; REGNELL, Björn; WESSLÉN, Anders. *Experimentation in Software Engineering*. Berlin: Springer, 2012.

XUE, Zhipeng; GAO, Zhipeng; WANG, Shaohua; HU, Xing; XIA, Xin; LI Shanping. SelfPiCo: Self-Guided Partial Code Execution with LLMs. In: *Proceedings of the 33rd*

ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '24), Viena, Áustria, 16–20 set. 2024. New York: ACM, p. 13. Disponível em: https://doi.org/10.1145/3650212.3680368. Acesso em: 9 jun. 2025.

ANEXO A - CÓDIGO GERADO - MODELO A, ARTEFATO 1

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Formulário de Contato</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f3f3f3;
      padding: 20px;
    }

    .container {
      max-width: 500px;
      margin: auto;
      background: #fff;
      padding: 30px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }

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

    label {
      display: block;
      margin-top: 15px;
    }

    input, textarea {
      width: 100%;
      padding: 10px;
      margin-top: 5px;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-sizing: border-box;
    }

    button {
      margin-top: 20px;
      width: 100%;
      padding: 12px;
      background-color: #007bff;
      color: white;
      border: none;
      border-radius: 5px;
      font-size: 16px;
      cursor: pointer;
    }
```

```
    button:hover {
      background-color: #0056b3;
    }

    .error {
      color: red;
      font-size: 14px;
      margin-top: 5px;
    }

    .success {
      color: green;
      font-size: 16px;
      text-align: center;
      margin-top: 20px;
    }
  </style>
</head>
<body>

<div class="container">
  <h2>Contato</h2>
  <form id="contactForm" novalidate>
    <label for="name">Nome:</label>
    <input type="text" id="name" name="name">
    <div class="error" id="nameError"></div>

    <label for="email">E-mail:</label>
    <input type="email" id="email" name="email">
    <div class="error" id="emailError"></div>

    <label for="message">Mensagem:</label>
    <textarea id="message" name="message" rows="5"></textarea>
    <div class="error" id="messageError"></div>

    <button type="submit">Enviar</button>
    <div class="success" id="successMessage"></div>
  </form>
</div>

<script>
  const form = document.getElementById("contactForm");
  const nameInput = document.getElementById("name");
  const emailInput = document.getElementById("email");
  const messageInput = document.getElementById("message");

  const nameError = document.getElementById("nameError");
  const emailError = document.getElementById("emailError");
  const messageError = document.getElementById("messageError");
  const successMessage = document.getElementById("successMessage");

  form.addEventListener("submit", function(event) {
    event.preventDefault();

    // Limpar mensagens de erro
    nameError.textContent = "";
    emailError.textContent = "";
    messageError.textContent = "";
```

```
    successMessage.textContent = "";

    let isValid = true;

    // Validação do nome
    if (nameInput.value.trim() === "") {
      nameError.textContent = "Por favor, insira seu nome.";
      isValid = false;
    }

    // Validação do e-mail
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (emailInput.value.trim() === "") {
      emailError.textContent = "Por favor, insira seu e-mail.";
      isValid = false;
    } else if (!emailRegex.test(emailInput.value.trim())) {
      emailError.textContent = "Insira um e-mail válido.";
      isValid = false;
    }

    // Validação da mensagem
    if (messageInput.value.trim() === "") {
      messageError.textContent = "Por favor, escreva sua mensagem.";
      isValid = false;
    }

    if (isValid) {
      successMessage.textContent = "Mensagem enviada com sucesso!";
      form.reset();
    }
  });
</script>

</body>
</html>
```

ANEXO B - CÓDIGO GERADO - MODELO B, ARTEFATO 1

```
<!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 Contato</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
```

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 20px;
}

.container {
    background: rgba(255, 255, 255, 0.95);
    backdrop-filter: blur(10px);
    border-radius: 20px;
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.1);
    padding: 40px;
    width: 100%;
    max-width: 500px;
    transform: translateY(0);
    transition: transform 0.3s ease;
}

.container:hover {
    transform: translateY(-5px);
}

h1 {
    text-align: center;
    color: #333;
    margin-bottom: 30px;
    font-size: 2.5rem;
    background: linear-gradient(135deg, #667eea, #764ba2);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    background-clip: text;
}

.form-group {
    margin-bottom: 25px;
    position: relative;
}

label {
    display: block;
    margin-bottom: 8px;
    color: #555;
    font-weight: 600;
    font-size: 0.95rem;
}

input, textarea, select {
    width: 100%;
    padding: 15px;
    border: 2px solid #e1e5e9;
    border-radius: 12px;
    font-size: 1rem;
    transition: all 0.3s ease;
```

```
    background: rgba(255, 255, 255, 0.8);
}

input:focus, textarea:focus, select:focus {
    outline: none;
    border-color: #667eea;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
    transform: translateY(-2px);
}

textarea {
    resize: vertical;
    min-height: 120px;
}

.error {
    color: #e74c3c;
    font-size: 0.85rem;
    margin-top: 5px;
    display: block;
    opacity: 0;
    transform: translateY(-10px);
    transition: all 0.3s ease;
}

.error.show {
    opacity: 1;
    transform: translateY(0);
}

.input-error {
    border-color: #e74c3c !important;
    box-shadow: 0 0 0 3px rgba(231, 76, 60, 0.1) !important;
}

.input-success {
    border-color: #27ae60 !important;
}

button {
    width: 100%;
    padding: 18px;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    border: none;
    border-radius: 12px;
    font-size: 1.1rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.3s ease;
    position: relative;
    overflow: hidden;
}

button:hover {
    transform: translateY(-2px);
    box-shadow: 0 10px 25px rgba(102, 126, 234, 0.3);
}
```

```
button:active {
    transform: translateY(0);
}

button:disabled {
    opacity: 0.7;
    cursor: not-allowed;
    transform: none;
}

.success-message {
    background: linear-gradient(135deg, #27ae60, #2ecc71);
    color: white;
    padding: 20px;
    border-radius: 12px;
    text-align: center;
    margin-top: 20px;
    opacity: 0;
    transform: translateY(20px);
    transition: all 0.5s ease;
}

.success-message.show {
    opacity: 1;
    transform: translateY(0);
}

.loading {
    display: inline-block;
    width: 20px;
    height: 20px;
    border: 3px solid rgba(255, 255, 255, 0.3);
    border-radius: 50%;
    border-top-color: white;
    animation: spin 1s ease-in-out infinite;
    margin-right: 10px;
}

@keyframes spin {
    to { transform: rotate(360deg); }
}

.checkbox-group {
    display: flex;
    align-items: center;
    gap: 10px;
}

.checkbox-group input[type="checkbox"] {
    width: auto;
    margin: 0;
}

@media (max-width: 600px) {
    .container {
        padding: 25px;
        margin: 10px;
```

```
        }

        h1 {
            font-size: 2rem;
        }
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Entre em Contato</h1>
    <form id="contactForm">
      <div class="form-group">
        <label for="name">Nome Completo *</label>
        <input type="text" id="name" name="name" required>
        <span class="error" id="nameError"></span>
      </div>

      <div class="form-group">
        <label for="email">E-mail *</label>
        <input type="email" id="email" name="email" required>
        <span class="error" id="emailError"></span>
      </div>

      <div class="form-group">
        <label for="phone">Telefone</label>
        <input type="tel" id="phone" name="phone">
        <span class="error" id="phoneError"></span>
      </div>

      <div class="form-group">
        <label for="subject">Assunto *</label>
        <select id="subject" name="subject" required>
          <option value="">Selecione um assunto</option>
          <option value="duvida">Dúvida</option>
          <option value="sugestao">Sugestão</option>
          <option value="reclamacao">Reclamação</option>
          <option value="elogio">Elogio</option>
          <option value="outro">Outro</option>
        </select>
        <span class="error" id="subjectError"></span>
      </div>

      <div class="form-group">
        <label for="message">Mensagem *</label>
        <textarea id="message" name="message" placeholder="Digite sua mensagem..."
required></textarea>
        <span class="error" id="messageError"></span>
      </div>

      <div class="form-group">
        <div class="checkbox-group">
          <input type="checkbox" id="newsletter" name="newsletter">
          <label for="newsletter">Desejo receber newsletter</label>
        </div>
      </div>

      <div class="form-group">
```

```
            <div class="checkbox-group">
                <input type="checkbox" id="terms" name="terms" required>
                <label for="terms">Aceito os termos de uso e política de privacidade *</label>
            </div>
            <span class="error" id="termsError"></span>
        </div>

        <button type="submit" id="submitBtn">
            Enviar Mensagem
        </button>
    </form>

    <div class="success-message" id="successMessage">
        <h3>Mensagem enviada com sucesso!</h3>
        <p>Obrigado pelo seu contato. Responderemos em breve.</p>
    </div>
</div>

<script>
    class ContactForm {
        constructor() {
            this.form = document.getElementById('contactForm');
            this.submitBtn = document.getElementById('submitBtn');
            this.successMessage = document.getElementById('successMessage');
            this.init();
        }

        init() {
            this.form.addEventListener('submit', this.handleSubmit.bind(this));
            this.addRealTimeValidation();
            this.formatPhoneInput();
        }

        addRealTimeValidation() {
            const inputs = this.form.querySelectorAll('input, textarea, select');
            inputs.forEach(input => {
                input.addEventListener('blur', () => this.validateField(input));
                input.addEventListener('input', () => this.clearError(input));
            });
        }

        formatPhoneInput() {
            const phoneInput = document.getElementById('phone');
            phoneInput.addEventListener('input', (e) => {
                let value = e.target.value.replace(/\D/g, '');
                if (value.length <= 11) {
                    value = value.replace(/(\d{2})(\d{5})(\d{4})/, '($1) $2-$3');
                    value = value.replace(/(\d{2})(\d{4})(\d{4})/, '($1) $2-$3');
                    value = value.replace(/(\d{2})(\d{1,5})/, '($1) $2');
                    value = value.replace(/(\d{2})/, '($1');
                }
                e.target.value = value;
            });
        }

        validateField(field) {
            const fieldName = field.name;
            const value = field.value.trim();
```

```
        let isValid = true;
        let errorMessage = '';

        switch (fieldName) {
            case 'name':
                if (!value) {
                    errorMessage = 'Nome é obrigatório';
                    isValid = false;
                } else if (value.length < 2) {
                    errorMessage = 'Nome deve ter pelo menos 2 caracteres';
                    isValid = false;
                } else if (!/^[a-zA-ZÀ-ÿ\s]+$/.test(value)) {
                    errorMessage = 'Nome deve conter apenas letras';
                    isValid = false;
                }
                break;

            case 'email':
                const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
                if (!value) {
                    errorMessage = 'E-mail é obrigatório';
                    isValid = false;
                } else if (!emailRegex.test(value)) {
                    errorMessage = 'E-mail inválido';
                    isValid = false;
                }
                break;

            case 'phone':
                if (value) {
                    const phoneRegex = /^\(\d{2}\)\s\d{4,5}-\d{4}$/;
                    if (!phoneRegex.test(value)) {
                        errorMessage = 'Formato de telefone inválido';
                        isValid = false;
                    }
                }
                break;

            case 'subject':
                if (!value) {
                    errorMessage = 'Selecione um assunto';
                    isValid = false;
                }
                break;

            case 'message':
                if (!value) {
                    errorMessage = 'Mensagem é obrigatória';
                    isValid = false;
                } else if (value.length < 10) {
                    errorMessage = 'Mensagem deve ter pelo menos 10 caracteres';
                    isValid = false;
                }
                break;

            case 'terms':
                if (!field.checked) {
                    errorMessage = 'Você deve aceitar os termos';
```

```
                isValid = false;
            }
            break;
        }

        this.showFieldValidation(field, isValid, errorMessage);
        return isValid;
    }

    showFieldValidation(field, isValid, errorMessage) {
        const errorElement = document.getElementById(field.name + 'Error');

        if (isValid) {
            field.classList.remove('input-error');
            field.classList.add('input-success');
            if (errorElement) {
                errorElement.textContent = '';
                errorElement.classList.remove('show');
            }
        } else {
            field.classList.remove('input-success');
            field.classList.add('input-error');
            if (errorElement) {
                errorElement.textContent = errorMessage;
                errorElement.classList.add('show');
            }
        }
    }

    clearError(field) {
        field.classList.remove('input-error');
        const errorElement = document.getElementById(field.name + 'Error');
        if (errorElement && errorElement.classList.contains('show')) {
            errorElement.classList.remove('show');
            setTimeout(() => {
                errorElement.textContent = '';
            }, 300);
        }
    }

    validateForm() {
        const requiredFields = this.form.querySelectorAll('[required]');
        let isFormValid = true;

        requiredFields.forEach(field => {
            if (!this.validateField(field)) {
                isFormValid = false;
            }
        });

        return isFormValid;
    }

    async handleSubmit(e) {
        e.preventDefault();

        if (!this.validateForm()) {
            return;
```

```
        }

        this.setLoadingState(true);

        // Simular envio do formulário
        try {
            await this.simulateFormSubmission();
            this.showSuccess();
            this.resetForm();
        } catch (error) {
            alert('Erro ao enviar formulário. Tente novamente.');
        } finally {
            this.setLoadingState(false);
        }
    }

    setLoadingState(loading) {
        if (loading) {
            this.submitBtn.disabled = true;
            this.submitBtn.innerHTML = '<span class="loading"></span>Enviando...';
        } else {
            this.submitBtn.disabled = false;
            this.submitBtn.innerHTML = 'Enviar Mensagem';
        }
    }

    simulateFormSubmission() {
        return new Promise((resolve) => {
            setTimeout(() => {
                console.log('Formulário enviado:', this.getFormData());
                resolve();
            }, 2000);
        });
    }

    getFormData() {
        const formData = new FormData(this.form);
        const data = {};
        for (let [key, value] of formData.entries()) {
            data[key] = value;
        }
        return data;
    }

    showSuccess() {
        this.form.style.display = 'none';
        this.successMessage.classList.add('show');

        setTimeout(() => {
            this.form.style.display = 'block';
            this.successMessage.classList.remove('show');
        }, 5000);
    }

    resetForm() {
        this.form.reset();
        const inputs = this.form.querySelectorAll('input, textarea, select');
        inputs.forEach(input => {
```

```
                input.classList.remove('input-error', 'input-success');
            });
        }
    }

    // Inicializar o formulário quando o DOM estiver carregado
    document.addEventListener('DOMContentLoaded', () => {
        new ContactForm();
    });
</script>
</body>
</html>
```

ANEXO C - CÓDIGO GERADO - MODELO C, ARTEFATO 1

```
<!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 Contato</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
        }

        .container {
            background-color: #fff;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 100%;
            max-width: 500px;
        }

        h2 {
            text-align: center;
            color: #333;
            margin-bottom: 25px;
        }

        .form-group {
            margin-bottom: 15px;
        }
```

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

        input[type="text"],
        input[type="email"],
        textarea {
            width: calc(100% - 20px);
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box;
            font-size: 16px;
        }

        textarea {
            resize: vertical;
            min-height: 100px;
        }

        input[type="submit"] {
            background-color: #007bff;
            color: white;
            padding: 12px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 18px;
            width: 100%;
            transition: background-color 0.3s ease;
        }

        input[type="submit"]:hover {
            background-color: #0056b3;
        }

        .error-message {
            color: red;
            font-size: 0.9em;
            margin-top: 5px;
            display: none; /* Escondido por padrão */
        }

        .success-message {
            color: green;
            font-size: 1em;
            text-align: center;
            margin-top: 20px;
            display: none; /* Escondido por padrão */
        }
    </style>
</head>
<body>
    <div class="container">
```

```
        <h2>Fale Conosco</h2>
        <form id="contactForm">
            <div class="form-group">
                <label for="name">Nome:</label>
                <input type="text" id="name" name="name" required>
                <div class="error-message" id="nameError">Por favor, insira seu nome.</div>
            </div>

            <div class="form-group">
                <label for="email">E-mail:</label>
                <input type="email" id="email" name="email" required>
                <div class="error-message" id="emailError">Por favor, insira um e-mail válido.</div>
            </div>

            <div class="form-group">
                <label for="message">Mensagem:</label>
                <textarea id="message" name="message" required></textarea>
                <div class="error-message" id="messageError">Por favor, digite sua
mensagem.</div>
            </div>

            <input type="submit" value="Enviar Mensagem">
            <div class="success-message" id="successMessage">Mensagem enviada com
sucesso!</div>
        </form>
    </div>

    <script>
        document.addEventListener('DOMContentLoaded', function() {
            const form = document.getElementById('contactForm');
            const nameInput = document.getElementById('name');
            const emailInput = document.getElementById('email');
            const messageInput = document.getElementById('message');

            const nameError = document.getElementById('nameError');
            const emailError = document.getElementById('emailError');
            const messageError = document.getElementById('messageError');
            const successMessage = document.getElementById('successMessage');

            form.addEventListener('submit', function(event) {
                event.preventDefault(); // Impede o envio padrão do formulário
                let isValid = true;

                // Limpa mensagens de erro anteriores
                nameError.style.display = 'none';
                emailError.style.display = 'none';
                messageError.style.display = 'none';
                successMessage.style.display = 'none';

                // Validação do Nome
                if (nameInput.value.trim() === '') {
                    nameError.style.display = 'block';
                    isValid = false;
                }

                // Validação do E-mail
                const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
                if (!emailPattern.test(emailInput.value.trim())) {
```

```
            emailError.style.display = 'block';
            isValid = false;
        }

        // Validação da Mensagem
        if (messageInput.value.trim() === '') {
            messageError.style.display = 'block';
            isValid = false;
        }

        if (isValid) {
            // Se tudo estiver válido, exibe a mensagem de sucesso
            successMessage.style.display = 'block';
            form.reset(); // Limpa o formulário
            // Aqui você poderia adicionar a lógica para enviar os dados para um servidor,
            // por exemplo, usando Fetch API ou XMLHttpRequest.
            console.log('Formulário enviado com sucesso!');
            console.log('Nome:', nameInput.value.trim());
            console.log('E-mail:', emailInput.value.trim());
            console.log('Mensagem:', messageInput.value.trim());
        }
    });
  });
  </script>
</body>
</html>
```

ANEXO D - CÓDIGO GERADO - MODELO D, ARTEFATO 1

```
<!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 Contato</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      background-color: #f0f2f5;
      margin: 0;
    }
    .container {
      background: white;
      padding: 2rem;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0,0,0,0.1);
```

```
            width: 100%;
            max-width: 400px;
        }
        h1 {
            text-align: center;
            color: #333;
            margin-bottom: 1.5rem;
        }
        .form-group {
            margin-bottom: 1rem;
        }
        label {
            display: block;
            margin-bottom: 0.5rem;
            color: #555;
        }
        input, textarea {
            width: 100%;
            padding: 0.75rem;
            border: 1px solid #ddd;
            border-radius: 4px;
            font-size: 1rem;
        }
        textarea {
            resize: vertical;
            min-height: 100px;
        }
        button {
            width: 100%;
            padding: 0.75rem;
            background: #007bff;
            color: white;
            border: none;
            border-radius: 4px;
            font-size: 1rem;
            cursor: pointer;
        }
        button:hover {
            background: #0056b3;
        }
        .error {
            color: #dc3545;
            font-size: 0.875rem;
            margin-top: 0.25rem;
            display: none;
        }
        .error.show {
            display: block;
        }
        input.error-border, textarea.error-border {
            border-color: #dc3545;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Formulário de Contato</h1>
        <form id="contactForm" novalidate>
```

```
        <div class="form-group">
            <label for="name">Nome:</label>
            <input type="text" id="name" name="name" required>
            <div id="nameError" class="error">Por favor, insira seu nome.</div>
        </div>
        <div class="form-group">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required>
            <div id="emailError" class="error">Por favor, insira um email válido.</div>
        </div>
        <div class="form-group">
            <label for="message">Mensagem:</label>
            <textarea id="message" name="message" required></textarea>
            <div id="messageError" class="error">Por favor, insira uma mensagem.</div>
        </div>
        <button type="submit">Enviar</button>
    </form>
</div>

<script>
    document.getElementById('contactForm').addEventListener('submit', function(e) {
        e.preventDefault();

        // Resetar estados de erro
        const inputs = document.querySelectorAll('input, textarea');
        const errors = document.querySelectorAll('.error');
        inputs.forEach(input => input.classList.remove('error-border'));
        errors.forEach(error => error.classList.remove('show'));

        // Obter valores dos campos
        const name = document.getElementById('name').value.trim();
        const email = document.getElementById('email').value.trim();
        const message = document.getElementById('message').value.trim();

        let isValid = true;

        // Validação do nome
        if (name === '') {
            document.getElementById('nameError').classList.add('show');
            document.getElementById('name').classList.add('error-border');
            isValid = false;
        }

        // Validação do email
        const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailPattern.test(email)) {
            document.getElementById('emailError').classList.add('show');
            document.getElementById('email').classList.add('error-border');
            isValid = false;
        }

        // Validação da mensagem
        if (message === '') {
            document.getElementById('messageError').classList.add('show');
            document.getElementById('message').classList.add('error-border');
            isValid = false;
        }
```

```
            // Se válido, exibir mensagem de sucesso
            if (isValid) {
                alert('Formulário enviado com sucesso!');
                this.reset();
            }
        });
    </script>
</body>
</html>
```

ANEXO E - CÓDIGO GERADO - MODELO A, ARTEFATO 2

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title>Filtro Dinâmico de Produtos</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 2rem;
    }

    h1 {
      text-align: center;
      margin-bottom: 1.5rem;
    }

    .search-box {
      max-width: 400px;
      margin: 0 auto 2rem;
      display: flex;
      justify-content: center;
    }

    .search-box input {
      width: 100%;
      padding: 0.75rem;
      font-size: 1rem;
      border: 1px solid #ccc;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }

    .product-grid {
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
      gap: 1rem;
```

```
    }

    .product-card {
      background: white;
      padding: 1rem;
      border-radius: 12px;
      box-shadow: 0 2px 8px rgba(0,0,0,0.1);
      text-align: center;
      transition: transform 0.2s;
    }

    .product-card:hover {
      transform: translateY(-5px);
    }

    .product-name {
      font-weight: bold;
      margin-bottom: 0.5rem;
    }

    .product-desc {
      font-size: 0.9rem;
      color: #666;
    }

    .no-results {
      text-align: center;
      margin-top: 2rem;
      font-size: 1.2rem;
      color: #888;
    }
  </style>
</head>
<body>
  <h1>Filtro Dinâmico de Produtos</h1>

  <div class="search-box">
    <input type="text" id="searchInput" placeholder="Digite para filtrar produtos...">
  </div>

  <div id="productGrid" class="product-grid"></div>
  <div id="noResults" class="no-results" style="display: none;">Nenhum produto
encontrado.</div>

  <script>
    const products = [
      { name: "Notebook Dell", desc: "Intel i5, 8GB RAM, SSD 256GB" },
      { name: "Smartphone Galaxy", desc: "Samsung S21 com câmera tripla" },
      { name: "Fone de Ouvido", desc: "Bluetooth, com cancelamento de ruído" },
      { name: "Monitor LG 24\"", desc: "Full HD com painel IPS" },
      { name: "Teclado Mecânico", desc: "Switch Red, RGB" },
      { name: "Mouse Gamer", desc: "Alta precisão com 6 botões" },
      { name: "Cadeira Gamer", desc: "Ergonômica, ajustável e confortável" },
      { name: "Tablet Android", desc: "10 polegadas, quad-core" },
    ];

    const productGrid = document.getElementById('productGrid');
    const searchInput = document.getElementById('searchInput');
```

```
    const noResults = document.getElementById('noResults');

    function renderProducts(filteredProducts) {
      productGrid.innerHTML = "";
      if (filteredProducts.length === 0) {
        noResults.style.display = "block";
        return;
      }

      noResults.style.display = "none";
      filteredProducts.forEach(p => {
        const card = document.createElement('div');
        card.className = 'product-card';
        card.innerHTML = `
          <div class="product-name">${p.name}</div>
          <div class="product-desc">${p.desc}</div>
        `;
        productGrid.appendChild(card);
      });
    }

    searchInput.addEventListener('input', () => {
      const query = searchInput.value.toLowerCase();
      const filtered = products.filter(p =>
        p.name.toLowerCase().includes(query) || p.desc.toLowerCase().includes(query)
      );
      renderProducts(filtered);
    });

    // Render inicial
    renderProducts(products);
  </script>
</body>
</html>
```

ANEXO F - CÓDIGO GERADO - MODELO B, ARTEFATO 2

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard - Filtro Dinâmico de Produtos</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
```

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    padding: 20px;
}

.dashboard {
    max-width: 1400px;
    margin: 0 auto;
    background: rgba(255, 255, 255, 0.95);
    border-radius: 20px;
    padding: 30px;
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.1);
    backdrop-filter: blur(10px);
}

.header {
    text-align: center;
    margin-bottom: 40px;
}

.header h1 {
    color: #2d3748;
    font-size: 2.5rem;
    margin-bottom: 10px;
    font-weight: 700;
}

.header p {
    color: #718096;
    font-size: 1.1rem;
}

.filters-section {
    background: white;
    padding: 25px;
    border-radius: 15px;
    margin-bottom: 30px;
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.08);
}

.filters-grid {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr 1fr auto;
    gap: 20px;
    align-items: end;
}

.filter-group {
    display: flex;
    flex-direction: column;
}

.filter-group label {
    font-weight: 600;
    color: #4a5568;
    margin-bottom: 8px;
```

```
    font-size: 0.9rem;
}

.search-input {
    padding: 12px 16px;
    border: 2px solid #e2e8f0;
    border-radius: 10px;
    font-size: 1rem;
    transition: all 0.3s ease;
    background: #f7fafc;
}

.search-input:focus {
    outline: none;
    border-color: #667eea;
    background: white;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.select-input {
    padding: 12px 16px;
    border: 2px solid #e2e8f0;
    border-radius: 10px;
    font-size: 1rem;
    background: #f7fafc;
    cursor: pointer;
    transition: all 0.3s ease;
}

.select-input:focus {
    outline: none;
    border-color: #667eea;
    background: white;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.clear-btn {
    padding: 12px 20px;
    background: #e53e3e;
    color: white;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    font-weight: 600;
    transition: all 0.3s ease;
    align-self: end;
}

.clear-btn:hover {
    background: #c53030;
    transform: translateY(-2px);
    box-shadow: 0 5px 15px rgba(197, 48, 48, 0.3);
}

.stats {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 20px;
```

```
    margin-bottom: 30px;
}

.stat-card {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    padding: 25px;
    border-radius: 15px;
    text-align: center;
    transform: translateY(0);
    transition: all 0.3s ease;
}

.stat-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 15px 30px rgba(102, 126, 234, 0.3);
}

.stat-number {
    font-size: 2.5rem;
    font-weight: 700;
    margin-bottom: 5px;
}

.stat-label {
    font-size: 0.9rem;
    opacity: 0.9;
}

.products-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 25px;
}

.product-card {
    background: white;
    border-radius: 15px;
    padding: 25px;
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.08);
    transition: all 0.3s ease;
    border: 2px solid transparent;
}

.product-card:hover {
    transform: translateY(-8px);
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.15);
    border-color: #667eea;
}

.product-header {
    display: flex;
    justify-content: space-between;
    align-items: start;
    margin-bottom: 15px;
}

.product-name {
```

```
    font-size: 1.3rem;
    font-weight: 700;
    color: #2d3748;
    margin-bottom: 5px;
}

.product-category {
    background: linear-gradient(45deg, #667eea, #764ba2);
    color: white;
    padding: 6px 12px;
    border-radius: 20px;
    font-size: 0.8rem;
    font-weight: 600;
}

.product-price {
    font-size: 1.5rem;
    font-weight: 700;
    color: #38a169;
    margin: 15px 0;
}

.product-stock {
    display: flex;
    align-items: center;
    gap: 8px;
    margin-bottom: 15px;
}

.stock-indicator {
    width: 10px;
    height: 10px;
    border-radius: 50%;
}

.stock-high { background: #38a169; }
.stock-medium { background: #ed8936; }
.stock-low { background: #e53e3e; }

.product-description {
    color: #718096;
    line-height: 1.5;
    margin-bottom: 20px;
}

.product-actions {
    display: flex;
    gap: 10px;
}

.btn {
    padding: 10px 20px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-weight: 600;
    transition: all 0.3s ease;
    flex: 1;
```

```
}

.btn-primary {
    background: #667eea;
    color: white;
}

.btn-primary:hover {
    background: #5a67d8;
    transform: translateY(-2px);
}

.btn-secondary {
    background: #e2e8f0;
    color: #4a5568;
}

.btn-secondary:hover {
    background: #cbd5e0;
    transform: translateY(-2px);
}

.no-results {
    text-align: center;
    padding: 60px 20px;
    color: #718096;
}

.no-results h3 {
    font-size: 1.5rem;
    margin-bottom: 10px;
}

.no-results p {
    font-size: 1.1rem;
}

.filter-animation {
    animation: filterFadeIn 0.5s ease-out;
}

@keyframes filterFadeIn {
    from {
        opacity: 0;
        transform: translateY(20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

@media (max-width: 768px) {
    .filters-grid {
        grid-template-columns: 1fr;
        gap: 15px;
    }
```

```
            .products-grid {
                grid-template-columns: 1fr;
            }

            .header h1 {
                font-size: 2rem;
            }
        }
    </style>
</head>
<body>
    <div class="dashboard">
        <div class="header">
            <h1>🛍️ Dashboard de Produtos</h1>
            <p>Filtros dinâmicos em tempo real</p>
        </div>

        <div class="filters-section">
            <div class="filters-grid">
                <div class="filter-group">
                    <label for="searchInput">🔍 Buscar Produto</label>
                    <input type="text" id="searchInput" class="search-input" placeholder="Digite o
nome do produto...">
                </div>

                <div class="filter-group">
                    <label for="categoryFilter">📂 Categoria</label>
                    <select id="categoryFilter" class="select-input">
                        <option value="">Todas as categorias</option>
                    </select>
                </div>

                <div class="filter-group">
                    <label for="priceFilter">💰 Faixa de Preço</label>
                    <select id="priceFilter" class="select-input">
                        <option value="">Todos os preços</option>
                        <option value="0-50">R$ 0 - R$ 50</option>
                        <option value="50-100">R$ 50 - R$ 100</option>
                        <option value="100-200">R$ 100 - R$ 200</option>
                        <option value="200+">R$ 200+</option>
                    </select>
                </div>

                <div class="filter-group">
                    <label for="stockFilter">📦 Estoque</label>
                    <select id="stockFilter" class="select-input">
                        <option value="">Todos</option>
                        <option value="high">Alto (50+)</option>
                        <option value="medium">Médio (10-49)</option>
                        <option value="low">Baixo (1-9)</option>
                        <option value="out">Sem estoque</option>
                    </select>
                </div>

                <button id="clearFilters" class="clear-btn">🗑️ Limpar</button>
            </div>
        </div>
```

```
        <div class="stats">
            <div class="stat-card">
                <div class="stat-number" id="totalProducts">0</div>
                <div class="stat-label">Total de Produtos</div>
            </div>
            <div class="stat-card">
                <div class="stat-number" id="visibleProducts">0</div>
                <div class="stat-label">Produtos Visíveis</div>
            </div>
            <div class="stat-card">
                <div class="stat-number" id="totalValue">R$ 0</div>
                <div class="stat-label">Valor Total</div>
            </div>
            <div class="stat-card">
                <div class="stat-number" id="avgPrice">R$ 0</div>
                <div class="stat-label">Preço Médio</div>
            </div>
        </div>

        <div id="productsContainer" class="products-grid"></div>

        <div id="noResults" class="no-results" style="display: none;">
            <h3>🔍 Nenhum produto encontrado</h3>
            <p>Tente ajustar os filtros para encontrar produtos</p>
        </div>
    </div>

    <script>
        // Base de dados dos produtos
        const products = [
            {
                id: 1,
                name: "Smartphone Galaxy Ultra",
                category: "Eletrônicos",
                price: 899.99,
                stock: 25,
                description: "Smartphone premium com câmera de 108MP e tela AMOLED de 6.8
polegadas"
            },
            {
                id: 2,
                name: "Notebook Gaming Pro",
                category: "Eletrônicos",
                price: 2499.99,
                stock: 8,
                description: "Notebook gamer com RTX 4060, Intel i7 e 16GB RAM para máxima
performance"
            },
            {
                id: 3,
                name: "Tênis Running Elite",
                category: "Esportes",
                price: 299.99,
                stock: 45,
                description: "Tênis de corrida com tecnologia de amortecimento avançada e design
ergonômico"
            },
            {
```

```
            id: 4,
            name: "Cafeteira Espresso Deluxe",
            category: "Casa",
            price: 449.99,
            stock: 12,
            description: "Cafeteira espresso automática com moedor integrado e sistema de
espuma"
        },
        {
            id: 5,
            name: "Headphone Wireless Pro",
            category: "Eletrônicos",
            price: 199.99,
            stock: 67,
            description: "Headphone sem fio com cancelamento de ruído ativo e bateria de 30
horas"
        },
        {
            id: 6,
            name: "Mesa de Escritório Premium",
            category: "Móveis",
            price: 799.99,
            stock: 3,
            description: "Mesa executiva com gavetas, suporte para monitor e acabamento em
madeira nobre"
        },
        {
            id: 7,
            name: "Kit Panelas Antiaderentes",
            category: "Casa",
            price: 89.99,
            stock: 89,
            description: "Conjunto de 5 panelas antiaderentes com revestimento cerâmico e
cabos ergonômicos"
        },
        {
            id: 8,
            name: "Bicicleta Mountain Bike",
            category: "Esportes",
            price: 1299.99,
            stock: 6,
            description: "Mountain bike com quadro de alumínio, suspensão dianteira e 21
marchas"
        },
        {
            id: 9,
            name: "Perfume Masculino Elegance",
            category: "Beleza",
            price: 79.99,
            stock: 156,
            description: "Perfume masculino com notas amadeiradas e cítricas, longa duração"
        },
        {
            id: 10,
            name: "Cadeira Gamer RGB",
            category: "Móveis",
            price: 599.99,
            stock: 0,
```

```
            description: "Cadeira gamer com iluminação RGB, apoio lombar e material premium"
        },
        {
            id: 11,
            name: "Smartwatch Fitness Pro",
            category: "Eletrônicos",
            price: 249.99,
            stock: 34,
            description: "Smartwatch com GPS, monitor cardíaco e resistência à água IP68"
        },
        {
            id: 12,
            name: "Conjunto de Maquiagem",
            category: "Beleza",
            price: 129.99,
            stock: 23,
            description: "Kit completo de maquiagem com paleta de sombras, batons e pincéis
profissionais"
        }
    ];

    let filteredProducts = [...products];

    // Elementos do DOM
    const searchInput = document.getElementById('searchInput');
    const categoryFilter = document.getElementById('categoryFilter');
    const priceFilter = document.getElementById('priceFilter');
    const stockFilter = document.getElementById('stockFilter');
    const clearFilters = document.getElementById('clearFilters');
    const productsContainer = document.getElementById('productsContainer');
    const noResults = document.getElementById('noResults');

    // Elementos de estatísticas
    const totalProducts = document.getElementById('totalProducts');
    const visibleProducts = document.getElementById('visibleProducts');
    const totalValue = document.getElementById('totalValue');
    const avgPrice = document.getElementById('avgPrice');

    // Inicialização
    function init() {
        populateCategories();
        renderProducts();
        updateStats();
        setupEventListeners();
    }

    // Popula o select de categorias
    function populateCategories() {
        const categories = [...new Set(products.map(p => p.category))];
        categories.forEach(category => {
            const option = document.createElement('option');
            option.value = category;
            option.textContent = category;
            categoryFilter.appendChild(option);
        });
    }

    // Configura os event listeners
```

```
        function setupEventListeners() {
            searchInput.addEventListener('input', debounce(applyFilters, 300));
            categoryFilter.addEventListener('change', applyFilters);
            priceFilter.addEventListener('change', applyFilters);
            stockFilter.addEventListener('change', applyFilters);
            clearFilters.addEventListener('click', clearAllFilters);
        }

        // Debounce para otimizar a busca
        function debounce(func, wait) {
            let timeout;
            return function executedFunction(...args) {
                const later = () => {
                    clearTimeout(timeout);
                    func(...args);
                };
                clearTimeout(timeout);
                timeout = setTimeout(later, wait);
            };
        }

        // Aplica os filtros
        function applyFilters() {
            const searchTerm = searchInput.value.toLowerCase();
            const selectedCategory = categoryFilter.value;
            const selectedPriceRange = priceFilter.value;
            const selectedStock = stockFilter.value;

            filteredProducts = products.filter(product => {
                // Filtro de busca
                const matchesSearch = product.name.toLowerCase().includes(searchTerm) ||
                                      product.description.toLowerCase().includes(searchTerm);

                // Filtro de categoria
                const matchesCategory = !selectedCategory || product.category ===
selectedCategory;

                // Filtro de preço
                let matchesPrice = true;
                if (selectedPriceRange) {
                    if (selectedPriceRange === '0-50') {
                        matchesPrice = product.price <= 50;
                    } else if (selectedPriceRange === '50-100') {
                        matchesPrice = product.price > 50 && product.price <= 100;
                    } else if (selectedPriceRange === '100-200') {
                        matchesPrice = product.price > 100 && product.price <= 200;
                    } else if (selectedPriceRange === '200+') {
                        matchesPrice = product.price > 200;
                    }
                }

                // Filtro de estoque
                let matchesStock = true;
                if (selectedStock) {
                    if (selectedStock === 'high') {
                        matchesStock = product.stock >= 50;
                    } else if (selectedStock === 'medium') {
                        matchesStock = product.stock >= 10 && product.stock < 50;
```

```
            } else if (selectedStock === 'low') {
                matchesStock = product.stock >= 1 && product.stock < 10;
            } else if (selectedStock === 'out') {
                matchesStock = product.stock === 0;
            }
        }

        return matchesSearch && matchesCategory && matchesPrice && matchesStock;
    });

    renderProducts();
    updateStats();
}

// Renderiza os produtos
function renderProducts() {
    productsContainer.innerHTML = '';

    if (filteredProducts.length === 0) {
        noResults.style.display = 'block';
        productsContainer.style.display = 'none';
        return;
    }

    noResults.style.display = 'none';
    productsContainer.style.display = 'grid';

    filteredProducts.forEach(product => {
        const productCard = createProductCard(product);
        productsContainer.appendChild(productCard);
    });
}

// Cria um card de produto
function createProductCard(product) {
    const card = document.createElement('div');
    card.className = 'product-card filter-animation';

    const stockLevel = getStockLevel(product.stock);
    const stockClass = getStockClass(product.stock);

    card.innerHTML = `
        <div class="product-header">
            <div>
                <div class="product-name">${product.name}</div>
                <div class="product-category">${product.category}</div>
            </div>
        </div>
        <div class="product-price">R$ ${product.price.toFixed(2)}</div>
        <div class="product-stock">
            <span class="stock-indicator ${stockClass}"></span>
            <span>Estoque: ${product.stock} unidades (${stockLevel})</span>
        </div>
        <div class="product-description">${product.description}</div>
        <div class="product-actions">
            <button class="btn btn-primary" ${product.stock === 0 ? 'disabled' : ''}>
                ${product.stock === 0 ? '❌ Indisponível' : '🛒 Comprar'}
            </button>
```

```
                <button class="btn btn-secondary">❤️ Favoritar</button>
            </div>
        `;

        return card;
    }

    // Determina o nível do estoque
    function getStockLevel(stock) {
        if (stock === 0) return 'Sem estoque';
        if (stock < 10) return 'Baixo';
        if (stock < 50) return 'Médio';
        return 'Alto';
    }

    // Determina a classe CSS do estoque
    function getStockClass(stock) {
        if (stock === 0) return 'stock-low';
        if (stock < 10) return 'stock-low';
        if (stock < 50) return 'stock-medium';
        return 'stock-high';
    }

    // Atualiza as estatísticas
    function updateStats() {
        const total = products.length;
        const visible = filteredProducts.length;
        const totalVal = filteredProducts.reduce((sum, p) => sum + (p.price * p.stock), 0);
        const avgPriceVal = visible > 0 ? filteredProducts.reduce((sum, p) => sum + p.price, 0)
/ visible : 0;

        totalProducts.textContent = total;
        visibleProducts.textContent = visible;
        totalValue.textContent = `R$ ${totalVal.toLocaleString('pt-BR', { minimumFractionDigits:
2 })}`;
        avgPrice.textContent = `R$ ${avgPriceVal.toFixed(2)}`;
    }

    // Limpa todos os filtros
    function clearAllFilters() {
        searchInput.value = '';
        categoryFilter.value = '';
        priceFilter.value = '';
        stockFilter.value = '';

        filteredProducts = [...products];
        renderProducts();
        updateStats();
    }

    // Inicializa a aplicação
    init();
  </script>
</body>
</html>
```

ANEXO G - CÓDIGO GERADO - MODELO C, ARTEFATO 2

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Filtro Dinâmico de Produtos</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background-color: #f4f4f4;
            color: #333;
        }

        .container {
            max-width: 900px;
            margin: 0 auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            color: #0056b3;
            margin-bottom: 30px;
        }

        .search-bar {
            margin-bottom: 20px;
            text-align: center;
        }

        .search-bar input[type="text"] {
            width: 70%;
            padding: 12px 15px;
            border: 1px solid #ddd;
            border-radius: 5px;
            font-size: 16px;
            box-sizing: border-box; /* Garante que o padding não aumente a largura total */
        }

        .search-bar input[type="text"]:focus {
            outline: none;
            border-color: #007bff;
            box-shadow: 0 0 5px rgba(0, 123, 255, 0.25);
        }

        .product-dashboard {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
            gap: 20px;
```

```
        }

        .product-card {
            background-color: #fff;
            border: 1px solid #eee;
            border-radius: 8px;
            box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
            padding: 15px;
            text-align: center;
            transition: transform 0.2s ease-in-out;
            display: flex; /* Para alinhar o conteúdo verticalmente */
            flex-direction: column;
            justify-content: space-between;
        }

        .product-card:hover {
            transform: translateY(-5px);
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);
        }

        .product-card h3 {
            color: #333;
            margin-top: 10px;
            margin-bottom: 5px;
            font-size: 1.2em;
        }

        .product-card p {
            color: #666;
            font-size: 0.9em;
            margin-bottom: 10px;
            flex-grow: 1; /* Permite que o parágrafo ocupe o espaço disponível */
        }

        .product-card .price {
            font-size: 1.1em;
            font-weight: bold;
            color: #28a745;
            margin-top: 10px;
        }

        /* Classes para mostrar/esconder itens */
        .product-card.hidden {
            display: none;
        }

        @media (max-width: 768px) {
            .search-bar input[type="text"] {
                width: 90%;
            }

            .product-dashboard {
                grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
            }
        }
    </style>
</head>
<body>
```

```
    <div class="container">
        <h1>Dashboard de Produtos</h1>

        <div class="search-bar">
            <input type="text" id="searchInput" placeholder="Buscar produtos por nome ou
categoria..." onkeyup="filterProducts()">
        </div>

        <div class="product-dashboard" id="productDashboard">
            <div class="product-card" data-name="Smartphone X" data-category="Eletrônicos">
                <h3>Smartphone X</h3>
                <p>Um smartphone de última geração com câmera de alta resolução.</p>
                <div class="price">R$ 2.500,00</div>
            </div>
            <div class="product-card" data-name="Notebook Ultra" data-category="Eletrônicos">
                <h3>Notebook Ultra</h3>
                <p>Potente notebook para trabalho e jogos, com processador rápido.</p>
                <div class="price">R$ 4.800,00</div>
            </div>
            <div class="product-card" data-name="Teclado Mecânico Pro"
data-category="Acessórios">
                <h3>Teclado Mecânico Pro</h3>
                <p>Teclado mecânico com iluminação RGB e switches duráveis.</p>
                <div class="price">R$ 350,00</div>
            </div>
            <div class="product-card" data-name="Mouse Gamer Óptico"
data-category="Acessórios">
                <h3>Mouse Gamer Óptico</h3>
                <p>Mouse de alta precisão para gamers, com botões programáveis.</p>
                <div class="price">R$ 180,00</div>
            </div>
            <div class="product-card" data-name="Smart TV 55 Polegadas"
data-category="Eletrônicos">
                <h3>Smart TV 55 Polegadas</h3>
                <p>TV inteligente com resolução 4K e acesso a streamings.</p>
                <div class="price">R$ 3.200,00</div>
            </div>
            <div class="product-card" data-name="Fone de Ouvido Bluetooth"
data-category="Áudio">
                <h3>Fone de Ouvido Bluetooth</h3>
                <p>Fones sem fio com cancelamento de ruído e bateria de longa duração.</p>
                <div class="price">R$ 450,00</div>
            </div>
            <div class="product-card" data-name="Câmera Digital Full HD"
data-category="Fotografia">
                <h3>Câmera Digital Full HD</h3>
                <p>Câmera compacta para fotos e vídeos de alta qualidade.</p>
                <div class="price">R$ 1.200,00</div>
            </div>
            <div class="product-card" data-name="Monitor Curvo 27 polegadas"
data-category="Eletrônicos">
                <h3>Monitor Curvo 27 polegadas</h3>
                <p>Monitor ideal para imersão em jogos e produtividade.</p>
                <div class="price">R$ 1.600,00</div>
            </div>
            <div class="product-card" data-name="Caixa de Som Portátil" data-category="Áudio">
                <h3>Caixa de Som Portátil</h3>
                <p>Caixa de som Bluetooth com som potente e bateria recarregável.</p>
```

```
            <div class="price">R$ 250,00</div>
        </div>
        <div class="product-card" data-name="Webcam HD com Microfone"
data-category="Acessórios">
            <h3>Webcam HD com Microfone</h3>
            <p>Webcam para videochamadas e streaming, com alta definição.</p>
            <div class="price">R$ 120,00</div>
        </div>
    </div>
</div>

<script>
    function filterProducts() {
        const searchInput = document.getElementById('searchInput');
        const filter = searchInput.value.toLowerCase();
        const productCards = document.querySelectorAll('.product-card');

        productCards.forEach(card => {
            const productName = card.getAttribute('data-name').toLowerCase();
            const productCategory = card.getAttribute('data-category').toLowerCase();
            const productDescription = card.querySelector('p').textContent.toLowerCase(); //
Adiciona a descrição para busca

            // Verifica se o termo de busca está no nome, categoria ou descrição do produto
            if (productName.includes(filter) || productCategory.includes(filter) ||
productDescription.includes(filter)) {
                card.classList.remove('hidden');
            } else {
                card.classList.add('hidden');
            }
        });
    }

    // Adiciona um listener para que o filtro seja executado também ao limpar o campo
    document.addEventListener('DOMContentLoaded', () => {
        const searchInput = document.getElementById('searchInput');
        searchInput.addEventListener('input', filterProducts); // 'input' é melhor que 'keyup' para
capturar colar de texto, etc.
    });
</script>
</body>
</html>
```

ANEXO H - CÓDIGO GERADO - MODELO D, ARTEFATO 2

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

```
    <title>Filtro Dinâmico de Produtos</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 1200px;
            margin: 0 auto;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        .filter-container {
            margin-bottom: 20px;
            text-align: center;
        }
        #filterInput {
            padding: 10px;
            width: 300px;
            font-size: 16px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        .dashboard {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
            gap: 20px;
        }
        .product-card {
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 5px rgba(0,0,0,0.1);
            text-align: center;
        }
        .product-card h3 {
            margin: 0 0 10px;
            color: #333;
        }
        .product-card p {
            margin: 5px 0;
            color: #666;
        }
        .product-card .price {
            color: #2ecc71;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Dashboard de Produtos</h1>
        <div class="filter-container">
            <input type="text" id="filterInput" placeholder="Filtrar produtos..."
```

```
onkeyup="filterProducts()">
        </div>
        <div class="dashboard" id="productList">
            <div class="product-card">
                <h3>Smartphone X</h3>
                <p>Categoria: Eletrônicos</p>
                <p class="price">R$ 2999,00</p>
            </div>
            <div class="product-card">
                <h3>Fone de Ouvido Bluetooth</h3>
                <p>Categoria: Acessórios</p>
                <p class="price">R$ 199,90</p>
            </div>
            <div class="product-card">
                <h3>Notebook Pro</h3>
                <p>Categoria: Eletrônicos</p>
                <p class="price">R$ 5999,00</p>
            </div>
            <div class="product-card">
                <h3>Carregador Portátil</h3>
                <p>Categoria: Acessórios</p>
                <p class="price">R$ 89,90</p>
            </div>
            <div class="product-card">
                <h3>Câmera Digital</h3>
                <p>Categoria: Eletrônicos</p>
                <p class="price">R$ 2499,00</p>
            </div>
        </div>
    </div>

    <script>
        function filterProducts() {
            const input = document.getElementById('filterInput').value.toLowerCase();
            const products = document.querySelectorAll('.product-card');

            products.forEach(product => {
                const title = product.querySelector('h3').textContent.toLowerCase();
                const category = product.querySelector('p').textContent.toLowerCase();

                if (title.includes(input) || category.includes(input)) {
                    product.style.display = '';
                } else {
                    product.style.display = 'none';
                }
            });
        }
    </script>
</body>
</html>
```

ANEXO I - CÓDIGO GERADO - MODELO A, ARTEFATO 3

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Upload de Imagem com Pré-visualização</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f9f9f9;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }

    .upload-container {
      background: #fff;
      padding: 20px 30px;
      border-radius: 12px;
      box-shadow: 0 4px 12px rgba(0,0,0,0.1);
      text-align: center;
      width: 100%;
      max-width: 400px;
    }

    .upload-container h2 {
      margin-bottom: 15px;
    }

    input[type="file"] {
      display: none;
    }

    .custom-file-upload {
      display: inline-block;
      padding: 10px 20px;
      background-color: #007bff;
      color: white;
      border-radius: 6px;
      cursor: pointer;
      margin-bottom: 15px;
      transition: background-color 0.3s;
    }

    .custom-file-upload:hover {
      background-color: #0056b3;
    }

    .preview {
      margin-top: 15px;
    }
```

```
    .preview img {
      max-width: 100%;
      height: auto;
      border-radius: 8px;
      margin-top: 10px;
    }

    .error {
      color: red;
      font-size: 0.9em;
      margin-top: 10px;
    }

    .success {
      color: green;
      font-size: 0.9em;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="upload-container">
    <h2>Enviar Imagem</h2>
    <label for="file-upload" class="custom-file-upload">Escolher Imagem</label>
    <input type="file" id="file-upload" accept="image/*"/>
    <div class="preview" id="preview"></div>
    <div class="error" id="error-message"></div>
    <div class="success" id="success-message"></div>
  </div>

  <script>
    const fileInput = document.getElementById('file-upload');
    const preview = document.getElementById('preview');
    const errorMsg = document.getElementById('error-message');
    const successMsg = document.getElementById('success-message');

    const MAX_SIZE_MB = 5;
    const VALID_TYPES = ['image/jpeg', 'image/png', 'image/gif', 'image/webp'];

    fileInput.addEventListener('change', function () {
      const file = this.files[0];
      preview.innerHTML = '';
      errorMsg.textContent = '';
      successMsg.textContent = '';

      if (!file) return;

      // Validação de tipo
      if (!VALID_TYPES.includes(file.type)) {
        errorMsg.textContent = 'Formato inválido. Use JPG, PNG, GIF ou WEBP.';
        return;
      }

      // Validação de tamanho
      const fileSizeMB = file.size / (1024 * 1024);
      if (fileSizeMB > MAX_SIZE_MB) {
        errorMsg.textContent = `A imagem é muito grande. Máximo permitido:
```

```
${MAX_SIZE_MB}MB.`;
        return;
      }

      // Mostrar pré-visualização
      const reader = new FileReader();
      reader.onload = function (e) {
        const img = document.createElement('img');
        img.src = e.target.result;
        preview.appendChild(img);
        successMsg.textContent = 'Imagem carregada com sucesso!';
      };
      reader.readAsDataURL(file);
    });
  </script>
</body>
</html>
```

ANEXO J - CÓDIGO GERADO - MODELO B, ARTEFATO 3

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Upload de Imagens com Pré-visualização</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            min-height: 100vh;
            padding: 20px;
        }

        .container {
            max-width: 800px;
            margin: 0 auto;
            background: white;
            border-radius: 20px;
            box-shadow: 0 20px 40px rgba(0,0,0,0.1);
            overflow: hidden;
        }

        .header {
```

```
    background: linear-gradient(135deg, #4facfe 0%, #00f2fe 100%);
    color: white;
    padding: 30px;
    text-align: center;
}

.header h1 {
    font-size: 2.5rem;
    margin-bottom: 10px;
    font-weight: 300;
}

.header p {
    opacity: 0.9;
    font-size: 1.1rem;
}

.upload-section {
    padding: 40px;
}

.upload-zone {
    border: 3px dashed #cbd5e0;
    border-radius: 15px;
    padding: 60px 20px;
    text-align: center;
    transition: all 0.3s ease;
    cursor: pointer;
    position: relative;
    background: #f8fafc;
}

.upload-zone:hover {
    border-color: #4facfe;
    background: #f0f9ff;
    transform: translateY(-2px);
}

.upload-zone.dragover {
    border-color: #00f2fe;
    background: #e0f8ff;
    transform: scale(1.02);
}

.upload-icon {
    font-size: 4rem;
    color: #a0aec0;
    margin-bottom: 20px;
}

.upload-text {
    font-size: 1.2rem;
    color: #4a5568;
    margin-bottom: 10px;
}

.upload-hint {
    color: #718096;
```

```
    font-size: 0.9rem;
}

#fileInput {
    display: none;
}

.file-info {
    margin-top: 30px;
    padding: 20px;
    background: #f7fafc;
    border-radius: 10px;
    border-left: 4px solid #4facfe;
}

.validation-message {
    padding: 15px;
    border-radius: 8px;
    margin: 20px 0;
    font-weight: 500;
}

.validation-success {
    background: #f0fff4;
    color: #276749;
    border: 1px solid #9ae6b4;
}

.validation-error {
    background: #fed7d7;
    color: #c53030;
    border: 1px solid #feb2b2;
}

.preview-container {
    margin-top: 30px;
}

.preview-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
    gap: 20px;
    margin-top: 20px;
}

.preview-item {
    position: relative;
    border-radius: 15px;
    overflow: hidden;
    background: white;
    box-shadow: 0 10px 25px rgba(0,0,0,0.1);
    transition: transform 0.3s ease;
}

.preview-item:hover {
    transform: translateY(-5px);
}
```

```
.preview-image {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

.preview-info {
    padding: 15px;
}

.preview-name {
    font-weight: 600;
    color: #2d3748;
    margin-bottom: 5px;
    font-size: 0.9rem;
    overflow: hidden;
    text-overflow: ellipsis;
    white-space: nowrap;
}

.preview-size {
    color: #718096;
    font-size: 0.8rem;
}

.remove-btn {
    position: absolute;
    top: 10px;
    right: 10px;
    background: rgba(239, 68, 68, 0.9);
    color: white;
    border: none;
    border-radius: 50%;
    width: 30px;
    height: 30px;
    cursor: pointer;
    font-size: 1.2rem;
    display: flex;
    align-items: center;
    justify-content: center;
    transition: all 0.3s ease;
}

.remove-btn:hover {
    background: rgba(239, 68, 68, 1);
    transform: scale(1.1);
}

.upload-btn {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    border: none;
    padding: 15px 40px;
    border-radius: 50px;
    font-size: 1.1rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.3s ease;
```

```
        margin-top: 20px;
        width: 100%;
    }

    .upload-btn:hover {
        transform: translateY(-2px);
        box-shadow: 0 10px 25px rgba(102, 126, 234, 0.3);
    }

    .upload-btn:disabled {
        background: #a0aec0;
        cursor: not-allowed;
        transform: none;
        box-shadow: none;
    }

    .progress-bar {
        width: 100%;
        height: 8px;
        background: #e2e8f0;
        border-radius: 4px;
        overflow: hidden;
        margin-top: 20px;
    }

    .progress-fill {
        height: 100%;
        background: linear-gradient(90deg, #4facfe 0%, #00f2fe 100%);
        width: 0%;
        transition: width 0.3s ease;
    }

    .stats {
        display: flex;
        justify-content: space-between;
        margin-top: 30px;
        padding: 20px;
        background: #f8fafc;
        border-radius: 10px;
    }

    .stat-item {
        text-align: center;
    }

    .stat-number {
        font-size: 2rem;
        font-weight: bold;
        color: #4facfe;
    }

    .stat-label {
        color: #718096;
        font-size: 0.9rem;
        margin-top: 5px;
    }

    @media (max-width: 768px) {
```

```
            .container {
                margin: 10px;
                border-radius: 15px;
            }

            .header {
                padding: 20px;
            }

            .header h1 {
                font-size: 2rem;
            }

            .upload-section {
                padding: 20px;
            }

            .upload-zone {
                padding: 40px 15px;
            }

            .preview-grid {
                grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
            }

            .stats {
                flex-direction: column;
                gap: 15px;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>📸 Upload de Imagens</h1>
            <p>Faça upload das suas imagens com pré-visualização e validação automática</p>
        </div>

        <div class="upload-section">
            <div class="upload-zone" id="uploadZone">
                <div class="upload-icon">☁️</div>
                <div class="upload-text">Clique aqui ou arraste suas imagens</div>
                <div class="upload-hint">Formatos aceitos: JPG, PNG, GIF, WebP (máx. 5MB por
arquivo)</div>
                <input type="file" id="fileInput" multiple accept="image/*">
            </div>

            <div id="validationMessage"></div>

            <div class="stats">
                <div class="stat-item">
                    <div class="stat-number" id="totalFiles">0</div>
                    <div class="stat-label">Arquivos</div>
                </div>
                <div class="stat-item">
                    <div class="stat-number" id="totalSize">0KB</div>
                    <div class="stat-label">Tamanho Total</div>
```

```
            </div>
            <div class="stat-item">
                <div class="stat-number" id="validFiles">0</div>
                <div class="stat-label">Válidos</div>
            </div>
        </div>

        <div class="preview-container">
            <div class="preview-grid" id="previewGrid"></div>
        </div>

        <button class="upload-btn" id="uploadBtn" disabled>
            🚀 Fazer Upload das Imagens
        </button>

        <div class="progress-bar" id="progressBar" style="display: none;">
            <div class="progress-fill" id="progressFill"></div>
        </div>
    </div>
</div>

<script>
    class ImageUploadSystem {
        constructor() {
            this.files = [];
            this.maxFileSize = 5 * 1024 * 1024; // 5MB
            this.allowedTypes = ['image/jpeg', 'image/jpg', 'image/png', 'image/gif', 'image/webp'];

            this.initializeElements();
            this.bindEvents();
        }

        initializeElements() {
            this.uploadZone = document.getElementById('uploadZone');
            this.fileInput = document.getElementById('fileInput');
            this.validationMessage = document.getElementById('validationMessage');
            this.previewGrid = document.getElementById('previewGrid');
            this.uploadBtn = document.getElementById('uploadBtn');
            this.progressBar = document.getElementById('progressBar');
            this.progressFill = document.getElementById('progressFill');

            // Elementos de estatísticas
            this.totalFilesEl = document.getElementById('totalFiles');
            this.totalSizeEl = document.getElementById('totalSize');
            this.validFilesEl = document.getElementById('validFiles');
        }

        bindEvents() {
            // Eventos de clique e seleção
            this.uploadZone.addEventListener('click', () => this.fileInput.click());
            this.fileInput.addEventListener('change', (e) => this.handleFileSelect(e));

            // Eventos de drag and drop
            this.uploadZone.addEventListener('dragover', (e) => this.handleDragOver(e));
            this.uploadZone.addEventListener('dragleave', (e) => this.handleDragLeave(e));
            this.uploadZone.addEventListener('drop', (e) => this.handleDrop(e));

            // Evento de upload
```

```
        this.uploadBtn.addEventListener('click', () => this.simulateUpload());
    }

    handleDragOver(e) {
        e.preventDefault();
        this.uploadZone.classList.add('dragover');
    }

    handleDragLeave(e) {
        e.preventDefault();
        this.uploadZone.classList.remove('dragover');
    }

    handleDrop(e) {
        e.preventDefault();
        this.uploadZone.classList.remove('dragover');
        const files = Array.from(e.dataTransfer.files);
        this.processFiles(files);
    }

    handleFileSelect(e) {
        const files = Array.from(e.target.files);
        this.processFiles(files);
    }

    processFiles(newFiles) {
        const validFiles = [];
        const invalidFiles = [];

        newFiles.forEach(file => {
            if (this.validateFile(file)) {
                // Verificar se o arquivo já existe
                const exists = this.files.some(existingFile =>
                    existingFile.name === file.name && existingFile.size === file.size
                );

                if (!exists) {
                    const fileObj = {
                        file: file,
                        name: file.name,
                        size: file.size,
                        type: file.type,
                        id: Date.now() + Math.random()
                    };
                    validFiles.push(fileObj);
                }
            } else {
                invalidFiles.push(file);
            }
        });

        this.files = [...this.files, ...validFiles];
        this.showValidationMessage(validFiles, invalidFiles);
        this.updatePreview();
        this.updateStats();
        this.updateUploadButton();
    }
```

```
        validateFile(file) {
            // Verificar tipo de arquivo
            if (!this.allowedTypes.includes(file.type)) {
                return false;
            }

            // Verificar tamanho
            if (file.size > this.maxFileSize) {
                return false;
            }

            return true;
        }

        showValidationMessage(validFiles, invalidFiles) {
            let message = '';
            let messageClass = '';

            if (validFiles.length > 0 && invalidFiles.length === 0) {
                message = `✅ ${validFiles.length} arquivo(s) adicionado(s) com sucesso!`;
                messageClass = 'validation-success';
            } else if (validFiles.length > 0 && invalidFiles.length > 0) {
                message = `⚠️ ${validFiles.length} arquivo(s) válido(s) adicionado(s).
${invalidFiles.length} arquivo(s) rejeitado(s) (formato ou tamanho inválido).`;
                messageClass = 'validation-error';
            } else if (invalidFiles.length > 0) {
                message = `❌ ${invalidFiles.length} arquivo(s) rejeitado(s). Verifique o formato
(JPG, PNG, GIF, WebP) e tamanho (máx. 5MB).`;
                messageClass = 'validation-error';
            }

            if (message) {
                this.validationMessage.innerHTML = `<div class="validation-message
${messageClass}">${message}</div>`;

                // Remover a mensagem após 5 segundos
                setTimeout(() => {
                    this.validationMessage.innerHTML = '';
                }, 5000);
            }
        }

        updatePreview() {
            this.previewGrid.innerHTML = '';

            this.files.forEach(fileObj => {
                const previewItem = this.createPreviewItem(fileObj);
                this.previewGrid.appendChild(previewItem);
            });
        }

        createPreviewItem(fileObj) {
            const item = document.createElement('div');
            item.className = 'preview-item';

            const img = document.createElement('img');
            img.className = 'preview-image';
            img.alt = fileObj.name;
```

```
        // Criar URL para pré-visualização
        const reader = new FileReader();
        reader.onload = (e) => {
            img.src = e.target.result;
        };
        reader.readAsDataURL(fileObj.file);

        const info = document.createElement('div');
        info.className = 'preview-info';

        const name = document.createElement('div');
        name.className = 'preview-name';
        name.textContent = fileObj.name;

        const size = document.createElement('div');
        size.className = 'preview-size';
        size.textContent = this.formatFileSize(fileObj.size);

        const removeBtn = document.createElement('button');
        removeBtn.className = 'remove-btn';
        removeBtn.innerHTML = '×';
        removeBtn.onclick = () => this.removeFile(fileObj.id);

        info.appendChild(name);
        info.appendChild(size);

        item.appendChild(img);
        item.appendChild(info);
        item.appendChild(removeBtn);

        return item;
    }

    removeFile(fileId) {
        this.files = this.files.filter(file => file.id !== fileId);
        this.updatePreview();
        this.updateStats();
        this.updateUploadButton();
    }

    updateStats() {
        const totalFiles = this.files.length;
        const totalSize = this.files.reduce((sum, file) => sum + file.size, 0);
        const validFiles = this.files.length; // Todos os arquivos na lista são válidos

        this.totalFilesEl.textContent = totalFiles;
        this.totalSizeEl.textContent = this.formatFileSize(totalSize);
        this.validFilesEl.textContent = validFiles;
    }

    updateUploadButton() {
        this.uploadBtn.disabled = this.files.length === 0;
    }

    formatFileSize(bytes) {
        if (bytes === 0) return '0KB';
```

```
            const k = 1024;
            const sizes = ['B', 'KB', 'MB', 'GB'];
            const i = Math.floor(Math.log(bytes) / Math.log(k));

            return parseFloat((bytes / Math.pow(k, i)).toFixed(1)) + ' ' + sizes[i];
        }

        simulateUpload() {
            if (this.files.length === 0) return;

            this.uploadBtn.disabled = true;
            this.uploadBtn.textContent = '📤 Fazendo Upload...';
            this.progressBar.style.display = 'block';

            let progress = 0;
            const interval = setInterval(() => {
                progress += Math.random() * 15;
                if (progress >= 100) {
                    progress = 100;
                    clearInterval(interval);
                    this.completeUpload();
                }
                this.progressFill.style.width = progress + '%';
            }, 200);
        }

        completeUpload() {
            setTimeout(() => {
                this.uploadBtn.textContent = '✅ Upload Concluído!';
                this.uploadBtn.style.background = 'linear-gradient(135deg, #48bb78 0%, #38a169
100%)';

                // Mostrar mensagem de sucesso
                this.validationMessage.innerHTML = `
                    <div class="validation-message validation-success">
                        🎉 ${this.files.length} arquivo(s) enviado(s) com sucesso!
                    </div>
                `;

                // Reset após 3 segundos
                setTimeout(() => {
                    this.resetUpload();
                }, 3000);
            }, 500);
        }

        resetUpload() {
            this.files = [];
            this.updatePreview();
            this.updateStats();
            this.progressBar.style.display = 'none';
            this.progressFill.style.width = '0%';
            this.uploadBtn.disabled = true;
            this.uploadBtn.textContent = '🚀 Fazer Upload das Imagens';
            this.uploadBtn.style.background = 'linear-gradient(135deg, #667eea 0%, #764ba2
100%)';
            this.validationMessage.innerHTML = '';
            this.fileInput.value = '';
```

```
        }
    }

    // Inicializar o sistema quando a página carregar
    document.addEventListener('DOMContentLoaded', () => {
        new ImageUploadSystem();
    });
  </script>
</body>
</html>
```

ANEXO K - CÓDIGO GERADO - MODELO C, ARTEFATO 3

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Upload de Imagem com Pré-visualização e Validação</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      background-color: #f4f4f4;
      margin: 0;
    }

    .container {
      background-color: #fff;
      padding: 30px;
      border-radius: 8px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
      text-align: center;
      width: 100%;
      max-width: 500px;
    }

    h1 {
      color: #333;
      margin-bottom: 25px;
    }

    .upload-area {
      border: 2px dashed #ccc;
      padding: 20px;
      border-radius: 8px;
      cursor: pointer;
```

```
    transition: border-color 0.3s ease;
    margin-bottom: 20px;
}

.upload-area:hover {
    border-color: #007bff;
}

.upload-area input[type="file"] {
    display: none;
}

.upload-area p {
    color: #777;
    margin: 0;
}

.image-preview {
    margin-top: 20px;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 10px;
    min-height: 150px;
    display: flex;
    justify-content: center;
    align-items: center;
    overflow: hidden; /* Garante que a imagem não saia do container */
    background-color: #eee;
}

.image-preview img {
    max-width: 100%;
    max-height: 150px; /* Limita a altura da pré-visualização */
    display: block;
    border-radius: 4px;
}

.image-preview p {
    color: #999;
    text-align: center;
}

.message {
    margin-top: 15px;
    padding: 10px;
    border-radius: 5px;
    font-weight: bold;
    display: none; /* Escondido por padrão */
}

.message.success {
    background-color: #d4edda;
    color: #155724;
    border: 1px solid #c3e6cb;
}

.message.error {
    background-color: #f8d7da;
```

```
            color: #721c24;
            border: 1px solid #f5c6cb;
        }

        .upload-button {
            background-color: #28a745;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
            margin-top: 20px;
            transition: background-color 0.3s ease;
        }

        .upload-button:hover {
            background-color: #218838;
        }

        .upload-button:disabled {
            background-color: #cccccc;
            cursor: not-allowed;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Upload de Imagem</h1>
        <div class="upload-area" id="uploadArea">
            <input type="file" id="imageUpload" accept="image/*">
            <p>Arraste e solte sua imagem aqui ou clique para selecionar</p>
        </div>

        <div class="image-preview" id="imagePreview">
            <p>Nenhuma imagem selecionada</p>
        </div>

        <div class="message" id="message"></div>

        <button class="upload-button" id="uploadButton" disabled>Fazer Upload</button>
    </div>

    <script>
        const imageUpload = document.getElementById('imageUpload');
        const uploadArea = document.getElementById('uploadArea');
        const imagePreview = document.getElementById('imagePreview');
        const messageElement = document.getElementById('message');
        const uploadButton = document.getElementById('uploadButton');

        const MAX_FILE_SIZE = 2 * 1024 * 1024; // 2MB em bytes

        // Função para exibir mensagem
        function showMessage(msg, type) {
            messageElement.textContent = msg;
            messageElement.className = `message ${type}`; // Adiciona a classe de tipo
(success ou error)
            messageElement.style.display = 'block';
```

```
        setTimeout(() => {
            messageElement.style.display = 'none';
        }, 3000); // Esconde a mensagem após 3 segundos
    }

    // Função para pré-visualizar a imagem
    function previewImage(file) {
        const reader = new FileReader();

        reader.onload = function(e) {
            imagePreview.innerHTML = `<img src="${e.target.result}" alt="Pré-visualização da
Imagem">`;
        };

        reader.onerror = function() {
            imagePreview.innerHTML = '<p>Erro ao carregar a imagem.</p>';
        };

        reader.readAsDataURL(file);
    }

    // Função para lidar com a seleção de arquivo
    function handleFile(file) {
        // Resetar mensagens e pré-visualização
        messageElement.style.display = 'none';
        imagePreview.innerHTML = '<p>Nenhuma imagem selecionada</p>';
        uploadButton.disabled = true;

        if (!file) {
            showMessage('Nenhum arquivo selecionado.', 'error');
            return;
        }

        // Validação de tipo de arquivo
        if (!file.type.startsWith('image/')) {
            showMessage('Por favor, selecione apenas arquivos de imagem (JPG, PNG, GIF,
etc.).', 'error');
            return;
        }

        // Validação de tamanho de arquivo
        if (file.size > MAX_FILE_SIZE) {
            showMessage(`O arquivo é muito grande. O tamanho máximo permitido é
${MAX_FILE_SIZE / (1024 * 1024)}MB.`, 'error');
            return;
        }

        // Tudo certo, pré-visualizar e habilitar botão de upload
        previewImage(file);
        showMessage('Imagem selecionada com sucesso. Clique em "Fazer Upload" para
enviar.', 'success');
        uploadButton.disabled = false;
    }

    // Evento para input de arquivo (clique)
    imageUpload.addEventListener('change', (event) => {
        const file = event.target.files[0];
        handleFile(file);
```

```
        });

        // Eventos para arrastar e soltar
        uploadArea.addEventListener('dragover', (event) => {
            event.preventDefault(); // Necessário para permitir o drop
            uploadArea.style.borderColor = '#007bff';
        });

        uploadArea.addEventListener('dragleave', () => {
            uploadArea.style.borderColor = '#ccc';
        });

        uploadArea.addEventListener('drop', (event) => {
            event.preventDefault();
            uploadArea.style.borderColor = '#ccc';
            const file = event.dataTransfer.files[0];
            imageUpload.files = event.dataTransfer.files; // Sincroniza o input file com o drop
            handleFile(file);
        });

        // Evento para o botão de upload (simulação)
        uploadButton.addEventListener('click', () => {
            const file = imageUpload.files[0];
            if (file) {
                // Aqui você faria a lógica real de upload, por exemplo, usando FormData e Fetch
API
                // Para este exemplo, apenas simulamos um upload bem-sucedido
                showMessage(`Upload da imagem "${file.name}" iniciado... (Simulação)`, 'success');

                // Simulação de delay para upload
                setTimeout(() => {
                    showMessage(`Upload da imagem "${file.name}" concluído com sucesso!`,
'success');
                    // Resetar formulário após upload (opcional)
                    imageUpload.value = ''; // Limpa o input file
                    imagePreview.innerHTML = '<p>Nenhuma imagem selecionada</p>';
                    uploadButton.disabled = true;
                }, 1500);
            } else {
                showMessage('Nenhuma imagem para fazer upload.', 'error');
            }
        });

        // Evento para abrir o seletor de arquivos ao clicar na área de upload
        uploadArea.addEventListener('click', () => {
            imageUpload.click();
        });
    </script>
</body>
</html>
```

ANEXO L - CÓDIGO GERADO - MODELO D, ARTEFATO 3

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Upload de Imagem com Pré-visualização</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            margin: 0;
            background-color: #f0f0f0;
        }
        .upload-container {
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            text-align: center;
        }
        .preview {
            margin-top: 20px;
            max-width: 300px;
            max-height: 300px;
            display: none;
        }
        .error {
            color: red;
            margin-top: 10px;
            display: none;
        }
        input[type="file"] {
            margin: 10px 0;
        }
        button {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="upload-container">
        <h2>Upload de Imagem</h2>
        <input type="file" id="imageInput" accept="image/*">
        <img id="previewImage" class="preview" alt="Pré-visualização">
        <p id="errorMessage" class="error"></p>
        <button onclick="uploadImage()">Enviar Imagem</button>
```

```
    </div>

    <script>
        const imageInput = document.getElementById('imageInput');
        const previewImage = document.getElementById('previewImage');
        const errorMessage = document.getElementById('errorMessage');

        imageInput.addEventListener('change', function(event) {
            const file = event.target.files[0];
            errorMessage.style.display = 'none';
            previewImage.style.display = 'none';

            if (file) {
                // Validação do tipo de arquivo
                if (!file.type.startsWith('image/')) {
                    showError('Por favor, selecione uma imagem válida (JPG, PNG, etc).');
                    return;
                }

                // Validação do tamanho do arquivo (máximo 5MB)
                const maxSize = 5 * 1024 * 1024; // 5MB em bytes
                if (file.size > maxSize) {
                    showError('A imagem deve ter no máximo 5MB.');
                    return;
                }

                // Pré-visualização da imagem
                const reader = new FileReader();
                reader.onload = function(e) {
                    previewImage.src = e.target.result;
                    previewImage.style.display = 'block';
                };
                reader.readAsDataURL(file);
            }
        });

        function showError(message) {
            errorMessage.textContent = message;
            errorMessage.style.display = 'block';
            imageInput.value = ''; // Limpa o input
            previewImage.style.display = 'none';
        }

        function uploadImage() {
            const file = imageInput.files[0];
            if (!file) {
                showError('Por favor, selecione uma imagem antes de enviar.');
                return;
            }

            // Simulação de upload (substitua por uma chamada real à API, se necessário)
            console.log('Imagem pronta para upload:', file.name);
            alert('Imagem enviada com sucesso! (Simulação)');
            imageInput.value = ''; // Limpa o input
            previewImage.style.display = 'none';
        }
    </script>
</body>
```

```
</html>
```

ANEXO M - CÓDIGO GERADO - MODELO A, ARTEFATO 4

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Editor de Código com Execução em Tempo Real</title>
<style>
 body {
   margin: 0; padding: 0; height: 100vh;
   display: flex; flex-direction: column;
   font-family: monospace, monospace;
   background: #1e1e1e; color: #ddd;
 }
 header {
   padding: 10px;
   background: #333;
   text-align: center;
   font-weight: bold;
   font-size: 1.2em;
   color: #eee;
 }
 #container {
   display: flex;
   flex: 1;
   overflow: hidden;
 }
 #editor-container {
   flex: 1;
   display: flex;
   flex-direction: column;
   border-right: 1px solid #444;
   background: #1e1e1e;
 }
 #highlighting {
   flex: 1;
   padding: 10px;
   overflow: auto;
   white-space: pre-wrap;
   word-wrap: break-word;
   outline: none;
   font-family: monospace;
   font-size: 14px;
   line-height: 1.4em;
   background: #1e1e1e;
   color: transparent; /* esconder texto para usar só o span colorido */
   caret-color: white;
```

```
    position: absolute;
    top: 0; left: 0; right: 0; bottom: 0;
    pointer-events: none; /* não interativo */
   }
   #code {
    flex: 1;
    padding: 10px;
    background: transparent;
    color: #eee;
    font-family: monospace;
    font-size: 14px;
    line-height: 1.4em;
    border: none;
    resize: none;
    outline: none;
    position: relative;
    z-index: 1;
    overflow: auto;
    white-space: pre-wrap;
    word-wrap: break-word;
   }
   #editor-wrapper {
    position: relative;
    flex: 1;
    height: 100%;
   }
   #output {
    flex: 1;
    background: #252525;
    color: #eee;
    padding: 10px;
    overflow: auto;
    font-family: monospace;
    font-size: 14px;
    border-left: 1px solid #444;
   }
   /* Destaque básico de sintaxe JS */
   .keyword { color: #569cd6; font-weight: bold; }
   .string { color: #d69d85; }
   .number { color: #b5cea8; }
   .comment { color: #6a9955; font-style: italic; }
   .operator { color: #d4d4d4; }
   .function { color: #dcdcaa; }
  </style>
  </head>
  <body>

  <header>Editor JS com Sintaxe e Execução em Tempo Real</header>

  <div id="container">
    <div id="editor-container">
      <div id="editor-wrapper">
        <pre id="highlighting"></pre>
        <textarea id="code" spellcheck="false" autocomplete="off" autocorrect="off"
  autocapitalize="off"></textarea>
      </div>
    </div>
    <div id="output" tabindex="0"></div>
```

```
</div>

<script>
 // Elementos
 const code = document.getElementById('code');
 const highlighting = document.getElementById('highlighting');
 const output = document.getElementById('output');

 // Palavras-chave JS para destaque
 const keywords = [
   "break", "case", "catch", "class", "const", "continue", "debugger",
   "default", "delete", "do", "else", "export", "extends", "finally", "for",
   "function", "if", "import", "in", "instanceof", "let", "new", "return",
   "super", "switch", "this", "throw", "try", "typeof", "var", "void",
   "while", "with", "yield", "async", "await", "static"
 ];

 // Função para escapar HTML
 function escapeHtml(str) {
   return str.replace(/&/g, '&amp;').replace(/</g, '&lt;').replace(/>/g, '&gt;');
 }

 // Função para aplicar o destaque de sintaxe
 function highlight(codeText) {
   if (!codeText) return '';

   // Escape inicial para HTML
   let html = escapeHtml(codeText);

   // Comentários multilinha e linha única
   html = html.replace(/(\/\*[\s\S]*?\*\/)|(\/\/.*$)/gm, match => {
     return `<span class="comment">${escapeHtml(match)}</span>`;
   });

   // Strings (aspas simples, duplas e template literals)
   html = html.replace(/(["'`])((?:\\.|(?!\1).)*)(\1)/g, (m, quote, content, quote2) => {
     return `<span class="string">${escapeHtml(m)}</span>`;
   });

   // Números
   html = html.replace(/\b(\d+(\.\d+)?|0x[a-fA-F0-9]+)\b/g, '<span class="number">$1</span>');

   // Palavras-chave
   const kwRegex = new RegExp('\\b(' + keywords.join('|') + ')\\b', 'g');
   html = html.replace(kwRegex, '<span class="keyword">$1</span>');

   // Funções (palavras seguidas de parênteses)
   html = html.replace(/\b([a-zA-Z_]\w*)(?=\s*\()/g, '<span class="function">$1</span>');

   // Operadores básicos
   html = html.replace(/([=+\-*\/%<>!&|^~?:]+)/g, '<span class="operator">$1</span>');

   return html;
 }

 // Sincronizar scroll
 function syncScroll() {
   highlighting.scrollTop = code.scrollTop;
```

```
    highlighting.scrollLeft = code.scrollLeft;
  }

  // Executar código e mostrar resultado
  function executeCode(jsCode) {
    // Limpa output
    output.textContent = '';

    try {
      // Capturar console.log
      const logs = [];
      const fakeConsole = {
        log: (...args) => {
          logs.push(args.map(a => {
            if(typeof a === 'object') return JSON.stringify(a);
            return a;
          }).join(' '));
        }
      };

      // Criar função com console substituído
      const func = new Function('console', jsCode);
      func(fakeConsole);

      output.textContent = logs.join('\n');
    } catch (e) {
      output.textContent = 'Erro: ' + e.message;
    }
  }

  // Atualiza tudo: highlight + execução
  function update() {
    const val = code.value;
    highlighting.innerHTML = highlight(val) + '<br />'; // <br> para manter última linha visível
    executeCode(val);
  }

  // Evento input do textarea
  code.addEventListener('input', () => {
    update();
  });

  // Sincronizar scroll
  code.addEventListener('scroll', () => {
    syncScroll();
  });

  // Inicializa com um código exemplo
  code.value = `// Experimente seu código JS aqui\nconsole.log("Olá, mundo!");\n\nfunction
soma(a, b) {\n  return a + b;\n}\n\nconsole.log(soma(2, 3));`;
  update();
</script>

</body>
</html>
```

ANEXO N - CÓDIGO GERADO - MODELO B, ARTEFATO 4

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Editor de Código - Sintaxe Destacada</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Courier New', monospace;
            background: #1e1e1e;
            color: #d4d4d4;
            height: 100vh;
            display: flex;
            flex-direction: column;
        }

        .header {
            background: #2d2d30;
            padding: 12px 20px;
            border-bottom: 1px solid #3e3e42;
            display: flex;
            align-items: center;
            gap: 15px;
        }

        .header h1 {
            color: #fff;
            font-size: 18px;
            font-weight: 600;
        }

        .language-selector {
            background: #3c3c3c;
            border: 1px solid #5a5a5a;
            color: #fff;
            padding: 6px 12px;
            border-radius: 4px;
            font-size: 12px;
        }

        .run-button {
            background: #0e639c;
            color: white;
            border: none;
```

```
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    font-size: 12px;
    font-weight: 600;
    transition: background 0.2s;
}

.run-button:hover {
    background: #1177bb;
}

.main-container {
    display: flex;
    flex: 1;
    overflow: hidden;
}

.editor-container {
    flex: 1;
    display: flex;
    flex-direction: column;
    border-right: 1px solid #3e3e42;
}

.editor-wrapper {
    position: relative;
    flex: 1;
    overflow: hidden;
}

.line-numbers {
    position: absolute;
    left: 0;
    top: 0;
    width: 60px;
    height: 100%;
    background: #252526;
    border-right: 1px solid #3e3e42;
    padding: 20px 8px;
    font-size: 14px;
    line-height: 1.5;
    color: #858585;
    overflow: hidden;
    user-select: none;
}

.code-editor {
    position: absolute;
    left: 60px;
    top: 0;
    right: 0;
    bottom: 0;
    padding: 20px;
    font-size: 14px;
    line-height: 1.5;
    background: transparent;
    border: none;
```

```
    outline: none;
    resize: none;
    color: #d4d4d4;
    font-family: 'Courier New', monospace;
    tab-size: 4;
    white-space: pre;
    overflow: auto;
}

.syntax-highlight {
    position: absolute;
    left: 60px;
    top: 0;
    right: 0;
    bottom: 0;
    padding: 20px;
    font-size: 14px;
    line-height: 1.5;
    font-family: 'Courier New', monospace;
    white-space: pre;
    pointer-events: none;
    overflow: auto;
    tab-size: 4;
}

.output-container {
    width: 40%;
    display: flex;
    flex-direction: column;
    background: #1e1e1e;
}

.output-header {
    background: #2d2d30;
    padding: 12px 20px;
    border-bottom: 1px solid #3e3e42;
    font-size: 14px;
    font-weight: 600;
    color: #fff;
}

.output-content {
    flex: 1;
    padding: 20px;
    overflow: auto;
    font-size: 14px;
    line-height: 1.5;
}

.console-output {
    background: #0d1117;
    border: 1px solid #30363d;
    border-radius: 6px;
    padding: 15px;
    min-height: 200px;
    white-space: pre-wrap;
    font-family: 'Courier New', monospace;
}
```

```
        .error {
            color: #f85149;
        }

        .success {
            color: #56d364;
        }

        .log {
            color: #79c0ff;
        }

        /* Syntax highlighting colors */
        .keyword { color: #569cd6; }
        .string { color: #ce9178; }
        .number { color: #b5cea8; }
        .comment { color: #6a9955; font-style: italic; }
        .function { color: #dcdcaa; }
        .operator { color: #d4d4d4; }
        .punctuation { color: #d4d4d4; }
        .variable { color: #9cdcfe; }
        .property { color: #92c5f7; }
        .tag { color: #569cd6; }
        .attribute { color: #92c5f7; }
        .value { color: #ce9178; }

        @media (max-width: 768px) {
            .main-container {
                flex-direction: column;
            }

            .output-container {
                width: 100%;
                height: 40%;
            }
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>🚀 Editor de Código</h1>
        <select class="language-selector" id="languageSelect">
            <option value="javascript">JavaScript</option>
            <option value="html">HTML</option>
            <option value="css">CSS</option>
            <option value="python">Python</option>
        </select>
        <button class="run-button" id="runButton">▶ Executar</button>
    </div>

    <div class="main-container">
        <div class="editor-container">
            <div class="editor-wrapper">
                <div class="line-numbers" id="lineNumbers">1</div>
                <div class="syntax-highlight" id="syntaxHighlight"></div>
                <textarea class="code-editor" id="codeEditor" placeholder="Digite seu código
aqui...">// Exemplo em JavaScript
```

```
function saudacao(nome) {
    return `Olá, ${nome}! Bem-vindo ao editor.`;
}

console.log(saudacao("Desenvolvedor"));

// Teste com números
const numeros = [1, 2, 3, 4, 5];
const dobrados = numeros.map(n => n * 2);
console.log("Números dobrados:", dobrados);

// Exemplo de função recursiva
function fibonacci(n) {
    if (n <= 1) return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

console.log("Fibonacci(8):", fibonacci(8));</textarea>
            </div>
        </div>

        <div class="output-container">
            <div class="output-header">💻 Saída do Código</div>
            <div class="output-content">
                <div class="console-output" id="consoleOutput">Clique em "Executar" para ver o
resultado do seu código...</div>
            </div>
        </div>
    </div>

    <script>
        class CodeEditor {
            constructor() {
                this.editor = document.getElementById('codeEditor');
                this.highlight = document.getElementById('syntaxHighlight');
                this.lineNumbers = document.getElementById('lineNumbers');
                this.output = document.getElementById('consoleOutput');
                this.languageSelect = document.getElementById('languageSelect');
                this.runButton = document.getElementById('runButton');

                this.currentLanguage = 'javascript';
                this.setupEventListeners();
                this.updateHighlight();
                this.updateLineNumbers();
            }

            setupEventListeners() {
                this.editor.addEventListener('input', () => {
                    this.updateHighlight();
                    this.updateLineNumbers();
                });

                this.editor.addEventListener('scroll', () => {
                    this.highlight.scrollTop = this.editor.scrollTop;
                    this.highlight.scrollLeft = this.editor.scrollLeft;
                    this.lineNumbers.scrollTop = this.editor.scrollTop;
                });
```

```
        this.editor.addEventListener('keydown', (e) => {
            if (e.key === 'Tab') {
                e.preventDefault();
                this.insertTab();
            }
        });

        this.languageSelect.addEventListener('change', (e) => {
            this.currentLanguage = e.target.value;
            this.updateHighlight();
            this.updateExampleCode();
        });

        this.runButton.addEventListener('click', () => {
            this.executeCode();
        });
    }

    insertTab() {
        const start = this.editor.selectionStart;
        const end = this.editor.selectionEnd;
        const value = this.editor.value;

        this.editor.value = value.substring(0, start) + '    ' + value.substring(end);
        this.editor.selectionStart = this.editor.selectionEnd = start + 4;

        this.updateHighlight();
    }

    updateLineNumbers() {
        const lines = this.editor.value.split('\n').length;
        let lineNumbersText = '';

        for (let i = 1; i <= lines; i++) {
            lineNumbersText += i + '\n';
        }

        this.lineNumbers.textContent = lineNumbersText;
    }

    updateHighlight() {
        const code = this.editor.value;
        let highlightedCode = '';

        switch (this.currentLanguage) {
            case 'javascript':
                highlightedCode = this.highlightJavaScript(code);
                break;
            case 'html':
                highlightedCode = this.highlightHTML(code);
                break;
            case 'css':
                highlightedCode = this.highlightCSS(code);
                break;
            case 'python':
                highlightedCode = this.highlightPython(code);
                break;
            default:
```

```
                highlightedCode = this.escapeHtml(code);
            }

            this.highlight.innerHTML = highlightedCode;
        }

        highlightJavaScript(code) {
            return code
                .replace(/\/\/.*$/gm, '<span class="comment">$&</span>')
                .replace(/\/\*[\s\S]*?\*\//g, '<span class="comment">$&</span>')

.replace(/\b(function|var|let|const|if|else|for|while|do|switch|case|break|continue|return|try|catch|
finally|throw|new|this|class|extends|import|export|default|async|await)\b/g, '<span
class="keyword">$&</span>')

.replace(/\b(console|document|window|Array|Object|String|Number|Boolean|Date|Math|JSON)\
b/g, '<span class="variable">$&</span>')
                .replace(/(["'`])((?:\\.|(?!\1)[^\\])*?)\1/g, '<span class="string">$&</span>')
                .replace(/\b(\d+\.?\d*)\b/g, '<span class="number">$&</span>')
                .replace(/\b([a-zA-Z_$][a-zA-Z0-9_$]*)\s*(?=\()/g, '<span
class="function">$&</span>');
        }

        highlightHTML(code) {
            return code
                .replace(/<!--[\s\S]*?-->/g, '<span class="comment">$&</span>')
                .replace(/<\/?([a-zA-Z][a-zA-Z0-9]*)\b[^>]*>/g, (match, tagName) => {
                    return match
                        .replace(/^<\/?[a-zA-Z][a-zA-Z0-9]*/, '<span class="tag">$&</span>')
                        .replace(/\s+([a-zA-Z-]+)=/g, ' <span class="attribute">$1</span>=')
                        .replace(/=(["'`])(.*?)\1/g, '=<span class="value">$1$2$1</span>');
                });
        }

        highlightCSS(code) {
            return code
                .replace(/\/\*[\s\S]*?\*\//g, '<span class="comment">$&</span>')
                .replace(/([.#]?[a-zA-Z-]+)\s*{/g, '<span class="tag">$1</span> {')
                .replace(/([a-zA-Z-]+)\s*:/g, '<span class="property">$1</span>:')
                .replace(/:([^;{}]+);/g, ': <span class="value">$1</span>;')
                .replace(/(["'`])((?:\\.|(?!\1)[^\\])*?)\1/g, '<span class="string">$&</span>');
        }

        highlightPython(code) {
            return code
                .replace(/#.*$/gm, '<span class="comment">$&</span>')

.replace(/\b(def|class|if|elif|else|for|while|try|except|finally|with|as|import|from|return|yield|break|
continue|pass|and|or|not|in|is|lambda|global|nonlocal)\b/g, '<span
class="keyword">$&</span>')
                .replace(/\b(print|len|range|str|int|float|list|dict|tuple|set)\b/g, '<span
class="function">$&</span>')
                .replace(/(["'`])((?:\\.|(?!\1)[^\\])*?)\1/g, '<span class="string">$&</span>')
                .replace(/\b(\d+\.?\d*)\b/g, '<span class="number">$&</span>');
        }

        escapeHtml(text) {
            const div = document.createElement('div');
```

```
            div.textContent = text;
            return div.innerHTML;
        }

        updateExampleCode() {
            const examples = {
                javascript: `// Exemplo em JavaScript
function saudacao(nome) {
    return \`Olá, \${nome}! Bem-vindo ao editor.\`;
}

console.log(saudacao("Desenvolvedor"));

// Teste com números
const numeros = [1, 2, 3, 4, 5];
const dobrados = numeros.map(n => n * 2);
console.log("Números dobrados:", dobrados);`,

                html: `<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Minha Página</title>
</head>
<body>
    <h1>Olá, Mundo!</h1>
    <p>Este é um exemplo de HTML.</p>
    <button onclick="alert('Botão clicado!')">Clique aqui</button>
</body>
</html>`,

                css: `/* Exemplo de CSS */
body {
    font-family: Arial, sans-serif;
    background: linear-gradient(45deg, #ff6b6b, #4ecdc4);
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
}

h1 {
    color: #333;
    text-align: center;
}`,

                python: `# Exemplo em Python
def saudacao(nome):
    return f"Olá, {nome}! Bem-vindo ao editor."

print(saudacao("Desenvolvedor"))
```

```
# Lista de números
numeros = [1, 2, 3, 4, 5]
dobrados = [n * 2 for n in numeros]
print("Números dobrados:", dobrados)

# Função recursiva
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

print("Fibonacci(8):", fibonacci(8))`
            };

            this.editor.value = examples[this.currentLanguage] || '';
            this.updateHighlight();
            this.updateLineNumbers();
        }

        executeCode() {
            const code = this.editor.value;

            if (this.currentLanguage === 'javascript') {
                this.executeJavaScript(code);
            } else if (this.currentLanguage === 'html') {
                this.executeHTML(code);
            } else {
                this.output.innerHTML = `<span class="error">Execução disponível apenas para
JavaScript e HTML no momento.</span>`;
            }
        }

        executeJavaScript(code) {
            this.output.innerHTML = '';

            // Redirecionar console.log
            const originalLog = console.log;
            const originalError = console.error;
            const originalWarn = console.warn;

            const logs = [];

            console.log = (...args) => {
                logs.push({ type: 'log', content: args.map(String).join(' ') });
            };

            console.error = (...args) => {
                logs.push({ type: 'error', content: args.map(String).join(' ') });
            };

            console.warn = (...args) => {
                logs.push({ type: 'warn', content: args.map(String).join(' ') });
            };

            try {
                // Executar o código
                eval(code);
```

```
                // Mostrar logs
                if (logs.length === 0) {
                    this.output.innerHTML = '<span class="success">Código executado com
sucesso! (Sem saída)</span>';
                } else {
                    let output = '';
                    logs.forEach(log => {
                        const className = log.type === 'error' ? 'error' : log.type === 'warn' ? 'error' :
'log';
                        output += `<span
class="${className}">${this.escapeHtml(log.content)}</span>\n`;
                    });
                    this.output.innerHTML = output;
                }

            } catch (error) {
                this.output.innerHTML = `<span class="error">Erro:
${this.escapeHtml(error.message)}</span>`;
            } finally {
                // Restaurar console original
                console.log = originalLog;
                console.error = originalError;
                console.warn = originalWarn;
            }
        }

        executeHTML(code) {
            // Criar iframe para renderizar HTML
            const iframe = document.createElement('iframe');
            iframe.style.width = '100%';
            iframe.style.height = '300px';
            iframe.style.border = '1px solid #3e3e42';
            iframe.style.borderRadius = '4px';
            iframe.style.background = 'white';

            this.output.innerHTML = '';
            this.output.appendChild(iframe);

            // Escrever HTML no iframe
            const iframeDoc = iframe.contentDocument || iframe.contentWindow.document;
            iframeDoc.open();
            iframeDoc.write(code);
            iframeDoc.close();
        }
    }

    // Inicializar o editor quando a página carregar
    document.addEventListener('DOMContentLoaded', () => {
        new CodeEditor();
    });
  </script>
</body>
</html>
```

ANEXO O - CÓDIGO GERADO - MODELO C, ARTEFATO 4

```
!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Editor de Código com Destaque e Execução</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            display: flex;
            flex-direction: column;
            height: 100vh;
            margin: 0;
            background-color: #f0f2f5;
            color: #333;
        }

        header {
            background-color: #282c34;
            color: #fff;
            padding: 15px 20px;
            text-align: center;
            font-size: 1.5em;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
        }

        .container {
            display: flex;
            flex: 1;
            padding: 20px;
            gap: 20px;
        }

        .panel {
            flex: 1;
            display: flex;
            flex-direction: column;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
            overflow: hidden;
        }

        .panel-header {
            background-color: #e0e0e0;
            padding: 10px 15px;
            font-weight: bold;
            border-bottom: 1px solid #ccc;
            color: #555;
        }

        .code-editor, .output-panel {
            flex: 1;
```

```
    padding: 15px;
    box-sizing: border-box;
    font-family: 'Fira Code', 'Cascadia Code', 'Consolas', 'Monaco', monospace;
    font-size: 0.95em;
    line-height: 1.5;
    white-space: pre-wrap;
    overflow: auto;
    tab-size: 4;
    -moz-tab-size: 4;
    border: none;
    outline: none;
}

.code-editor {
    resize: none;
    background-color: #282c34; /* Fundo do editor */
    color: #abb2bf; /* Cor do texto padrão */
    caret-color: #528bff; /* Cor do cursor */
    position: relative; /* Para o overlay de destaque */
}

.code-textarea {
    width: 100%;
    height: 100%;
    background: transparent;
    color: transparent; /* Torna o texto invisível */
    caret-color: #fff; /* Torna o cursor visível */
    border: none;
    padding: 0;
    margin: 0;
    resize: none;
    outline: none;
    font-family: inherit;
    font-size: inherit;
    line-height: inherit;
    position: absolute;
    top: 0;
    left: 0;
    overflow: auto; /* Para a barra de rolagem funcionar */
    z-index: 2; /* Acima do destaque */
}

.highlighted-code {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    padding: 0;
    margin: 0;
    pointer-events: none; /* Não interfere com o textarea */
    z-index: 1; /* Abaixo do textarea */
    background-color: #282c34; /* Garante que o fundo seja igual ao textarea */
    color: #abb2bf;
}

.token {
    /* Estilos de destaque de sintaxe */
```

```
        }

        .keyword { color: #c678dd; } /* roxo */
        .operator { color: #56b6c2; } /* azul claro/ciano */
        .string { color: #98c379; } /* verde */
        .number { color: #d19a66; } /* laranja */
        .comment { color: #5c6370; font-style: italic; } /* cinza */
        .variable { color: #e06c75; } /* vermelho */
        .function { color: #61afef; } /* azul */
        .punctuation { color: #abb2bf; } /* cinza claro */
        .builtin { color: #e5c07b; } /* amarelo */

        .output-panel {
            background-color: #f7f7f7;
            border-top: 1px solid #eee;
            color: #333;
            white-space: pre-wrap; /* Mantém quebras de linha e espaços */
        }

        .output-panel code {
            display: block; /* Garante que cada linha seja um bloco */
            padding: 2px 0;
        }

        .error {
            color: #e74c3c;
            font-weight: bold;
        }

        .success {
            color: #2ecc71;
            font-weight: bold;
        }

        footer {
            background-color: #282c34;
            color: #fff;
            padding: 10px 20px;
            text-align: center;
            font-size: 0.9em;
            box-shadow: 0 -2px 5px rgba(0, 0, 0, 0.2);
        }
    </style>
</head>
<body>
    <header>
        <h1>Editor de Código Online</h1>
    </header>

    <div class="container">
        <div class="panel">
            <div class="panel-header">Código (JavaScript)</div>
            <div class="code-editor">
                <div class="highlighted-code"></div>
                <textarea id="codeTextarea" class="code-textarea" spellcheck="false"
autocorrect="off" autocapitalize="off"></textarea>
            </div>
        </div>
```

```
        <div class="panel">
            <div class="panel-header">Saída</div>
            <pre id="output" class="output-panel"></pre>
        </div>
    </div>

    <footer>
        <p>&copy; 2023 Editor Simples de Código. Desenvolvido com HTML, CSS e
JavaScript.</p>
    </footer>

    <script>
        const codeTextarea = document.getElementById('codeTextarea');
        const highlightedCode = document.querySelector('.highlighted-code');
        const outputPanel = document.getElementById('output');

        // Função de escape HTML para segurança na saída
        function escapeHtml(text) {
            return text.replace(/&/g, "&amp;")
                       .replace(/</g, "&lt;")
                       .replace(/>/g, "&gt;")
                       .replace(/"/g, "&quot;")
                       .replace(/'/g, "&#039;");
        }

        // Função para destaque de sintaxe
        function highlightSyntax(code) {
            // Regras de destaque (mais simples para JavaScript)
            const keywords =
/\b(function|var|let|const|if|else|for|while|do|switch|case|break|continue|return|try|catch|finally|th
row|new|this|super|class|extends|import|export|await|async)\b/g;
            const operators = /(\+|-|\*|\/|=|<|>|!|&|\||\?|:|%|\^|\~)/g;
            const strings = /("(.*?)"|'(.*?)'|`(.*?)`)/g;
            const numbers = /\b(\d+(\.\d+)?)\b/g;
            const comments = /(\/\/.*|\/\*[\s\S]*?\*\/)/g;
            const variables = /\b([a-zA-Z_$][0-9a-zA-Z_$]*)\b(?![\(])/g; // Variáveis que não são
seguidas por (
            const functions = /\b([a-zA-Z_$][0-9a-zA-Z_$]*)\b(?=\()/g; // Funções seguidas por (
            const punctuation = /([{}()[].,;])/g;
            const builtin =
/\b(console|document|window|alert|setTimeout|setInterval|clearTimeout|clearInterval)\b/g;


            let highlighted = escapeHtml(code);

            // A ordem importa para evitar sobreposição errada
            highlighted = highlighted.replace(comments, '<span class="comment">$&</span>');
            highlighted = highlighted.replace(strings, '<span class="string">$&</span>');
            highlighted = highlighted.replace(numbers, '<span class="number">$&</span>');
            highlighted = highlighted.replace(keywords, '<span class="keyword">$&</span>');
            highlighted = highlighted.replace(operators, '<span class="operator">$&</span>');
            highlighted = highlighted.replace(builtin, '<span class="builtin">$&</span>');
            highlighted = highlighted.replace(functions, '<span class="function">$&</span>');
            highlighted = highlighted.replace(punctuation, '<span class="punctuation">$&</span>');
            // A regex de variável precisa ser mais cuidadosa para não pegar palavras já
destacadas
            // Uma abordagem mais robusta para isso seria processar token a token, mas para
esta demonstração
```

```
        // vamos considerar que as outras regexes já "consumiram" as palavras-chave,
funções etc.
        // Para simplificar, não faremos uma regex para 'variable' para evitar conflitos diretos e
focar nas mais importantes.

        return highlighted;
    }

    // Função para executar o código
    function executeCode() {
        const code = codeTextarea.value;
        outputPanel.innerHTML = ''; // Limpa a saída anterior

        // Captura logs do console
        const originalLog = console.log;
        const originalError = console.error;
        const originalWarn = console.warn;
        const originalInfo = console.info;

        console.log = (...args) => {
            outputPanel.innerHTML += `<div
class="log-message">${args.map(escapeHtml).join(' ')}</div>`;
        };
        console.error = (...args) => {
            outputPanel.innerHTML += `<div class="error">${args.map(escapeHtml).join('
')}</div>`;
        };
        console.warn = (...args) => {
            outputPanel.innerHTML += `<div
class="warn-message">${args.map(escapeHtml).join(' ')}</div>`;
        };
        console.info = (...args) => {
            outputPanel.innerHTML += `<div
class="info-message">${args.map(escapeHtml).join(' ')}</div>`;
        };

        try {
            // Usa um wrapper para evitar que variáveis globais do script vazem para o escopo
global do navegador
            // e para simular um ambiente de execução mais controlado.
            // A função setTimeout é usada para evitar bloquear o thread principal do navegador
para códigos muito longos.
            // No entanto, para fins de demonstração, a execução é síncrona aqui.
            // Para uma execução realmente assíncrona/segura, seria necessário um Web
Worker ou um iframe com sandboxing.
            new Function(code)();
            if (outputPanel.innerHTML === '') {
                outputPanel.innerHTML = '<span class="success">Execução concluída sem
saída.</span>';
            }
        } catch (error) {
            outputPanel.innerHTML += `<div class="error">Erro:
${escapeHtml(error.message)}</div>`;
        } finally {
            // Restaura o console
            console.log = originalLog;
            console.error = originalError;
            console.warn = originalWarn;
```

```
            console.info = originalInfo;
        }
    }

    // Sincroniza a rolagem entre o textarea e o div de destaque
    function syncScroll() {
        highlightedCode.scrollTop = codeTextarea.scrollTop;
        highlightedCode.scrollLeft = codeTextarea.scrollLeft;
    }

    // Atualiza o destaque e executa o código
    function updateEditor() {
        const code = codeTextarea.value;
        highlightedCode.innerHTML = highlightSyntax(code);
        syncScroll();
        executeCode(); // Executa o código em tempo real
    }

    // Adiciona um listener para a entrada do usuário
    codeTextarea.addEventListener('input', updateEditor);
    codeTextarea.addEventListener('scroll', syncScroll);

    // Adiciona suporte para TAB (indentação) no textarea
    codeTextarea.addEventListener('keydown', function(e) {
        if (e.key === 'Tab') {
            e.preventDefault();
            const start = this.selectionStart;
            const end = this.selectionEnd;

            // Define o valor do textarea com a indentação
            this.value = this.value.substring(0, start) + "\t" + this.value.substring(end);

            // Coloca o cursor após a indentação
            this.selectionStart = this.selectionEnd = start + 1;

            updateEditor(); // Atualiza após a tab
        }
    });

    // Exemplo de código inicial
    codeTextarea.value = `// Bem-vindo ao editor de código!
// Digite seu código JavaScript abaixo e veja o resultado em tempo real.

function saudacao(nome) {
    console.log("Olá, " + nome + "!");
}

saudacao("Mundo");

let numero = 10;
const PI = 3.14;

if (numero > 5) {
    console.log("Número é maior que 5.");
} else {
    console.log("Número é menor ou igual a 5.");
}
```

```
for (let i = 0; i < 3; i++) {
    console.log("Contador: " + i);
}

/*
Este é um comentário
de múltiplas linhas.
*/

console.log("Resultado de 5 + 3 =", 5 + 3);
`;

        // Inicializa o editor com o código de exemplo
        updateEditor();
    </script>
</body>
</html>
```

ANEXO P - CÓDIGO GERADO - MODELO D, ARTEFATO 4

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Editor de Código</title>
    <style>
        body {
            margin: 0;
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            height: 100vh;
            background-color: #f0f0f0;
        }
        .container {
            display: flex;
            flex: 1;
            padding: 10px;
            gap: 10px;
        }
        .editor-section {
            flex: 1;
            display: flex;
            flex-direction: column;
        }
        .editor-section label {
            font-weight: bold;
            margin-bottom: 5px;
        }
```

```
        textarea {
            flex: 1;
            resize: none;
            border: 1px solid #ccc;
            padding: 10px;
            font-family: 'Courier New', Courier, monospace;
            font-size: 14px;
            background-color: #fff;
        }
        #output-section {
            flex: 1;
            display: flex;
            flex-direction: column;
        }
        #output {
            flex: 1;
            border: 1px solid #ccc;
            background-color: #fff;
            overflow: auto;
        }
        .controls {
            padding: 10px;
            background-color: #e0e0e0;
            text-align: right;
        }
        button {
            padding: 8px 16px;
            background-color: #007bff;
            color: #fff;
            border: none;
            cursor: pointer;
            font-size: 14px;
        }
        button:hover {
            background-color: #0056b3;
        }
        .keyword { color: #0000ff; }
        .string { color: #a31515; }
        .comment { color: #008000; }
        .selector { color: #800080; }
        .property { color: #ff0000; }
        .value { color: #0000ff; }
    </style>
</head>
<body>
    <div class="container">
        <div class="editor-section">
            <label for="html">HTML</label>
            <textarea id="html" placeholder="Digite seu HTML aqui..."></textarea>
        </div>
        <div class="editor-section">
            <label for="css">CSS</label>
            <textarea id="css" placeholder="Digite seu CSS aqui..."></textarea>
        </div>
        <div class="editor-section">
            <label for="js">JavaScript</label>
            <textarea id="js" placeholder="Digite seu JavaScript aqui..."></textarea>
        </div>
```

```
        <div id="output-section">
            <label>Saída</label>
            <iframe id="output"></iframe>
        </div>
    </div>
    <div class="controls">
        <button onclick="runCode()">Executar</button>
    </div>

    <script>
        const htmlEditor = document.getElementById('html');
        const cssEditor = document.getElementById('css');
        const jsEditor = document.getElementById('js');
        const output = document.getElementById('output');

        function highlightSyntax(text, type) {
            let highlighted = text.replace(/</g, '&lt;').replace(/>/g, '&gt;');
            if (type === 'html') {
                highlighted = highlighted
                    .replace(/(<\/?[\w\s="/.':;#-\/\?]+>)/gi, '<span class="keyword">$1</span>')
                    .replace(/"([^"]*)"/g, '<span class="string">"$1"</span>');
            } else if (type === 'css') {
                highlighted = highlighted
                    .replace(/([^{]+)(?=\s*{)/g, '<span class="selector">$1</span>')
                    .replace(/([a-z\-]+)(?=\s*:)/gi, '<span class="property">$1</span>')
                    .replace(/:\s*([^;]+)/g, ': <span class="value">$1</span>')
                    .replace(/\/\*[\s\S]*?\*\//g, '<span class="comment">$&</span>');
            } else if (type === 'js') {
                highlighted = highlighted

.replace(/\b(function|return|var|let|const|if|else|for|while|do|switch|case|break|continue|new|try|c
atch|throw)\b/g, '<span class="keyword">$1</span>')
                    .replace(/"([^"]*)"|'([^']*)'/g, '<span class="string">$&</span>')
                    .replace(/\/\/.*/g, '<span class="comment">$&</span>')
                    .replace(/\/\*[\s\S]*?\*\//g, '<span class="comment">$&</span>');
            }
            return highlighted;
        }

        function applyHighlighting() {
            const htmlContent = htmlEditor.value;
            const cssContent = cssEditor.value;
            const jsContent = jsEditor.value;

            const htmlDisplay = document.createElement('div');
            htmlDisplay.innerHTML = highlightSyntax(htmlContent, 'html');
            htmlEditor.parentNode.appendChild(htmlDisplay);
            htmlDisplay.style.position = 'absolute';
            htmlDisplay.style.pointerEvents = 'none';
            htmlDisplay.style.opacity = '0.7';

            const cssDisplay = document.createElement('div');
            cssDisplay.innerHTML = highlightSyntax(cssContent, 'css');
            cssEditor.parentNode.appendChild(cssDisplay);
            cssDisplay.style.position = 'absolute';
            cssDisplay.style.pointerEvents = 'none';
            cssDisplay.style.opacity = '0.7';
```

```
            const jsDisplay = document.createElement('div');
            jsDisplay.innerHTML = highlightSyntax(jsContent, 'js');
            jsEditor.parentNode.appendChild(jsDisplay);
            jsDisplay.style.position = 'absolute';
            .style.pointerEvents = 'none';
            jsDisplay.style.opacity = '0.7';
        }

        function runCode() {
            const html = document.getElementById('html').value;
            const css = document.getElementById('css').value;
            const js = document.getElementById('js').value;
            const output = document.getElementById('output').contentWindow.document;

            output.open();
            output.write(`
                <html>
                <head>
                    <style>${css}</style>
                </head>
                <body>
                    ${html}
                    <script>${js}<\/script>
                </body>
            </html>
            `);
            output.close();
        }

        // Atualizar destaque de sintaxe em tempo real
        htmlEditor.addEventListener('input', applyHighlighting);
        cssEditor.addEventListener('input', applyHighlighting);
        jsEditor.addEventListener('input', applyHighlighting);

        // Executar código inicial (se houver)
        runCode();
    </script>
</body>
</html>
```