Engenharia de *Prompt* na Geração de Interfaces a partir de Protótipos Visuais: uma Avaliação Empírica em IDE Agêntica

Caio Rian Reinaldo de Sousa
Campus Quixadá, Universidade Federal do Ceará
Quixadá, Brasil
caio.rianbr@alu.ufc.br

Beatriz Nascimento de Oliveira
Campus Quixadá, Universidade Federal do Ceará
Quixadá, Brasil
beatriznascimento@alu.ufc.br

Francisco Jerferson Martins da Silva
Campus Quixadá, Universidade Federal do Ceará
Quixadá, Brasil
jeffhyuu1610@alu.ufc.br

Carla Bezerra
Campus Quixadá, Universidade Federal do Ceará
Quixadá, Brasil
carlailane@gmail.com

Resumo

A geração de código assistida por *Large Language Models* (LLMs) em ambientes de desenvolvimento agênticos tem deslocado o esforço humano da escrita literal de código para a formulação de instruções em linguagem natural, consolidando a Engenharia de *Prompt* (EP) como disciplina central para orientar modelos generativos. Este trabalho compara empiricamente cinco técnicas de EP (*zero-shot*, *role prompting*, *structured prompting*, *Chain-of-Thought* e *combined prompting*) na geração de interfaces React a partir de protótipos visuais de alta fidelidade do Figma, no Cursor, IDE agêntica integrada ao modelo Claude Sonnet 4.6. Em três execuções por técnica, seis métricas de esforço de geração (tempo, intervenções humanas, contexto consumido, linhas de código, *tokens* e *prompts* adicionais) foram combinadas à avaliação de três desenvolvedores sobre fidelidade visual, *layout*, funcionalidade, completude, consistência e erros visuais. Os resultados indicam uma tensão estrutural entre a autonomia do agente e a aderência ao protótipo: o *role prompting* obteve a maior qualidade média (4,00) ao custo do maior esforço humano do conjunto, enquanto o *combined prompting* emergiu como a abordagem de melhor equilíbrio, com o menor *score* de esforço (0,05) e qualidade competitiva (3,71), sendo a única a concluir as execuções representativas sem correção adicional. A complexidade da tela amplifica essas diferenças, com *prompts* mais elaborados levando vantagem em interfaces estruturalmente mais complexas, resultado com implicações diretas sobre o esforço corretivo exigido quando essas interfaces entram em manutenção.

Palavras-chave

Engenharia de *Prompt*, Modelos de Linguagem de Grande Escala, Geração de Interfaces, IDEs Agênticas, Prototipação de Interfaces

1 Introdução

A geração de código apoiada por *Large Language Models* (LLMs) tem deslocado parte do esforço humano de desenvolvimento da escrita literal de código para a formulação de instruções em linguagem natural [15], o que consolidou a Engenharia de *Prompt* (EP) como disciplina voltada a métodos sistemáticos de elaboração e refinamento de *prompts* para orientar modelos generativos a produzirem saídas mais precisas e controláveis [17, 18]. Estratégias como *zero-shot* [3], *role prompting* [25], *prompts* estruturados [31] e *Chain-of-Thought* (CoT) [24] são amplamente investigadas na área, cada uma operando por um mecanismo distinto de orientação do modelo.

No domínio das interfaces, bibliotecas como o *React* [7] ocupam papel central na construção de aplicações web, mas traduzir *designs* em código executável permanece desafiador e gera inconsistências entre o protótipo planejado e a implementação final. A maturação de LLMs multimodais abriu uma fronteira nesse contexto: a conversão direta de protótipos de alta fidelidade [16], amplamente produzidos em ferramentas como o Figma[1] [11, 12], em código funcional. Em paralelo, o ferramental evoluiu de assistentes de autocompletar para IDEs agênticas, capazes de planejar, executar e iterar tarefas com mínima intervenção humana [9]; o Cursor[2] desloca a interação para o próprio ambiente de codificação [8], introduzindo variáveis novas, como contexto consumido, intervenções pontuais e variabilidade entre execuções.

Apesar do crescente interesse, ainda são escassas as investigações que combinem três dimensões simultaneamente: a aplicação sistemática de técnicas de EP à geração de interfaces, o uso de protótipos visuais de alta fidelidade como entrada e o emprego de IDEs agênticas como ambiente de execução. Boa parte dos estudos sobre LLMs em desenvolvimento concentra-se em problemas algorítmicos ou *benchmarks* genéricos [5, 23], enquanto avaliações de geração de interfaces privilegiam métricas automatizadas em ambientes sintéticos, sem considerar a percepção humana sobre a aderência visual ao *design* [27].

Esse deslocamento tem implicações diretas para a evolução e a manutenção de software. O código gerado por uma IDE agêntica a partir de um protótipo não é um artefato terminal: ele passa a integrar uma base que será lida, corrigida e evoluída por desenvolvedores. Divergências entre protótipo e implementação convertem-se em dívida de *design* a ser reconciliada em ciclos posteriores, e o esforço corretivo exigido já durante a geração, aqui capturado pelas intervenções humanas e pelos *prompts* adicionais, é um custo de manutenção antecipado, embutido na escolha da instrução.

Diante desse cenário, este trabalho conduz uma avaliação empírica comparativa de cinco técnicas de EP (*zero-shot*, *role prompting*, *structured prompting*, CoT e *combined prompting*) na geração de interfaces em React a partir de protótipos de alta fidelidade do Figma, no Cursor, IDE agêntica integrada ao modelo Claude Sonnet 4.6 [1], cruzando seis métricas de esforço de geração com a avaliação

[1] Figma: https://www.figma.com
[2] Cursor: https://cursor.com.

de três desenvolvedores sobre seis critérios de qualidade visual e funcional.

2 Trabalhos Relacionados

Estudos recentes investigam, sob diferentes ângulos, a geração de interfaces apoiada por LLMs. Na comparação sistemática de técnicas de EP, Wang et al. [22] avaliam *zero-shot*, *role prompting* e CoT em LLMs avançados para tarefas de engenharia de software, sem considerar entradas visuais ou interfaces gráficas, lacuna discutida na Seção 5. Zhang et al. [29] já identificavam o valor de imagens de GUIs como entrada para síntese de código, antes dos LLMs multimodais.

Si et al. [19] propõem o *Design2Code*, *benchmark* para conversão de *designs* em HTML e CSS que combina métricas automáticas e avaliação humana, sem aplicar técnicas explícitas de EP nem tecnologias como React, e Wan et al. [21] propõem o *DCGen*, que segmenta capturas de tela para reduzir omissões e distorções, também restrito a HTML/CSS e sem EP ou IDEs agênticas. Li et al. [13] avaliam, no *Sketch2Code*, a conversão de esboços de baixa fidelidade em protótipos web por refinamento iterativo com humanos, sem protótipos de alta fidelidade nem EP estruturada. Voltados a protótipos de alta fidelidade, Xiao et al. [28] propõem o *Prototype2Code*, geração de código *front-end* de ponta a ponta, e Zhou et al. [30] o *DeclarUI*, que conecta *design* e desenvolvimento por geração automatizada de UI declarativa, evidenciando lacunas de fidelidade visual e completude funcional que motivam este estudo.

Em nível de abstração intermediário, Nirumand and Cabot [14] derivam modelos de interface no padrão IFML a partir de *mock-ups* usando LLMs, sem gerar código executável nem comparar técnicas de EP. Em relação aos trabalhos identificados na literatura, nenhum estudo combina a comparação sistemática de múltiplas técnicas de EP, o uso de protótipos de alta fidelidade como entrada, a avaliação por desenvolvedores e o uso de uma IDE agêntica como ambiente de geração, o que constitui uma lacuna que o presente trabalho busca preencher. Os trabalhos mais próximos deste ilustram a lacuna por ângulos complementares: Wang et al. [22] comparam técnicas de EP, mas sobre tarefas textuais de engenharia de software; Si et al. [19] e Wan et al. [21] partem de entradas visuais e avaliam a saída, mas sob um único regime de instrução e sem IDE agêntica.

3 Metodologia

O objetivo é avaliar empiricamente o impacto de diferentes técnicas de EP na geração de interfaces a partir de protótipos visuais em uma IDE agêntica, sob três dimensões: esforço de geração, aderência visual ao protótipo e percepção de desenvolvedores. O estudo é orientado pelas questões:

- **QP1:** Como o esforço de geração varia entre as cinco técnicas de EP avaliadas?
- **QP2:** Qual o impacto de cada técnica de EP na aderência visual e na qualidade percebida das interfaces geradas, segundo a avaliação de desenvolvedores?
- **QP3:** Como a complexidade estrutural da interface influencia o desempenho relativo das técnicas de EP?
- **QP4:** Qual técnica de EP oferece o melhor equilíbrio entre esforço de geração e qualidade das interfaces produzidas?

3.1 Seleção das Técnicas de Engenharia de *Prompt*

As cinco técnicas foram selecionadas por cobrirem mecanismos de orientação distintos e por sua recorrência na literatura de EP aplicada à geração de código. O *zero-shot* fornece apenas a descrição da tarefa, explorando a capacidade de generalização em contexto dos LLMs [3]; o *role prompting* atribui ao modelo um papel profissional que condiciona o estilo e a profundidade das respostas [25]; o *structured prompting* organiza a instrução em seções explícitas, como contexto, tarefa, requisitos e restrições, reduzindo a ambiguidade da tarefa [31]; o CoT solicita que o modelo explicite um raciocínio intermediário antes da resposta final, originalmente validado em raciocínio aritmético e lógico [24]; e o *combined prompting* integra em um único *prompt* os três mecanismos anteriores, a atribuição de papel, a estrutura em seções e o planejamento prévio, hipótese sustentada por evidências de que a composição de técnicas tende a ser mais eficaz em tarefas de raciocínio complexo [6, 10].

As cinco configurações não são mutuamente exclusivas em sentido estrito: nenhuma fornece demonstrações ao modelo, de modo que todas operam em regime *zero-shot* no sentido literal. Aqui, o *zero-shot* designa a condição de referência, sobre a qual as demais acrescentam seus mecanismos.

3.2 Elaboração dos *Prompts*

Para cada técnica, foram elaborados três *prompts* em inglês, um por fluxo da aplicação avaliada (detalhada na Seção 3.3): a tela de boas-vindas, o fluxo de listagem de listas de tarefas (com três modais de criação, edição e exclusão) e o fluxo de tarefas, com visualizações em lista e em quadro *Kanban*. Um *prompt* extra padronizado, idêntico para as cinco técnicas, foi definido previamente aos experimentos e disparado ao final da execução sempre que a verificação visual constatasse que a navegação entre as três telas não era possível, requisito mínimo para que a interface pudesse ser avaliada. O *prompt* descrevia o comportamento observado e delimitava os arquivos passíveis de correção, sem autorizar refatorações.

A estrutura interna dos *prompts* variou conforme os mecanismos de cada técnica: no *zero-shot*, a instrução limitou-se à listagem dos arquivos a implementar, dos protótipos de referência e de uma enumeração sucinta das funcionalidades esperadas, sem orientação sobre papel, estrutura ou raciocínio; no *role prompting*, cada *prompt* iniciava com a atribuição de um papel profissional específico ao fluxo (por exemplo, *senior front-end engineer specialized in reusable UI components* no fluxo de listas); no *structured prompting*, a instrução foi organizada nas seções *Context*, *Task*, *Files*, *Requirements*, *Constraints* e *Output Format*; no CoT, o modelo era instruído a planejar a implementação em etapas antes de codificar; e o *combined prompting* reuniu a atribuição de papel, a estrutura em seções e a solicitação de planejamento prévio em um único *prompt*. Além do mecanismo, as instruções variam na especificação das funcionalidades, ausente no CoT, sucinta no *zero-shot* e mais detalhada nas demais, e também na extensão; ambas as variações são retomadas na Seção 6. Os quinze *prompts* base e o *prompt* extra estão disponibilizados publicamente (ver Disponibilidade de Artefatos).

3.3 Protótipo Visual e Repositório Base

O protótipo de entrada representa uma aplicação de gerenciamento de tarefas, feita no Figma com tema escuro e ícones *Phosphor*, cobrindo três fluxos: a tela de boas-vindas, a listagem de listas de tarefas e o fluxo de tarefas, com visualizações em lista e em quadro *Kanban*. Foram prototipados ainda seis modais de criação, edição e exclusão, três associados às listas de tarefas e três às tarefas.

O repositório base disponibilizado publicamente reúne um *back-end* completo em *Ruby on Rails* com *PostgreSQL* e um *front-end* inicializado com *Vite*, React, TypeScript, TailwindCSS, React Hook Form, Zod, Axios e React Router, no qual doze arquivos de componentes e páginas foram deixados em branco para serem implementados pelo modelo durante os experimentos. A escolha do React se justifica por sua ampla adoção no mercado [20] e pelo desempenho consolidado de LLMs em linguagens e *frameworks* amplamente representados nos dados de treinamento [4].

Esses arquivos distribuem-se entre os fluxos em ordem crescente de complexidade estrutural, segundo classificação operacional adotada neste estudo a partir do número de arquivos, modais e visualizações alternativas: um arquivo estático e sem estado na tela de boas-vindas; quatro arquivos, com listagem dinâmica, busca e três modais, na listagem de listas; e oito arquivos, com duas visualizações sobre o mesmo conjunto de dados e três modais próprios, no fluxo de tarefas, sendo o modal de exclusão compartilhado pelos dois últimos fluxos. Não se trata de escala validada na literatura, e sim de ordenação restrita ao protótipo avaliado, limitação retomada na Seção 6.

3.4 Execução dos Experimentos e Coleta de Métricas

Cada uma das cinco técnicas foi executada três vezes de forma independente, a partir de um repositório isolado clonado do repositório base, totalizando quinze execuções. Todas as execuções foram conduzidas por um dos pesquisadores, na mesma máquina e sob a mesma configuração do agente. Em cada execução, o repositório era aberto no Cursor (versão *Pro*) configurado para utilizar o modelo Claude Sonnet 4.6 [1], e os três *prompts* base da técnica eram disparados em sequência; ao final, a aplicação era executada no navegador para verificação visual, e o *prompt* extra era disparado caso a navegação entre as três telas não fosse possível.

As intervenções correspondem às autorizações concedidas ao agente e dependem, portanto, da configuração de execução automática do Cursor, mantida idêntica nas quinze execuções. Nenhuma regra de projeto do Cursor esteve ativa: o diretório `.cursor/rules` foi removido antes de cada execução e suas regras foram preservadas, inativas, no diretório `cursor-rules-nao-aplicadas` do repositório base; a única regra de usuário do ambiente definia o idioma das respostas conversacionais, sem orientação sobre geração de código.

Seis métricas de esforço foram coletadas por execução: tempo total (gravação de tela, em minutos), número de intervenções (autorizações concedidas ao agente durante a execução), contexto consumido (em milhares de *tokens* sobre a janela máxima de 200 mil *tokens* do modelo), linhas de código geradas, *tokens* totais utilizados e uso de *prompt* extra (variável binária).

3.5 Seleção das Execuções Representativas

A seleção do par mais próximo submete à avaliação humana o comportamento *modal* de cada técnica, e não sua média aritmética, que poderia mascarar padrões distintos. O critério prioriza estabilidade e, por construção, atenua a variabilidade que motivou a repetição; para preservar essa informação, reportamos a amplitude entre execuções (por exemplo, 11 e 4 intervenções no *role prompting*) como indicador de dispersão, retomado na Seção 6. Essa variabilidade entre execuções de um mesmo *prompt* é fenômeno documentado na literatura sobre LLMs [2].

A proximidade entre um par de execuções i e j foi calculada por um *score* de distância ponderado:

$$S_{ij} = \sum_{k=1}^{6} w_k \cdot d_{ij}^k \tag{1}$$

em que d_{ij}^k é a distância normalizada entre as execuções na métrica k, dada por:

$$d_{ij}^k = \frac{|m_{ki} - m_{kj}|}{\max(m_k) - \min(m_k)} \tag{2}$$

Nessa expressão, m_{ki} é o valor da métrica k na execução i, e $\max(m_k)$ e $\min(m_k)$ são o maior e o menor valor dessa métrica entre as três execuções da própria técnica, de modo que a normalização é interna a cada técnica; quando a amplitude é nula, caso da variável binária de *prompt* extra em três técnicas, define-se $d_{ij}^k = 0$. O peso w_k atribuído a cada métrica foi definido a partir do princípio de privilegiar métricas que refletem diretamente o esforço humano: tempo (30%), intervenções (20%), *prompt* extra (20%), *tokens* (15%), linhas de código (10%) e contexto consumido (5%). O par de menor *score* em cada técnica foi selecionado como representativo, resultando nas execuções 1 e 2 para o *zero-shot*, 2 e 3 para o *role prompting*, 2 e 3 para o *structured prompting*, 2 e 3 para o CoT e 1 e 3 para o *combined prompting*.

3.6 Avaliação com Desenvolvedores

As dez interfaces selecionadas, geradas a partir do protótipo descrito na Seção 3.3, foram avaliadas por três desenvolvedores selecionados por conveniência entre profissionais e acadêmicos da rede de contatos dos pesquisadores: dois graduandos e um mestrando em Engenharia de Software, de 22 a 23 anos, com 2 a 2,5 anos de experiência em desenvolvimento e prática prévia com *front-end* em React.

A avaliação ocorreu em duas etapas. Na primeira, os três avaliadores participaram de uma sessão conjunta de calibração sobre os objetivos do estudo, o protótipo e o funcionamento esperado de cada tela e modal. Na segunda, cada um avaliou individualmente, em sessão presencial dedicada e em dia separado, no computador do pesquisador, que executava os dez projetos e mantinha postura exclusivamente observadora. Os projetos receberam rótulos neutros, sem que os avaliadores soubessem qual técnica gerou cada interface; a ordem de apresentação foi a mesma para os três. O avaliador dispunha do protótipo do Figma e de um roteiro de navegação com ênfase em cenários de limite, como títulos extensos e listas com muitos itens, registrando notas e observações em planilha estruturada.

As notas variam de 0 a 5 (0 = não atendido; 1 = muito ruim; 2 = ruim; 3 = regular; 4 = bom; 5 = excelente), sempre no sentido em que

Tabela 1: Médias das métricas de esforço e *score* de esforço humano por técnica (Ctx = contexto consumido em milhares de *tokens*; *Tokens* em milhões; *Parcial* = *prompt* extra necessário em apenas uma das duas execuções representativas)

Técnica	Tempo	Interv.	Ctx	Linhas	*Tokens*	Extra	*Score*
Zero-shot	12,0	2,5	92,75	959,0	2,96	Sim	0,53 (M)
Role prompting	15,0	7,5	97,50	985,0	3,27	Sim	1,00 (A)
Structured prompting	12,5	1,5	94,70	981,0	2,86	Sim	0,51 (M)
CoT	14,5	0,0	95,65	1.115,0	3,23	Parcial	0,37 (M)
Combined prompting	12,0	1,0	91,85	1.152,5	2,72	Não	0,05 (B)

valores mais altos indicam melhor desempenho, inclusive em erros visuais, em que a nota mais alta corresponde a menos falhas, em seis critérios (fidelidade visual, *layout*, funcionalidade, completude, consistência visual e erros visuais), para cada uma das três telas de cada projeto: dezoito notas por avaliador por projeto. Os três avaliaram os dez projetos, de modo que cada valor da Tabela 2 é a média de dezoito observações (três avaliadores × três telas × duas execuções).

A concordância foi estimada pelo coeficiente de correlação intraclasse (modelo de duas vias, concordância absoluta). Sobre as 179 avaliações completas no nível do item (projeto × tela × critério), obteve-se ICC(2,3) = 0,56 (IC 95%: 0,41–0,66), concordância moderada; no nível de agregação usado nas análises (técnica × tela), ICC(2,3) = 0,75. A concordância variou por critério, de 0,74 em *layout* a nula em erros visuais, único em que os avaliadores não convergiram, o que é retomado na Seção 6.

4 Resultados

4.1 Esforço de Geração

A Tabela 1 consolida as médias das métricas de esforço nas execuções representativas e o *score* ponderado de esforço humano, calculado a partir de intervenções (peso 40%), *prompt* extra (peso 40%, com valores 1,0 para Sim, 0,5 para Parcial e 0,0 para Não) e tempo de execução (peso 20%), com classificação Baixo (B, 0,00–0,33), Médio (M, 0,34–0,66) ou Alto (A, 0,67–1,00), limiares que correspondem a terços iguais do intervalo normalizado e foram definidos previamente à análise.

O *score* restringe-se às três métricas de intervenção humana direta; *tokens*, linhas de código e contexto consumido, indicadores de custo computacional, permanecem fora do índice, embora reportados na Tabela 1. Os dois *scores* usam conjuntos e pesos distintos porque respondem a propósitos distintos: a Equação 1 mede proximidade entre execuções sobre as seis métricas, ao passo que o *score* de esforço mede apenas o envolvimento humano. Em ambos, os pesos apoiam-se em julgamento dos pesquisadores. Para aferir a dependência do resultado em relação a essa escolha, variaram-se os três pesos do *score* de esforço em incrementos de 0,05 sob soma unitária, gerando 231 combinações: o *combined prompting* mantém o menor *score* em 95,7% delas, deixando de ser o mínimo apenas quando o peso das intervenções ultrapassa 0,80, caso em que é superado pelo CoT.

O CoT foi a única técnica com zero intervenções médias e o *combined prompting* a única a dispensar *prompt* extra nas duas execuções representativas; juntas formam o grupo de maior autonomia do agente (*scores* 0,37 e 0,05). O *role prompting* concentrou o maior custo, com 7,5 intervenções e *score* máximo (1,00), além da maior variabilidade interna (11 e 4 intervenções). O *zero-shot* e o *structured prompting* ficam em posição intermediária (0,53 e 0,51), distinguindo-se pela origem do esforço: no primeiro, as intervenções elevam o *score* apesar do tempo mínimo; no segundo, menos intervenções são compensadas pela necessidade sistemática de *prompt* extra. O *combined prompting* apresentou ainda a maior densidade de saída por *token*, razão entre linhas geradas e *tokens* consumidos: 423,6 linhas por milhão, contra 344,8 do CoT.

Tabela 2: Médias consolidadas por técnica e critério (Fid. = fidelidade visual; Func. = funcionalidade; Compl. = completude; Cons. = consistência visual; Erros = erros visuais)

Técnica	Fid.	*Layout*	Func.	Compl.	Cons.	Erros	Média
Zero-shot	3,56	3,94	4,00	3,67	3,44	3,78	3,73
Role prompting	4,17	3,83	4,11	4,44	4,17	3,28	4,00
Structured prompting	3,11	3,39	3,50	3,33	2,89	3,33	3,26
CoT	3,72	3,56	3,83	4,06	3,44	3,50	3,69
Combined prompting	3,56	3,50	4,17	3,61	3,72	3,71	3,71

Achado 1 (QP1): o esforço de geração varia de forma expressiva entre as cinco técnicas, do *score* mínimo do *combined prompting* (0,05) ao máximo do *role prompting* (1,00); o CoT e o *combined prompting* formam o grupo de maior autonomia do agente, enquanto o *role prompting* concentra o maior custo humano de geração.

4.2 Qualidade Percebida: Análise Quantitativa

A Tabela 2 consolida as médias por técnica e critério de avaliação, calculadas a partir dos dados brutos disponibilizados. O *role prompting* obteve a maior média geral (4,00), destacando-se em completude (4,44), fidelidade visual (4,17) e consistência visual (4,17), mas com o pior desempenho relativo em erros visuais (3,28), critério em que a concordância entre avaliadores foi nula e cuja leitura isolada exige cautela adicional.

O *zero-shot* registrou a segunda maior média (3,73), com o perfil mais homogêneo entre critérios (amplitude de 0,56 pontos) e a maior pontuação em erros visuais (3,78); o *combined prompting* alcançou 3,71, com a maior pontuação em funcionalidade (4,17) do conjunto, e o CoT, 3,69. O *structured prompting* registrou a menor média (3,26) e o único valor abaixo de 3,00 de todo o conjunto (consistência visual, 2,89). Transversalmente, a funcionalidade foi o critério de maior média entre as cinco técnicas (3,92), variando de 3,50 a 4,17, ainda que não seja o critério de maior pontuação dentro de cada técnica isoladamente: no *role prompting* a completude (4,44) e a fidelidade visual (4,17) o superam, e no CoT a completude (4,06) também. A consistência visual apresentou a maior variação entre técnicas (amplitude de 1,28 pontos). O mapa de calor da Figura 1 revela ainda uma tendência de queda de desempenho entre a tela mais simples e a mais complexa em quatro das cinco técnicas, monotônica no *zero-shot*, no *role prompting* e no *structured prompting*; no CoT, a pontuação sobe da tela de boas-vindas (3,67) para a listagem de listas (3,86) antes de cair no fluxo de tarefas (3,53). O *combined prompting* é a única exceção ao padrão, registrando sua maior média justamente na tela de maior complexidade (fluxo de tarefas, 3,94 contra 3,72 na tela de boas-vindas).

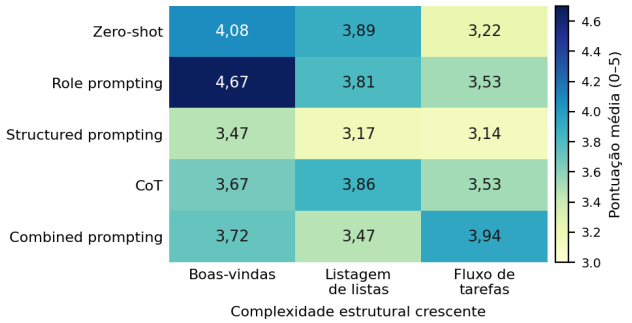

Figura 1: Pontuação média (0–5) por técnica de EP e tela, com complexidade estrutural crescente da esquerda para a direita.

Achado 2 (QP2): a qualidade percebida varia de 3,26 a 4,00: o *role prompting* lidera em fidelidade, completude e consistência visual ao custo do maior esforço, e o *structured prompting* registra a menor qualidade. As observações qualitativas corroboram o quadro: quebras de *layout* são transversais e a assimetria em imagem/ícone acompanha a fidelidade do *role prompting*.

A diferença entre as duas execuções representativas ficou abaixo de 0,40 ponto em todas as técnicas (de 0,07 no CoT a 0,37 no *structured prompting*), indicando que os pares capturam de forma estável o comportamento típico de cada abordagem.

Achado 3 (QP3): quatro das cinco técnicas pontuam menos no fluxo de tarefas do que na tela de boas-vindas; o *combined prompting* é a única exceção, com sua maior pontuação na tela mais complexa, o que indica vantagem relativa de *prompts* mais elaborados em interfaces estruturalmente exigentes.

4.3 Qualidade Percebida: Observações Qualitativas

As percepções textuais foram categorizadas indutivamente pelo primeiro autor, com as categorias emergindo da leitura integral dos comentários, sem esquema definido a priori; a planilha completa está disponibilizada. Uma ocorrência é um comentário registrado por um avaliador, o mesmo defeito relatado pelos três conta três ocorrências, e um comentário pode receber mais de uma categoria, o que ocorreu em 22,3% dos 157 comentários e explica o total de marcações da Tabela 3 exceder o número de comentários. O volume variou entre avaliadores (63, 34 e 60), de modo que as frequências refletem também a propensão individual ao registro.

Quebras de *layout* foram a categoria mais frequente (68 ocorrências), presentes em todas as técnicas e associadas a redimensionamento incorreto de cartões e sobreposição em cenários de limite, um problema transversal, menos sensível à formulação do *prompt*. Problemas de imagem e ícone (43 ocorrências) apresentaram a maior assimetria entre técnicas: o *role prompting* registrou apenas 2 ocorrências, contra 12 no CoT, padrão coerente com a

Tabela 3: Frequência de ocorrências por categoria de problema e técnica (ZS = *zero-shot*; RP = *role prompting*; SP = *structured prompting*; CP = *combined prompting*)

Categoria	ZS	RP	SP	CoT	CP	Total
Quebra de *layout*	18	15	13	8	14	68
Imagem/ícone	11	2	8	12	10	43
Funcionalidade	6	9	4	11	10	40
Inconsist. cores	9	6	4	5	3	27
Completude	4	3	4	3	2	16

maior fidelidade visual atribuída ao *role prompting* na avaliação quantitativa. Problemas de funcionalidade totalizaram 40 ocorrências, a terceira categoria mais frequente, com variação expressiva entre técnicas: o CoT e o *combined prompting* concentraram os maiores números de registros, com 11 e 10 ocorrências, seguidos pelo *role prompting* (9), enquanto o *zero-shot* e o *structured prompting* registraram as menores frequências (6 e 4). Os comentários descrevem falhas no indicador de progresso, reordenação incorreta após marcação e modais em estado indevido. O padrão contrasta com a Tabela 2, em que o *combined prompting* tem a maior pontuação em funcionalidade (4,17), sugerindo que esses problemas foram pontuais. Inconsistências de cores (27 ocorrências) tenderam a diminuir das técnicas de menor para as de maior complexidade de *prompt* (de 9 no *zero-shot* a 3 no *combined prompting*), com uma exceção pontual no CoT, corroborando a maior consistência visual atribuída às técnicas de *prompt* mais elaborado. Omissões de completude foram a categoria menos frequente (16 ocorrências), homogêneas entre técnicas e limitadas a ajustes tipográficos e à ausência de elementos secundários do protótipo.

4.4 Análise Conjunta

O cruzamento entre esforço e qualidade percebida revela uma relação de compromisso. Classificando a qualidade como Regular (3,00–3,49), Boa (3,50–3,99) ou Ótima (4,00–5,00), limiares alinhados à escala de notas, e o equilíbrio como Alto quando o esforço é Baixo e a qualidade Boa ou Ótima, Médio quando o esforço é Médio e a qualidade Boa ou o esforço Baixo e a qualidade Regular, e Baixo quando o esforço é Alto ou a qualidade Regular, obtém-se: equilíbrio Alto apenas para o *combined prompting* (esforço 0,05; qualidade 3,71); Médio para o *zero-shot* (0,53; 3,73) e o CoT (0,37; 3,69); e Baixo para o *role prompting* (1,00; 4,00) e o *structured prompting* (0,51; 3,26), o primeiro pelo custo humano de geração e o segundo por combinar esforço moderado com a menor qualidade do conjunto.

Achado 4 (QP4): o *combined prompting* oferece o melhor equilíbrio entre esforço e qualidade; o *role prompting* e o CoT são soluções de segunda ordem, cada um sacrificando uma das dimensões, e o *structured prompting* isolado é a opção menos vantajosa.

5 Discussão

O desempenho superior do *role prompting* em fidelidade e consistência visual indica que a atribuição de uma persona especializada por fluxo condiciona o modelo a convenções mais próximas das expectativas de um desenvolvedor, ao custo de maior supervisão humana.

O resultado dialoga com Wang et al. [22], que reporta ganhos de *role prompting* e CoT concentrados em modelos de raciocínio menos avançado: aqui, o CoT isolado não liderou em qualidade percebida (3,69) apesar do esforço reduzido (0,37), sugerindo que, em IDEs agênticas com modelos como o Claude Sonnet 4.6, seu benefício recai sobre a autonomia do agente, e não sobre a fidelidade visual da saída.

O melhor desempenho do *combined prompting* sugere que os três mecanismos combinados não competem, mas se complementam: a persona direciona o estilo do código, a estrutura em seções reduz a ambiguidade da tarefa e o planejamento prévio parece reduzir a necessidade de correções; foi a única técnica a entregar, nas duas execuções representativas, uma aplicação navegável entre as três telas sem *prompt* adicional. O contraste com o *structured prompting* isolado é informativo: a organização em seções, sem persona nem raciocínio explícito que oriente sua interpretação, mostrou-se insuficiente para garantir aderência visual, mesmo reduzindo intervenções. A estrutura sintática do *prompt*, por si só, não substitui a orientação semântica sobre o papel do agente ou o raciocínio esperado.

A vantagem de *prompts* mais elaborados em interfaces estruturalmente complexas, evidenciada pelo desempenho do *combined prompting* no fluxo de tarefas, é coerente com Si et al. [19] e Wan et al. [21], que identificam maior dificuldade dos LLMs multimodais em páginas com maior volume de elementos, ainda que sem avaliar o efeito de técnicas de EP sobre essa dificuldade. Na prática, a escolha da técnica deve considerar o perfil da tarefa: para telas simples, *zero-shot* ou CoT tendem a bastar; para interfaces de maior complexidade estrutural, o *combined prompting* apresentou o melhor equilíbrio entre esforço e qualidade, coerente com estudos que apontam ganhos da composição de técnicas em engenharia de software [6, 10, 31].

6 Ameaças à Validade

Este estudo apresenta ameaças à validade em quatro dimensões [26]. Na **validade de conclusão**, o número reduzido de avaliadores limita a representatividade estatística das médias, e a concordância entre eles é moderada (ICC(2,3) = 0,56), o que recomenda ler diferenças de pequena magnitude como indicativas; o critério erros visuais, sem concordância, não sustenta conclusão isolada. A categorização qualitativa foi conduzida por um único autor, sem verificação independente. As conclusões sobre qualidade referem-se ainda ao par de execuções representativas de cada técnica, selecionado por proximidade em esforço: similaridade em esforço não implica representatividade em qualidade, e as cinco execuções excluídas não foram avaliadas.

Na **validade interna**, o Cursor opera como intermediário entre o *prompt* e o modelo, podendo enriquecer a entrada com contexto do projeto e do histórico de interações, variável não controlada. A ordem dos dez projetos foi constante entre avaliadores, de modo que efeitos de ordem, como fadiga ou calibração progressiva do julgamento, não foram contrabalanceados.

Na **validade de construção**, os seis critérios cobrem as principais dimensões de qualidade visual e funcional em relação ao protótipo, mas não contemplam acessibilidade, desempenho de renderização nem qualidade e manutenibilidade do código gerado, dimensão relevante para a evolução posterior das interfaces. Cada técnica foi operacionalizada por um *prompt* por fluxo, redigido pelos autores, e as instruções diferem no mecanismo, na extensão e no nível de especificação das funcionalidades, de modo que o desenho não separa completamente o efeito do mecanismo do efeito do volume de instrução. A extensão isolada, contudo, não explica os resultados de qualidade: o *structured prompting*, com 114 palavras por *prompt* em média, é o segundo mais longo do conjunto e registra a menor qualidade percebida, enquanto o *role prompting*, com 70, registra a maior. A complexidade estrutural também não foi manipulada de forma independente, pois as três telas diferem simultaneamente em estrutura, funcionalidade, conteúdo e *prompt*; a evidência da QP3 indica uma tendência, e não um efeito isolado.

Na **validade externa**, os experimentos foram conduzidos exclusivamente com o Claude Sonnet 4.6 no Cursor sobre um único domínio de aplicação (um gerenciador de tarefas), de modo que os resultados podem não se generalizar para outros LLMs, IDEs agênticas ou tipos de aplicação.

7 Conclusões e Trabalhos Futuros

A adoção de IDEs agênticas na construção de interfaces coloca uma questão prática: como formular instruções que equilibrem o custo de geração com a qualidade visual do resultado. O esforço variou de forma expressiva entre as técnicas (QP1) e a percepção dos desenvolvedores acompanhou apenas parcialmente esse padrão (QP2), com o *role prompting* liderando em fidelidade ao maior custo humano e o *structured prompting* registrando o pior desempenho em aderência visual. O efeito se acentuou com a complexidade das interfaces (QP3) e, considerando esforço e qualidade em conjunto (QP4), o *combined prompting* emergiu como a abordagem de melhor equilíbrio no cenário avaliado, sugerindo que a tensão entre qualidade visual e autonomia do agente é atenuada pela composição de mecanismos de orientação em um único *prompt*.

Como trabalhos futuros, sugere-se a replicação com outros LLMs e IDEs agênticas, a ampliação do número de avaliadores e de domínios, métricas automatizadas de similaridade visual como complemento à avaliação humana e, sobretudo, a incorporação de critérios de manutenibilidade do código gerado, como duplicação e modularidade, determinantes para o custo de evolução das interfaces produzidas por IDEs agênticas.

Disponibilidade de Artefatos

Os artefatos deste trabalho estão organizados e disponibilizados publicamente: o repositório base, com protótipos, *back-end* e *front-end* incompleto (https://anonymous.4open.science/r/Projeto-E290); as quinze execuções realizadas (https://anonymous.4open.science/r/Testes-64F5); os *prompts* completos de cada técnica (https://zenodo.org/records/21326129); os dados de avaliação dos desenvolvedores (https://doi.org/10.5281/zenodo.21325380); e as observações qualitativas categorizadas (https://doi.org/10.5281/zenodo.21325426). O repositório base inclui um *script* que reproduz, dos dados brutos, todas as tabelas e figuras deste artigo.

Agradecimentos

Esta pesquisa foi apoiada pelo CNPq (processo 403304/2025-3).

Referências

[1] ANTHROPIC. 2026. Claude Sonnet 4.6 System Card. https://anthropic.com/claude-sonnet-4-6-system-card. Acesso em: 10 ago. 2026.

[2] Berk Atıl, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. 2025. Non-Determinism of "Deterministic" LLM System Settings in Hosted Environments. In *Proceedings of the 5th Workshop on Evaluation and Comparison of NLP Systems*, Mousumi Akter, Tahiya Chowdhury, Steffen Eger, Christoph Leiter, Juri Opitz, and Erion Çano (Eds.). Association for Computational Linguistics, Association for Computational Linguistics, Mumbai, India, 135–148. doi:10.18653/v1/2025.eval4nlp-1.12

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners *(NIPS '20)*. Neural Information Processing Systems Foundation, Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.

[4] Federico Cassano, John Gouwar, Francesca Lucchetti, Claire Schlesinger, Anders Freeman, Carolyn Jane Anderson, Molly Q. Feldman, Michael Greenberg, Abhinav Jangda, and Arjun Guha. 2024. Knowledge Transfer from High-Resource to Low-Resource Programming Languages for Code LLMs. *Proc. ACM Program. Lang.* 8, OOPSLA2, Article 338 (Oct. 2024), 28 pages. doi:10.1145/3689735

[5] Mark Chen, Jerry Tworek, Heewoo Jun, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG] https://doi.org/10.48550/arXiv.2107.03374

[6] Rogelio Cruz, Jonatan Contreras, Francisco Guerrero, Ezequiel Rodriguez, Carlos Valdez, and Citlali Carrillo. 2025. Prompt engineering and framework. arXiv:2506.10989 [cs.SE] https://doi.org/10.48550/arXiv.2506.10989

[7] Roy Derks. 2022. *React Projects.* Packt Publishing.

[8] Matthias Galster, Seyedmoein Mohsenimofidi, Jai Lal Lulla, Muhammad Auwal Abubakar, Christoph Treude, and Sebastian Baltes. 2026. Configuring Agentic AI Coding Tools. arXiv:2602.14690 [cs.SE] https://arxiv.org/abs/2602.14690

[9] Kosei Horikawa, Hao Li, Yutaro Kashiwa, Bram Adams, Hajimu Iida, and Ahmed E. Hassan. 2025. Agentic Refactoring. arXiv:2511.04824 [cs.SE] https://doi.org/10.48550/arXiv.2511.04824

[10] E. G. Santana Jr, Gabriel Benjamin, Melissa Araujo, Harrison Santos, David Freitas, Eduardo Almeida, Paulo Anselmo da M. S. Neto, Jiawei Li, Jina Chun, and Iftekhar Ahmed. 2025. Which Prompting Technique Should I Use? An Empirical Investigation of Prompting Techniques for Software Engineering Tasks. arXiv:2506.05614 [cs.SE] https://arxiv.org/abs/2506.05614

[11] Kristian Kolthoff, Felix Kretzer, Christian Bartelt, Alexander Maedche, and Simone Paolo Ponzetto. 2025. GUIDE. 4 pages. doi:10.1109/ICSE-Companion66252.2025.00010

[12] Felix Kretzer, Kristian Kolthoff, Christian Bartelt, Simone Paolo Ponzetto, and Alexander Maedche. 2025. Closing the Loop between User Stories and GUI Prototypes. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japão) *(CHI '25)*. ACM SIGCHI, Association for Computing Machinery, New York, NY, USA, Article 879, 19 pages. doi:10.1145/3706598.3713932

[13] Ryan Li, Yanzhe Zhang, and Diyi Yang. 2025. Sketch2Code. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Association for Computational Linguistics, Albuquerque, New Mexico, 3921–3955. doi:10.18653/v1/2025.naacl-long.198

[14] Atefeh Nirumand and Jordi Cabot. 2025. From Mock-Ups to IFML-Like GUI Models. In *Web Engineering: 25th International Conference, ICWE 2025, Delft, The Netherlands, June 30 – July 3, 2025, Proceedings* (Delft, The Netherlands). Springer-Verlag, Berlin, Heidelberg, 253–268. doi:10.1007/978-3-031-97207-2_20

[15] Zeeshan Rasheed, Muhammad Waseem, Kai Kristian Kemell, Aakash Ahmad, Malik Abdul Sami, Jussi Rasku, Kari Systä, and Pekka Abrahamsson. 2025. Large Language Models for Code Generation. arXiv:2501.16998 [cs.SE] https://doi.org/10.48550/arXiv.2501.16998

[16] Yvonne Rogers, Helen Sharp, and Jennifer Preece. 2023. *Interaction Design* (6th ed.). Wiley. https://www.wiley.com/en-us/Interaction%2BDesign%3A%2BBeyond%2BHuman-Computer%2BInteraction%2C%2B6th%2BEdition-p-9781119901099

[17] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2025. A Systematic Survey of Prompt Engineering in Large Language Models. arXiv:2402.07927 [cs.AI] https://arxiv.org/abs/2402.07927

[18] Sander Schulhoff, Michael Ilie, Nishant Balepur, et al. 2025. The Prompt Report. arXiv:2406.06608 [cs.CL] https://arxiv.org/abs/2406.06608

[19] Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2Code. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Association for Computational Linguistics, Albuquerque, New Mexico, 3956–3974. doi:10.18653/v1/2025.naacl-long.199

[20] STACK OVERFLOW. 2025. Stack Overflow Developer Survey 2025. Industry survey. https://survey.stackoverflow.co/2025/technology/. Acesso em: 10 ago. 2026.

[21] Yuxuan Wan, Chaozheng Wang, Yi Dong, Wenxuan Wang, Shuqing Li, Yintong Huo, and Michael Lyu. 2025. Divide-and-Conquer. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE094 (June 2025), 24 pages. doi:10.1145/3729364

[22] Guoqing Wang, Zeyu Sun, Sixiang Ye, Zhihao Gong, Yizhou Chen, Yifan Zhao, Qingyuan Liang, and Dan Hao. 2025. Do advanced language models eliminate the need for prompt engineering in software engineering? *ACM Trans. Softw. Eng. Methodol.* (Oct. 2025). doi:10.1145/3771933 Just Accepted.

[23] Yue Wang, Hung Le, Akhilesh Gotmare, Nghi Bui, Junnan Li, and Steven Hoi. 2023. CodeT5+. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). ACL SIGDAT, Association for Computational Linguistics, Singapore, 1069–1088. doi:10.18653/v1/2023.emnlp-main.68

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Neural Information Processing Systems Foundation, Curran Associates Inc., Red Hook, NY, USA, Article 1800, 14 pages.

[25] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. In *Proceedings of the 30th Conference on Pattern Languages of Programs* (Monticello, IL, USA) *(PLoP '23)*. The Hillside Group, The Hillside Group, USA, Article 5, 31 pages.

[26] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering.* Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-29044-2

[27] Jingyu Xiao, Ming Wang, Man Ho Lam, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R. Lyu. 2025. DesignBench. arXiv:2506.06251 [cs.SE] https://doi.org/10.48550/arXiv.2506.06251

[28] Shuhong Xiao, Yunnong Chen, Jiazhi Li, Liuqing Chen, Lingyun Sun, and Tingting Zhou. 2024. Prototype2Code. arXiv:2405.04975 [cs.SE] https://doi.org/10.48550/arXiv.2405.04975

[29] Zhihang Zhang, Ye Ding, and Chenlin Huang. 2023. Automatic Front-end Code Generation from image Via Multi-Head Attention. In *2023 4th International Conference on Computer Engineering and Application (ICCEA)* (Hangzhou, China). IEEE, 869–872. doi:10.1109/ICCEA58433.2023.10135462

[30] Ting Zhou, Yanjie Zhao, Xinyi Hou, Xiaoyu Sun, Kai Chen, and Haoyu Wang. 2025. DeclarUI. 2, FSE, Article FSE011 (June 2025), 23 pages. doi:10.1145/3715726

[31] Yangtian Zi, Harshitha Menon, and Arjun Guha. 2025. More Than a Score. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (Eds.). AFNLP, The Asian Federation of Natural Language Processing and The Association for Computational Linguistics, Mumbai, India, 2380–2402. doi:10.18653/v1/2025.ijcnlp-long.128