Inteligência Artificial Aplicada ao Desenvolvimento de Sistemas Web: um Estudo de Caso

João Janduy de Farias Filho[1], Sergio Muinhos Barroso Lima[2]

[1]Instituto Federal do Sudeste de Minas Gerais, Rio Pomba, Brasil
(joao.janduy@gmail.com)
[2]Instituto Federal do Sudeste de Minas Gerais, Rio Pomba, Brasil
(sergio.lima@ifsudestemg.edu.br)

Resumo: Esse artigo aborda a aplicação de inteligência artificial (IA) no desenvolvimento Web, explorando os benefícios, vantagens e desvantagens da aplicação da IA no desenvolvimento, com vistas à análise da qualidade dos documentos produzidos (modelagem), código fonte e interface do usuário. Para esse trabalho foi utilizado a plataforma V0 da Vercel, como interface para a interação com a IA, avaliando-se a qualidade, completude e corretude das respostas às solicitações da equipe. Foram avaliadas as principais etapas no desenvolvimento do sistema escolhido tais como: concepção, especificação de requisitos, modelagem, projeto, implementação e testes. Foram explorados ainda nesse artigo, os impactos da IA no mercado de trabalho.

Palavras-chave: Inteligência Artificial; Desenvolvimento web; V0.

INTRODUÇÃO

Esse artigo aborda a aplicação de inteligência artificial no desenvolvimento Web, explorando os benefícios, vantagens e desvantagens da aplicação da IA no desenvolvimento, com vistas a análise da qualidade dos documentos produzidos, o código-fonte, a interface com o usuário e prováveis impactos nas metodologias de desenvolvimento de software e no mercado de trabalho.

As origens da inteligência artificial generativa remontam a modelos estatísticos relativamente simples, como as Cadeias de Markov, introduzidas em 1906 e inicialmente utilizadas para a previsão de sequências (AL-AMIN et al., 2024). No entanto, foi na década de 1950 que o campo recebeu um marco conceitual crucial com o Teste de Turing, proposto por Alan Turing, que estabeleceu parâmetros fundamentais para avaliar a capacidade de uma máquina de simular a inteligência humana em diálogos (AL-AMIN et al., 2024).

A primeira materialização prática dessas ideias surgiu nos anos 1960 com o ELIZA (1966), considerado o primeiro *chatbot*. Ele simulava um terapeuta utilizando correspondência de padrões, embora sem qualquer compreensão real da linguagem. Seu sucessor, o PARRY (1972), inovou ao incorporar uma personalidade paranoica, testando assim a capacidade de imitar comportamentos humanos mais complexos e emocionais. Na década de 1980, programas como o Racter (1983) exploraram a geração aleatória e criativa de textos, ainda que frequentemente produzissem resultados incoerentes (AL-AMIN et al., 2024).

O período entre os anos 1980 e 1990 presenciou avanços na interação contínua, com sistemas como Jabberwacky (1988) e sua evolução para o Cleverbot (2008), que aprendiam diretamente com as interações humanas em tempo real. A virada para respostas mais estruturadas e naturais veio com o ALICE (1995), que utilizava a linguagem AIML e chegou a vencer prêmios por sua humanidade simulada (AL-AMIN et al., 2024).

A popularização dos *chatbots* ocorreu nos anos 2000, com o SmarterChild (2001), amplamente distribuído em mensageiros instantâneos e visto como um antecessor direto dos assistentes pessoais modernos. Paralelamente, projetos de pesquisa ambiciosos, como o CALO (2003), integraram aprendizado de máquina e contextualização, servindo de base para o desenvolvimento de assistentes futuros (AL-AMIN et al., 2024).

A década de 2010 consolidou essa tendência com o lançamento de assistentes virtuais amplamente adotados, como a Siri (2011), o Google Now (2012) e a Alexa (2014), todos empregando Processamento de Linguagem Natural (NLP) avançado. Nesse cenário, o XiaoIce (2014) se destacou ao priorizar a construção de conexões emocionais, conquistando milhões de usuários na Ásia (AL-AMIN et al., 2024).

Contudo, a revolução moderna da IA generativa efetivamente começou com o advento da arquitetura Transformer e seus modelos subsequentes. O GPT-1 (2018) pavimentou o caminho para o poderoso GPT-3 (2020) e, finalmente, para o fenômeno global ChatGPT (2022), que demonstrou capacidades impressionantes de geração e compreensão de linguagem. Em 2023, o Google Bard ampliou ainda mais o ecossistema, aplicando modelos como o LaMDA para diálogos abertos e multilingues, solidificando uma nova era de interação homem-máquina (AL-AMIN et al., 2024). Este percurso histórico ilustra uma trajetória evolutiva que partiu de regras simples e chegou a sistemas complexos capazes de diálogos contextualizados e criativos.

A arquitetura Transformer, introduzida em 2017 com o artigo "Attention Is All You Need", representou uma verdadeira revolução no campo da inteligência artificial generativa. Esse modelo inovador substituiu as limitações das redes neurais recorrentes ao implementar um mecanismo de autoatenção, que permite analisar relações contextuais em textos longos de maneira muito mais eficiente. Essa inovação tornou-se a base para os grandes modelos de linguagem que conhecemos hoje, como a família GPT da OpenAI e o Bart do Google, capazes de processar e gerar linguagem com impressionante fluência e coerência (Castro & Maciel, 2024).

A aplicação dos transformers permitiu avanços extraordinários, desde chatbots conversacionais como ChatGPT e Bard até sistemas multimodais que geram imagens, código e até música a partir de descrições textuais. Sua arquitetura escalável possibilitou o treinamento de modelos cada vez maiores, como o GPT-4 com seus trilhões de parâmetros, oferecendo capacidades sem precedentes em tradução automática, sumarização de textos e compreensão contextual. Essa tecnologia transformou radicalmente setores como atendimento ao cliente, educação e pesquisa acadêmica, criando assistentes virtuais com habilidades quase humanas COSTA (2024).

O impacto dos transformers se estende além do texto, influenciando áreas como visão computacional e processamento de áudio, enquanto continua a evoluir com técnicas mais eficientes e aplicações inovadoras. Eles representam o alicerce da atual revolução em IA generativa, abrindo caminho para sistemas cada vez mais sofisticados e integrados em nosso cotidiano COSTA (2024).

A ascensão da IA generativa tem gerado debates profundos sobre seu impacto no mercado de trabalho, particularmente em áreas como desenvolvimento web. Relatos indicam que um em cada quatro empregos está em risco devido à automação (Sakamoto, 2025), abrangendo funções técnicas como programação e suporte ao cliente. Startups de IA focadas em geração de código, por exemplo, têm alcançado avaliações financeiras elevadas (Thong, 2025), sinalizando um mercado ávido por soluções que substituam tarefas humanas. Esse cenário exige uma reavaliação urgente das competências profissionais no setor de tecnologia. Um fenômeno recente e crucial para essa discussão é o impacto desproporcional sobre os recém-formados. Conforme investigado por Scheiber (2025), formandos universitários estão enfrentando um mercado de trabalho transformado, onde muitas das posições de entrada que os aguardavam estão sendo automatizadas ou redefinidas, forçando-os a adaptar suas expectativas e buscar habilidades que complementam, e não somente compitam, com as capacidades da IA.

Os recém-formados em áreas como programação estão entre os mais afetados. Cortiz (2025) destaca que a IA está "implodindo" carreiras iniciantes, pois empresas preferem ferramentas automatizadas para tarefas básicas, como *debug* e escrita de código simples. Essa tendência pressiona os profissionais a se especializarem em funções menos replicáveis por IA, como arquitetura de sistemas ou gestão de projetos, reforçando a necessidade de adaptação contínua.

Contradições também emergem: enquanto algumas empresas investem pesadamente em IA, outras abandonam projetos devido a resultados insatisfatórios. Um exemplo é o caso de fintechs que desistiram de *chatbots* de atendimento ao cliente após problemas de eficiência, optando por contratar trabalhadores “uberizados” (Barros, 2025). Isso revela que, embora a IA prometa eficiência, sua aplicação prática ainda enfrenta limitações significativas, especialmente em contextos que exigem nuances humanas, como interações complexas com usuários finais.

Apesar dos desafios, há iniciativas para corrigir os excessos da substituição indiscriminada por IA. Empresas que demitiram funcionários para adotar soluções automatizadas estão revendo decisões após falhas operacionais (AL-SIBAI, 2025), como erros em código gerado por IA ou interfaces pouco intuitivas. Esses casos reforçam a tese central de que

a IA deve atuar como complemento — não substituto — ao trabalho humano, especialmente em desenvolvimento web, onde a qualidade final depende da combinação insubstituível entre eficiência técnica e criatividade humana.

A ascensão da IA generativa tem redefinido o desenvolvimento de software, prometendo acelerar a criação de aplicações web a partir de descrições textuais. No entanto, a qualidade, a consistência e a viabilidade de implantação desses sistemas gerados automaticamente permanecem como questões em aberto.

Este artigo tem como objetivo central avaliar uma ferramenta de desenvolvimento de código/sistema no intuito de verificar as vantagens e desvantagens da utilização da IA no contexto do desenvolvimento web. Para tanto, utilizou-se o v0 da Vercel, que é uma plataforma especializada que combina modelos de linguagem com um fluxo de trabalho otimizado para gerar interfaces React/Next.js implantáveis, em tese, rapidamente. A análise busca verificar:

- Se o código gerado é funcional, idiomático e aderente às boas práticas do ecossistema;
- Como a ferramenta lida com iterações e refinamentos via prompt;
- A eficácia do *deploy* automatizado integrado;
- As implicações desse nível de automação para desenvolvedores e o mercado.

O estudo adota uma abordagem prática: a partir de prompts progressivos, foi gerada uma aplicação web completa, documentando cada etapa do processo sob a ótica de um profissional de desenvolvimento, refletindo, à cada passo, se um leigo em computação seria capaz de desenvolver uma aplicação. Para tanto, fez-se necessário definir os instrumentos e o método que permitiram tal investigação.

Dentre as IAs generativas, foi escolhida a V0 para o desenvolvimento deste estudo de caso, por entender-se que ela é apropriada aos objetivos deste estudo, que é o desenvolvimento de uma aplicação web via *prompt*, para que a ferramenta não só gere o código, mas também construa o ambiente, como a organização e o versionamento dos arquivos automaticamente. Em outras IAs generativas como ChatGPT ou DeepSeek seria necessário que o usuário ficasse copiando e colando o código em ambientes de desenvolvimento (IDEs), estruturando pastas e organizando arquivos, o que atrasaria a execução do projeto. Interações futuras poderiam sair do contexto inicialmente proposto sem o versionamento e histórico de prompt. O V0 faz o versionamento de código, o que é extremamente importante e necessário para desfazer alterações, caso necessário.

MATERIAL E MÉTODOS

Os instrumentos e o método que sustentam esta investigação estão fundamentados em três pilares: a categorização do ecossistema de ferramentas de geração por IA, a seleção estratégica de um ferramenta e/ou plataforma de desenvolvimento como base, e a avaliação criteriosa da ferramenta escolhida, no caso a v0 da Vercel. Em resumo, a metodologia foi assim definida:

- levantamento, comparação e seleção de quais plataformas de IA generativas seriam utilizadas;
- a consulta nessas plataformas de IA sobre idéias de aplicativo para ser implementado;
- o refinamento, via IA, de um texto para uma especificação do software escolhido na etapa anterior;
- a elaboração de um documento de especificação de requisitos
- desenvolvimento clássico de sistemas: modelagem, implementação e testes;
- verificação de ergonomia
- reflexões sobre o processo.

O processo de escolha da IA envolveu desde as de uso genérico, como chat GPT e o Deep Seek, por exemplo, como as de uso específico para o desenvolvimento de software, como as embutidas em IDEs, como a V0, por exemplo. A V0 foi escolhida por apresentar robustez e integração com as melhores práticas e frameworks disponíveis no mercado atualmente.

Uma vez definida a IA, partiu-se para a definição da aplicação exemplo, que deveria ser atrativa e relevante no cenário social, educacional ou ambiental, por exemplo. A aplicação escolhida foi um site de achados e perdidos, uma aplicação com relevância social. Nessa fase utilizamos uma IA genérica, o DeepSeek, que ajudou não só a listar uma série de possibilidades, mas ajudou também, uma vez definida a aplicação de achados e perdidos, a refinar a concepção do sistema. A seguir, essa concepção foi detalhada em um documento de especificação de requisitos técnico, segundo modelo IEEE 830 (IEEE,

2025). Já a aplicação em si foi avaliada em termos de qualidade da interface com o usuário, facilidade de operação e acessibilidade. Em suma, os resultados gerados pela IA em cada um desses passos foram analisados quanto à sua completude, corretude e qualidade.

No contexto da aplicação de Inteligência Artificial ao desenvolvimento web, este estudo parte do pressuposto de que a IA pode gerar código automaticamente para uma aplicação desejada, com base em um *prompt* textual fornecido pelo usuário. Embora exista também o potencial da IA para identificar e corrigir erros em aplicações existentes, o foco deste artigo está especificamente na geração de código a partir do zero.

A automação na geração de código não é um conceito novo, precedida por anos de ferramentas de low-code e no-code, como Webflow e Bubble, por exemplo. A mudança radical, contudo, deu-se com a capacidade dos modelos de entender a linguagem natural e traduzi-la em código funcional e complexo, um avanço catapultado pelo advento dos Large Language Models (LLMs), como o GPT-4, Claude 3, e modelos especializados em código, como o Codex, base do GitHub Copilot (Bem, 2024).

A evolução dessa tecnologia segue uma trajetória clara: começou com o autocompletar inteligente (ex.: Copilot, 2021), que sugere a próxima linha ou função; avançou para a geração de componentes de interfaces com o usuário (UI) complexos (ex.: v0, 2023/2024); e progrediu para a geração de aplicações completas (ex.: GPT Engineer, Smithery, 2023), criando estruturas de projeto com múltiplos arquivos a partir de uma descrição de alto nível. O estágio mais recente envolve agentes autônomos (ex: Devon, Aider, 2024+), onde a IA não apenas gera código, mas também planeja, executa, depura e itera em um projeto de forma independente (Bem, 2024).

Para uma análise organizada, as ferramentas disponíveis podem ser divididas em três categorias principais. A Categoria 1 engloba os assistentes de IA de propósito geral integrados ao ambiente de desenvolvimento (IDE), como o Cursor Companion, o Claude Code da Anthropic e o pioneiro GitHub Copilot, que atuam principalmente na sugestão e geração de trechos de código dentro do fluxo do desenvolvedor. A Categoria 2 compreende os geradores de aplicações completas ou agentes de IA, como o GPT Engineer, o Smithery e o Aider, que possuem o escopo ambicioso de criar ou modificar bases de projeto inteiras a partir de uma especificação. Por fim, a Categoria 3 inclui plataformas de low-code e no-code que incorporaram IA, como DhiWise, Builder.io, Appsmith e Retool, utilizando inteligência artificial para converter designs em código ou gerar lógica de aplicação a partir de descrições textuais. O v0, da Vercel, enquadra-se nessa última categoria como um exemplo do fenômeno dos 'code-gen startups' mencionado na reportagem da Reuters: ferramentas que automatizam a escrita de código para desenvolvedores, diferenciando-se das plataformas low-code tradicionais por gerar componentes prontos para produção em vez de abstrair o desenvolvedor em interfaces visuais.

Dentre os modelos especializados, destaca-se o DeepSeek Coder, um LLM treinado em mais de 1 trilhão de tokens de código, abrangendo 87 linguagens de programação. Projetado para ser um especialista em geração, preenchimento e discussão de código, ele se aproxima da funcionalidade de ferramentas como o GPT Engineer por sua capacidade de gerar código contextual, entender especificações complexas e planejar tarefas de desenvolvimento de forma estruturada. Seu modelo sucessor, o DeepSeek-V2, embora generalista, mantém uma capacidade excepcional em raciocínio e codificação, suportando um contexto de 128K tokens. Isso permite analisar e gerar grandes volumes de código de uma só vez, mantendo a consistência em um projeto completo — uma capacidade crítica para atuar como um "engenheiro de IA" eficaz. Através de sua interface comum, o DeepSeek já fornece acesso a ambos os modelos, funcionando como um equivalente ao GPT Engineer sem a necessidade de uma implementação separada.(DEEPSEEK)

No cerne da geração de código com IA para aplicações web modernas, observa-se uma supremacia estratégica na adoção de frameworks full-stack. Quando solicitada a criar uma aplicação completa, a IA tende naturalmente a gerar blocos de código separados para *frontend* e *backend*. Embora válida, essa abordagem desacoplada impõe uma série de complexidades ao desenvolvedor individual ou à equipe multidisciplinar, incluindo a difícil integração entre dois projetos independentes, a sobrecarga de configuração de ambientes e fluxos distintos, e a complicação operacional de realizar *deploys* separados em hosts diferentes.

Portanto, a escolha da tecnologia base sobre a qual a IA gera o código é fundamental. A orientação para o uso de um framework full-stack moderno, como o Next.js, Nuxt ou SvelteKit, constitui uma estratégia metodológica central deste estudo para mitigar esses riscos e acelerar o ciclo de desenvolvimento. Esses frameworks unificam *frontend* e *backend* em uma

única base de código coesa, oferecendo benefícios tangíveis quando utilizados em conjunto com LLMs:

- unificação de contexto e redução de erros, pois a IA gera código dentro de um conjunto padronizado de regras, simplificando a comunicação entre as camadas da aplicação;
- otimização para *deploy* simplificado, uma vez que a aplicação é implantada como um projeto único em plataformas como Vercel ou Netlify, abstraindo a complexidade de gerenciar múltiplos servidores;
- maior coerência e manutenibilidade, já que todas as partes compartilham dependências e configurações; e
- geração de código mais inteligente e contextualizada, onde prompts mais precisos (ex.: "gere uma página Next.js com uma API Route") direcionam a IA a produzir soluções otimizadas e idiomáticas do framework, resultando em código não apenas funcional, mas também moderno e performático. Esta metodologia orienta a avaliação e discussão dos resultados apresentados neste trabalho.

Este estudo parte do pressuposto de que a IA pode gerar aplicações web completas a partir de prompts textuais, com foco na ferramenta v0 da Vercel. Embora o v0 utilize em seu núcleo modelos de linguagem como o DeepSeek Coder, sua proposta difere por ser especializada e opinativa: ele não apenas gera código, mas produz projetos Next.js com Tailwind CSS, integrando *deploy* automático via Vercel. O v0 é selecionado como objeto de estudo por representar uma ferramenta integrada e prática do conceito “do prompt à produção”.

A geração de código por IA não se resume apenas ao ato de produzir linhas de programação; trata-se de produzir um artefato funcional, implantável e sustentável. A escolha estratégica de uma ferramenta especializada como o v0, atua como um *guard rail* arquitetural. Ele guia a IA a produzir aplicações que são não apenas mais corretas por design, mas também radicalmente mais simples de publicar e gerenciar. Para este estudo, o v0 personifica a próxima fronteira na geração de software: a transição de "gerar código" para "gerar uma aplicação em produção" diretamente de um prompt.

O v0 é um motor de geração construído com o propósito específico de criar interfaces de usuário funcionais e implantáveis usando as melhores práticas do ecossistema Next.js, React e Tailwind CSS. Sua arquitetura oferece vantagens metodológicas críticas para esta pesquisa (VERCEL, 2026):

- geração com contexto estruturado e opinado**:** otimizado para gerar exclusivamente componentes React com Tailwind CSS seguindo as convenções do App Router do Next.js, eliminando variabilidade indesejada e garantindo compatibilidade nativa com a plataforma Vercel;
- abstração do fluxo de desenvolvimento: a ferramenta integra geração interativa em tempo real (permitindo refinamentos via prompts subsequentes) com *deploy* automático em um clique para uma URL pública, encurtando o ciclo "ideia para produção" para segundos;
- qualidade e consistência do código gerado: utiliza componentes de bibliotecas de alta qualidade como shadcn/ui, garantindo que o código produzido seja limpo, acessível, estilizado de forma consistente e fácil de manter; e
- alinhamento estratégico com a plataforma: a aplicação gerada é um projeto Next.js genuíno, pronto para evoluir com funcionalidades full-stack (API Routes, renderização híbrida), fornecendo uma fundação sólida para o experimento.

A tabela 1 faz uma comparação das IAs mais populares que temos atualmente disponíveis, e destaca as diferenças entre elas.

Tabela 1 - Comparativo das IAs no contexto de desenvolvimento de código

Nome	Empresa	Gratuito	Tipo	Transparência
ChatGPT	OpenAI	Parcial	Genérica	Não
DeepSeek	DeepSeek	Parcial	Genérica	Sim
Gemini	Google	Parcial	Genérica	Parcial
V0	Vercel	Parcial	Específica	Sim

A avaliação do sistema seguirá os seguintes critérios operacionais:

- funcionalidade: se a aplicação gerada atende integralmente aos requisitos especificados no prompt;

- qualidade do código: se o código adere às convenções idiomáticas do Next.js e React, é limpo e sustentável;
- iterabilidade: se a ferramenta permite ajustes coerentes e cumulativos via novos prompts, mantendo a integridade do projeto; e
- deploy: se o processo de publicação é contínuo, simples e resulta em uma aplicação acessível e funcional.

O procedimento experimental consistiu em uma sequência de interações iterativas com a ferramenta v0. Foi solicitada a criação de uma aplicação web progressivamente complexa, partindo de uma descrição inicial simples. Cada interação (prompt de entrada, resposta da ferramenta, estado do código e do deploy gerado) foi documentada de forma sistemática, registrando tanto os sucessos quanto os eventuais gaps, inconsistências ou necessidades de intervenção manual. Esta documentação formou a base empírica para a análise dos resultados.

RESULTADOS E DISCUSSÃO

Inicialmente desenvolveu-se um *prompt* que solicitou um sistema para o V0. Como descrito na introdução, a ideia aqui foi simular o pedido de um sistema como se fosse um usuário leigo solicitando. O *prompt* ficou assim:

“Crie um site para um sistema de achados e perdidos. O site deve possuir recursos de acessibilidade e deve se adequar a diferentes tamanhos de telas e dispositivos. A página inicial deve ter um botão “Perdi um item”, e deve mostrar os itens perdidos cadastrados. Em cada item perdido, deve haver um botão “Encontrei”. Para cadastrar um item perdido, a pessoa deve ter um cadastro no site. Com finalidade de facilitar o uso, o cadastro exigirá somente o nome da pessoa, e-mail e telefone/WhatsApp. Os dados necessários para o cadastro podem ser expandidos depois. Se o usuário estiver logado, ao clicar no botão “Perdi um item”, ele já deve ser levado para a tela de cadastro do item perdido. Se não, para a página de cadastro. Se já estiver logado, ao clicar no botão “Encontrei”, o sistema já deve mandar as informações da pessoa que clicou em encontrei para o e-mail da pessoa que perdeu, e emitir uma mensagem "Suas informações foram enviadas para a pessoa que perdeu o item". Se não estiver logado, deve mandar para a página de cadastro. A página de cadastro de itens perdidos, deve conter os seguintes campos: Título, que deve ser uma descrição breve e direta do item perdido, com 20 caracteres no máximo. Descrição, com no máximo 50 caracteres. Foto, que deve ser uma foto do item perdido, se tiver. Categoria, que define um conjunto de categorias mais comuns. Data, que vai ser a data que perdeu o item. Localização de Referência, que vai ser uma descrição do local onde perdeu o item, com 20 caracteres. O usuário deve poder acessar o sistema e marcar um item perdido como encontrado também. Os itens encontrados não aparecerão mais na página inicial.”

Após a inserção desse prompt, o V0 começou a pensar e criar a aplicação. Durante a criação, aparecem algumas interações e confirmamos a execução de *scripts* de criação do banco de dados. Em menos de 5 minutos, o V0 nos retorna a mensagem de informação da aplicação criada e disponível para download (figuras 1 e 2), com código versionado (controle de versões), e pronta para visualização dentro da própria ferramenta. O V0 cria a aplicação já replicando as práticas mais comuns de mercado para o tipo de sistema requisitado.

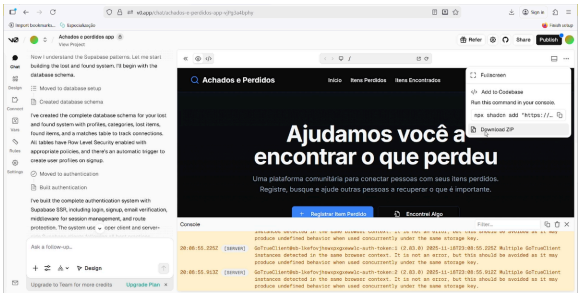

Figura 1 - Mensagem de criação da aplicação finalizada e opção de fazer download do código.

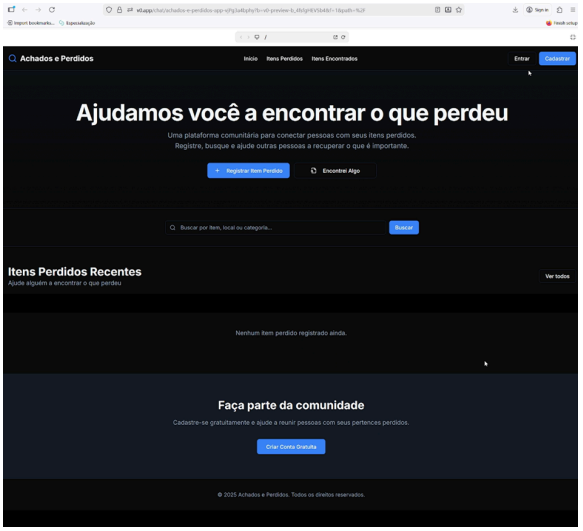

Figura 2 - Página inicial criada.

Além da página inicial da aplicação (figura 2), já foram criadas automaticamente a tela de login, de cadastro, e inclusive a tela que informa envio do email para confirmação após criação da conta.

Após a criação da aplicação, testou-se o principal ponto de entrada da aplicação, que é a criação de uma conta. Aqui foi encontrado um primeiro *bug* na aplicação. Mesmo após criar conta e validá-la por

email, ao tentar logar na conta no site, o login não acontece. Ao digitar e-mail e senha e clicar em "Entrar", o texto do botão muda para "Entrando", mas não avança. Sendo assim, partimos para uma segunda interação com o V0 para a resolução do problema encontrado.

Percebeu-se, nas iterações que foram feitas com o V0, que ao informar o erro ao logar, o V0 identificou que os scripts SQL para criar o banco de dados não foram executados, sendo que ao longo da criação da aplicação foi dada a permissão para executar os *scripts*.
Além disso, adicionalmente, a verificação identificou que havia um erro também no caminho para acesso ao banco de dados. O V0 identificou que era melhor usar outra forma de acesso, sendo que foi o próprio V0 que criou a aplicação.

Vemos aqui, portanto, que o V0 também comete erros ao criar as aplicações, inclusive corrige a si mesmo depois. Isso pode ser visto como bom, mas vem a pergunta: Porque não fez desde o início a melhor implementação?

É óbvio que qualquer ferramenta está em constante evolução, mas esperamos que ela sempre aplique as melhores práticas desde a primeira interação.

O próprio V0 cuidou de corrigir os erros, pedindo somente ao usuário a confirmação da criação do banco de dados, pois se o banco já estivesse populado, poderia haver alguma perda de dados.

Uma vez que o login foi corrigido e operacional, partiu-se para outras verificações. Como os testes de adequação do site a diferentes tipos de dispositivos, testando por exemplo se ele se adapta à tela de um celular. Acionamos a ferramenta de desenvolvimento do browser, e simulamos a tela de um celular. O resultado é que a tela se adequou corretamente ao celular (figura 3), por mais que considerasse que poderiam haver algumas melhorias ainda, como a redução do tamanho da fonte, quando o site fosse exibido no celular.

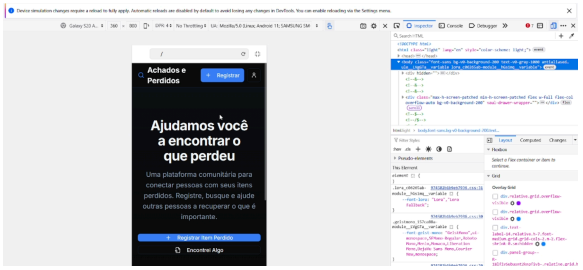

Figura 3 - Exibição no celular.

Ainda na questão de usabilidade e acessibilidade, pedimos no prompt inicial que o sistema deve possuir recursos de acessibilidade, como botões grandes, alto contraste, legendas e outros recursos previstos no padrão ARIA, o que não foi entregue pela ferramenta. Foi necessário uma nova interação com o V0 para pedir essa alteração.

A figura 4 exibe a resposta do V0 ao solicitado. O V0 incluiu um menu com três opções, onde se pode escolher o esquema de cores padrão, alto contraste ou alto contraste escuro. Porém o resultado não agradou a equipe, e o V0 foi solicitado a melhorar as opções de visual.

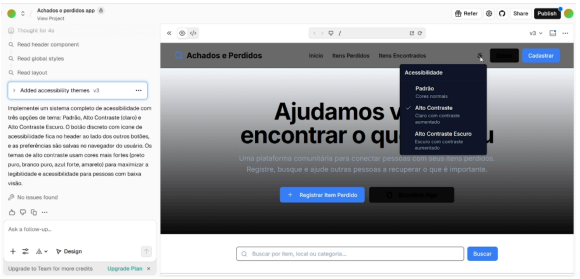

Figura 4 - Exibição da tela em alto contraste.

A resposta do V0 foi uma promessa de melhora no visual do site, com a aplicação de gradientes sutis e modernos, sombras suaves e transições animadas nos cards, paleta de cores ficou mais rica, inclusão de ícones de destaque na página inicial,cores mais equilibradas e agradáveis, garantindo-se boa legibilidade e acessibilidade. Porém o resultado ainda não tinha agradado a equipe de desenvolvimento (autores deste trabalho). Foi necessária, então, uma terceira interação com o V0 para tentar melhorar o visual. Ainda sim, sem sucesso. Foram necessárias mais três interações via prompt até que uma solução satisfatória fosse alcançada.

Percebeu-se, nesse ponto do projeto, que muitas vezes é preciso lapidar sucessivamente o prompt até que IA entregue o que se deseja. Talvez uma engenharia de prompt mais completa traga melhores resultados, como a escolha da paleta de cores seja já feita pelo usuário. A noção de que a IA pode entregar um sistema inteiro e pronto, por mais simples que seja o site, ainda é, na prática, algo questionável. Um bom prompt deve ser o mais simples e preciso possível, não dando margem a duplas interpretações ou com lacunas que possam ser resolvidas por conta própria pela IA.

Após os ajustes no visual, foram exploradas as funcionalidades do sistema, começando por testar o cadastro de um item perdido. Ao clicar-se no botão "Registrar Item Perdido", tem-se acesso à tela de cadastro (figura 5). Ao digitar os dados solicitados, o cadastro ocorreu normalmente. Porém ao clicar no item para acessar o item cadastrado, o sistema apresentou o erro "404 - The page could not be found." O erro foi informado ao V0, que informou ter corrigido o problema, porém o mesmo erro persistiu. Foi necessário abrir o log de eventos no navegador e

informar ao V0. Após algumas tentativas, o V0 conseguiu resolver o problema.

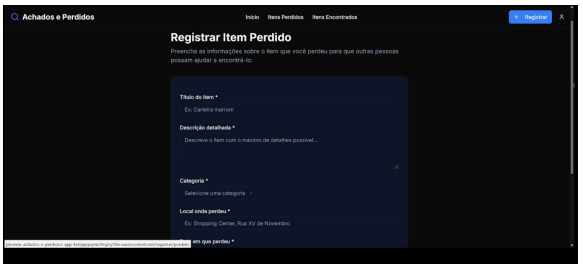

Figura 5 - Tela de cadastro de item perdido.

Algo similar ocorreu com as outras funcionalidades. Ou seja, erros pontuais ocorreram e foram corrigidos pela IA.

A equipe não mencionou estratégias ou ferramentas para a persistência de dados, como frameworks, SGBDs ou mesmo acesso via nuvem. O V0 decidiu, por conta própria, utilizar o Supabase como camada de persistência. O Supabase é uma plataforma de desenvolvimento baseado em SGBD Postgre, que permite propiciar ao projeto persistência em banco de dados relacional, acesso via nuvem, ou seja, banco de dados já hospedado, autenticação de usuários, APIs, dentre outros recursos.

CONCLUSÃO

Este artigo teve como objetivo avaliar se um usuário leigo em programação seria capaz de desenvolver um sistema web funcional utilizando a ferramenta V0 da Vercel, baseada em IA generativa. A partir da solicitação de um sistema de achados e perdidos, observou-se que, embora a ferramenta tenha gerado uma estrutura inicial coerente em poucos minutos, foram necessárias múltiplas interações para corrigir erros básicos — como falha no login, problemas de rota e inconsistências visuais — mesmo após a IA afirmar que os problemas haviam sido resolvidos.

Em alguns casos, o erro persistiu e exigiu que o usuário coletasse *logs* do navegador, o que inviabilizaria o uso por uma pessoa sem conhecimento técnico. Conclui-se que, no estágio atual, a IA ainda não é capaz de entregar um sistema completo e funcional a partir de um único prompt, especialmente quando o escopo envolve múltiplas funcionalidades e regras de negócio. A ferramenta atua como um acelerador para desenvolvedores experientes, mas não substitui a necessidade de conhecimento técnico para identificar, interpretar e corrigir falhas. Quanto ao mercado de trabalho, os resultados reforçam a tese de que a IA tende a eliminar funções mais operacionais, mas amplia a demanda por profissionais capazes de orquestrar, revisar e refinar o código gerado automaticamente — ou seja, exige-se mais qualificação, não menos.

Conforme o relatado neste trabalho, toda resposta da ferramenta V0 precisou ser meticulosamente analisada. A inclusão de funcionalidades ou outras peculiaridades não solicitadas diretamente pelo usuário pode ferir os requisitos do sistema. Outra ocorrência comum foi a alucinação, ou seja, uma resposta sem nexo ou claramente equivocada.

Tais problemas surgiram em um contexto de uma aplicação simples, um site de achados e perdidos, que é basicamente um CRUD (criação, leitura, atualização e remoção de dados). Em casos mais complexos, apesar do ambiente V0 conter um versionador de código, esses problemas podem se agravar, podendo se tornar um grande desafio logístico e metodológico.

A IA demonstrou, entretanto, pelo menos neste projeto, ser uma ferramenta poderosa, seja na prospecção de necessidades de sistemas, na definição e refinamento da sua concepção, modelagem, produção de código, definição da camada de persistência ou mesmo na interface gráfica com o usuário, mesmo frente aos problemas vividos e relatados, trazendo agilidade, liberdade em relação à necessidade de conhecimentos sobre detalhes/sintaxe de diferentes linguagens de programação e frameworks, além de configurações de ambientes. O que traz boas perspectivas para o processo de desenvolvimento de software, em questões de agilidade e qualidade, mas trazendo também preocupações em relação ao mercado de trabalho, especificamente para os inexperientes. Traz também desafios logísticos e metodológicos para a sua utilização em projetos complexos, onde, ao que parece, deva ser utilizada na realização de tarefas simples e pontuais.

Sobre o artefato implementado, um site de achados e perdidos, espera-se que ele seja testado, também com o auxílio da IA, para que possa ser refinado, hospedado e que entre em operação para que possa cumprir o papel social a que foi projetado.

AGRADECIMENTOS

Registro meus sinceros agradecimentos ao meu orientador, Prof. Sergio Muinhos Barroso Lima, cuja atenção, disponibilidade e instruções claras foram fundamentais para a construção deste artigo. Suas contribuições precisas e seu acompanhamento dedicado foram essenciais em todas as etapas da pesquisa e escrita desse trabalho.

REFERÊNCIAS

AL-AMIN, Md et al. History of generative Artificial Intelligence (AI) chatbots: past, present, and future development. arXiv preprint arXiv:2402.05122, 2024.

AL-SIBAI, Noor. Companies That Tried to Save Money With AI Are Now Spending a Fortune Hiring People to Fix Its Mistakes. Futurism, New York, 6 jul. 2025. Disponível em: https://futurism.com/companies-fixing-ai-replacement-mistakes. Acesso em: dezembro de 2025.

BARROS, Carlos Juliano. Fintech desiste de IA para atendimento a clientes e contrata 'uberizados'. UOL Economia, São Paulo, 3 jun. 2025. Disponível em: https://economia.uol.com.br/colunas/carlos-juliano-barros/2025/06/03/fintech-desiste-de-ia-para-atendimento-a-clientes-e-contrata-uberizados.htm. Acesso em: novembro de 2025.

BEM, Rafael Almeida de. Uso de IA generativa no apoio à aprendizagem de programação. 2024. Disponível em: https://repositorio.pucrs.br/dspace/bitstream/10923/26764/3/2024_1_RAFAEL_ALMEIDA_DE_BEM_TCC.pdf. Acesso em janeiro de 2026.

CASTRO, Michele Marta Moraes; MACIEL, Cristiano. Historiográficos da Inteligência Artificial e suas Implicações na Educação. In: Anais Principais do Seminário de Educação (SemiEdu). SBC, 2024. p. 1273-1282.

CORTIZ, Diogo**.** Os primeiros afetados: IA está implodindo a carreira de recém-formados. UOL Tilt, São Paulo, 8 jun. 2025. Disponível em: https://www.uol.com.br/tilt/colunas/diogo-cortiz/2025/06/08/os-primeiros-afetados-ia-esta-implodindo-a-carreira-de-recem-formados.htm. Acesso em: janeiro de 2026.

COSTA, Albert França Josuá et al. inteligência artificial generativa. Disponível em: https://www.gov.br/anpd/pt-br/documentos-e-publicacoes/documentos-de-publicacoes/radar_tecnologico_ia_generativa_anpd.pdf. Acesso em maio de 2025.

DEEPSEEK-AI. DeepSeek Coder: When the Large Language Model Meets Programming. 2023. Disponível em: https://github.com/deepseek-ai/DeepSeek-Coder. Acesso em maio de 2025.

DEEPSEEK-AI. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. arXiv:2405.04434, 2024. Disponível em: https://arxiv.org/abs/2405.04434. Acesso em maio de 2025.

IEEE. Software Requirements Specifications. Disponível em: https://www.computer.org/resources/software-requirements-specifications. Acesso em: junho de 2025.

SAKAMOTO, Leonardo. Um em cada 4 empregos corre risco por causa da IA. Veja como está o seu. UOL, São Paulo, 20 maio 2025. Disponível em: https://noticias.uol.com.br/colunas/leonardo-sakamoto/2025/05/20/um-em-cada-4-empregos-corre-risco-por-causa-da-ia-veja-como-esta-o-seu.htm. Acesso em: janeiro de 2026.

SCHEIBER, Noam. For Some Recent Graduates, the A.I. Job Apocalypse May Already Be Here. The New York Times, New York, 30 maio 2025. Disponível em: https://www.nytimes.com/2025/05/30/technology/ai-jobs-college-graduates.html. Acesso em: dezembro de 2025.

SUPABASE. Supabase: The Open Source Firebase Alternative. Open source backend as a service platform built on PostgreSQL. Disponível em: https://supabase.com. Acesso em: março de 2026.

TONG, Anna; HU, Krystal. AI startups revolutionize coding industry, leading to sky-high valuations. Reuters, São Francisco, 3 jun. 2025. Disponível em: https://www.reuters.com/business/ai-vibe-coding-startups-burst-onto-scene-with-sky-high-valuations-2025-06-03/. Acesso em: janeiro de 2026.

VERCEL. v0 Documentation. Disponível em: https://v0.dev/docs. Acesso em: janeiro de 2026.

VERCEL. How we made v0 an effective coding agent. 2026. Disponível em: https://vercel.com/blog/how-we-made-v0-an-effective-coding-agent. Acesso em janeiro de 2026.

VERCEL. Introducing the v0 composite model family. 2025. Disponível em: https://vercel.com/blog/how-we-built-the-v0-composite-model. Acesso em janeiro de 2026.

WORLD WIDE WEB CONSORTIUM. Accessible Rich Internet Applications (WAI-ARIA) 1.2. W3C Recommendation, 06 June 2023. Disponível em: https://www.w3.org/TR/wai-aria-1.2/. Acesso em: janeiro de 2026.