Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Uma avaliação prática de plataformas low-code e no-code frente ao cenário atual de desenvolvimento de software

Laura Neves Pires

JUIZ DE FORA
JULHO, 2026

Uma avaliação prática de plataformas low-code e no-code frente ao cenário atual de desenvolvimento de software

Laura Neves Pires

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Sistemas de Informação

Orientador: Fabrício Martins Mendonça

JUIZ DE FORA

JULHO, 2026

Uma avaliação prática de plataformas low-code e no-code frente ao cenário atual de desenvolvimento de software

Laura Neves Pires

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Fabrício Martins Mendonça
Doutor em Ciência da Informação pela UFMG

Gleiph Ghiotto Lima de Menezes
Doutor em Computação pela UFF

Igor de Oliveira Knop
Doutor em Modelagem Computacional pela UFJF

JUIZ DE FORA
08 DE JULHO, 2026

Resumo

O desenvolvimento de software tem passado por transformações associadas à busca por maior agilidade, automação de tarefas e redução do esforço de codificação manual. Nesse contexto, plataformas *low-code* e *no-code* surgiram como alternativas voltadas à construção de aplicações com menor dependência de codificação tradicional, por meio de interfaces visuais, componentes reutilizáveis e recursos configuráveis. No entanto, com o avanço recente de ferramentas baseadas em inteligência artificial generativa, torna-se necessário reavaliar em quais condições essas plataformas ainda oferecem ganhos práticos, quais limitações apresentam e como sua proposta de valor se mantém no cenário atual. Este trabalho tem como objetivo avaliar o uso de plataformas *low-code* e *no-code* no desenvolvimento de software, com base em um sistema de métricas avaliativas aplicado à implementação de uma prova de conceito. Para isso, foi conduzida uma metodologia de avaliação em duas plataformas, *Bubble* e *Budibase*, a partir do desenvolvimento de uma PoC de um sistema de agendamento de consultas médicas. A análise considerou critérios como velocidade de desenvolvimento, modelagem de dados, integrações e necessidade de conhecimento técnico. Os resultados indicaram que a classificação de uma ferramenta como *low-code* ou *no-code* não garante, por si só, maior produtividade, menor complexidade ou menor exigência técnica. Observou-se ainda que ferramentas de inteligência artificial generativa podem tensionar a utilidade percebida dessas plataformas, especialmente quanto à velocidade inicial de geração e à qualidade visual percebida. Como limitação, destaca-se que a avaliação foi realizada com apenas uma prova de conceito e duas plataformas, restringindo a generalização dos resultados.

Palavras-chave: *Low-code*, *no-code*, desenvolvimento de software, métricas avaliativas, prova de conceito.

Abstract

Software development has undergone transformations associated with the search for greater agility, task automation, and reduced manual coding effort. In this context, *low-code* and *no-code* platforms have become alternatives for building applications with less dependence on traditional coding, through visual interfaces, reusable components, and configurable resources. However, with the recent advancement of generative artificial intelligence tools, it becomes necessary to reassess under which conditions these platforms still provide practical benefits, what limitations they present, and how their value proposition remains relevant in the current scenario. This study aims to evaluate the use of *low-code* and *no-code* platforms in software development based on a set of evaluation metrics applied to the implementation of a proof of concept. To this end, an evaluation methodology was conducted in two platforms, *Bubble* and *Budibase*, through the development of a PoC for a medical appointment scheduling system. The analysis considered criteria such as development speed, data modeling, integrations, and required technical knowledge. The results indicated that classifying a tool as *low-code* or *no-code* does not, by itself, guarantee higher productivity, lower complexity, or reduced technical requirements. It was also observed that generative artificial intelligence tools may challenge the perceived usefulness of these platforms, especially regarding initial generation speed and perceived visual quality. As a limitation, the evaluation was conducted with only one proof of concept and two platforms, which restricts the generalization of the results.

Keywords: *Low-code*, *no-code*, software development, proof of concept.

Agradecimentos

Ao meu orientador, professor Fabrício, expresso minha sincera gratidão pela orientação, pela disponibilidade e pela confiança ao longo desta trajetória. Suas contribuições, questionamentos e sugestões foram fundamentais para o desenvolvimento desta pesquisa e para meu crescimento acadêmico durante a construção deste trabalho.

Ao meu parceiro de vida, a quem também dedico este trabalho, agradeço por caminhar ao meu lado em cada etapa deste percurso. Obrigada pelo amor, apoio, pela paciência nos momentos mais difíceis e por celebrar comigo cada pequena conquista. Sua presença tornou os desafios mais leves, as inseguranças mais suportáveis e esta jornada muito mais especial. Esta conquista também é sua.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
PoC	Prova de Conceito
UI	User Interface (Interface do usuário)
UX	User Experience (Experiência do usuário)
CRUD	Acrônimo para Create (Criar), Read (Ler), Update (Atualizar) e Delete (Excluir)
API	*Application Programming Interface* (Interface de Programação de Aplicações)
IA	Inteligência Artificial
LLM	*Large Language Model* (Grande Modelo de Linguagem)

1 Introdução

Nesse capítulo, são apresentados o tema da pesquisa, a justificativa do trabalho, os objetivos propostos, as questões de pesquisa que orientam o estudo e uma introdução à metodologia adotada para sua realização.

1.1 Contextualização

As plataformas *low-code* e *no-code* permitem o desenvolvimento de aplicações com pouca ou nenhuma codificação manual, utilizando interfaces gráficas, modelos visuais e componentes reutilizáveis. Essas plataformas se inserem em um contexto mais amplo de transformação do desenvolvimento de software, no qual diferentes ferramentas buscam reduzir esforço manual, acelerar etapas de construção e tornar o processo de desenvolvimento mais acessível.

Essas plataformas funcionam a partir da definição visual de fluxos de trabalho, modelos de dados e regras de negócio, que são interpretados ou executados por um ambiente próprio da ferramenta. Em vez de exigir que todo o código-fonte seja escrito manualmente pelo desenvolvedor, essas soluções permitem que parte da aplicação seja construída por meio de abstrações visuais, automações e componentes previamente disponibilizados pela plataforma.

Entre as funcionalidades mais recorrentes dessas plataformas, destacam-se a criação de interfaces visuais (*UI*), o suporte a operações de banco de dados que envolvem leitura, escrita, atualização e exclusão de informações (*CRUD*), mecanismos de autenticação, integração com serviços externos via *Application Programming Interfaces* (APIs) e recursos de automação de processos, conforme descrito em (LUO et al., 2021).

De forma complementar, Martins, Branco e Mamede (MARTINS; BRANCO; MAMEDE, 2023) descrevem o desenvolvimento *low-code* como um paradigma no qual a construção de aplicações ocorre com menor quantidade de codificação manual, por meio de interfaces gráficas, elementos pré-construídos, objetos de dados e componentes de negócio.

Essa caracterização reforça que plataformas *low-code* e *no-code* não devem ser compreendidas apenas como ferramentas de criação visual de telas, mas como ambientes que integram modelagem de dados, lógica de negócio, componentes reutilizáveis e mecanismos de automação. Ao mesmo tempo, os autores também indicam que essas plataformas podem apresentar limitações relacionadas à liberdade de customização, documentação, visão geral da aplicação e colaboração, o que evidencia a necessidade de avaliá-las em cenários práticos de desenvolvimento.

Essa combinação de recursos permite compreender as plataformas *low-code* e *no-code* como ambientes de desenvolvimento apoiados por componentes visuais, configurações e automações. Em tese, essas características podem facilitar a participação de diferentes perfis profissionais no processo de construção de soluções, incluindo não apenas desenvolvedores, mas também profissionais de áreas de negócio. No entanto, essa participação depende do nível de complexidade da aplicação, das limitações da plataforma utilizada e do conhecimento técnico exigido em cada contexto.

Nos últimos anos, entretanto, o cenário de desenvolvimento de software passou também a ser impactado pelo avanço das ferramentas baseadas em inteligência artificial generativa. Soluções capazes de gerar código, interfaces, protótipos e aplicações a partir de instruções em linguagem natural passaram a disputar parte do espaço anteriormente associado às plataformas *low-code* e *no-code*. Esse movimento torna necessário revisitar a discussão sobre essas plataformas, não apenas para identificar seus ganhos e limitações, mas também para analisar se sua proposta de valor permanece relevante em um contexto no qual ferramentas de IA generativa também são utilizadas como estratégia de desenvolvimento.

Ainda assim, este trabalho não tem como foco principal avaliar ferramentas de IA generativa. A escolha por avaliar plataformas *low-code* e *no-code* se justifica pelo fato de que ainda há espaço na literatura científica para análises práticas e criteriosas sobre o uso dessas plataformas, especialmente em relação à sua utilidade, limitações e competitividade no cenário atual. Dessa forma, a IA generativa é considerada neste trabalho apenas como elemento complementar de discussão, permitindo contextualizar os resultados obtidos e levantar indícios sobre como essas novas ferramentas podem impactar a adoção de soluções

low-code e *no-code*.

1.2 Justificativa

Embora a literatura científica já apresente estudos sobre plataformas *low-code* e *no-code*, como discutido por (FRANK; MAIER; BOCK, 2021; FERREIRA, 2024; SILVA, 2024), o tema ainda carece de aprofundamento, especialmente em relação aos contextos mais adequados para sua utilização, suas limitações técnicas e os impactos em termos de segurança, escalabilidade e manutenibilidade, configurando-se como um problema de pesquisa na literatura da área.

Nesse sentido, torna-se relevante avaliar se as plataformas *low-code* e *no-code* ainda oferecem ganhos práticos no desenvolvimento de aplicações e em quais aspectos esses ganhos aparecem ou deixam de aparecer. A avaliação dessas plataformas permite observar, de forma mais concreta, se características frequentemente associadas a elas, como agilidade, simplicidade e redução de esforço manual, se confirmam durante a implementação de uma aplicação prática.

A motivação para o desenvolvimento desta pesquisa está relacionada à necessidade de analisar essas plataformas de forma estruturada, utilizando critérios avaliativos que permitam comparar diferentes aspectos do processo de desenvolvimento. Assim, este trabalho propõe a aplicação de um sistema de métricas avaliativas sobre uma prova de conceito, buscando identificar ganhos, limitações e desafios observados durante o uso de plataformas representativas das abordagens *low-code* e *no-code*.

Além disso, considerando o avanço recente das ferramentas de IA generativa, este trabalho inclui uma breve comparação complementar com uma aplicação gerada por IA. Essa comparação não tem como objetivo substituir a avaliação principal nem oferecer uma resposta definitiva sobre a utilidade ou obsolescência das plataformas *low-code* e *no-code*. Seu papel é fornecer indícios e contextualizar a discussão para um cenário atual, contribuindo assim em compreender como essas plataformas se posicionam em um cenário no qual a IA também passa a ser utilizada como estratégia de desenvolvimento de software.

1.3 Questões de pesquisa

A partir da análise do uso de plataformas *low-code* e *no-code* no desenvolvimento de software, este trabalho busca responder as seguintes questões de pesquisa:

RQ1: *Como realizar uma avaliação prática de plataformas low-code e no-code no desenvolvimento de software com base em um sistema de métricas avaliativas?*

RQ2: *Quais possíveis ganhos, limitações e desafios no uso de plataformas low-code e no-code podem ser identificados com uma avaliação prática de algumas dessas plataformas?*

RQ3: *Em que medida ferramentas de desenvolvimento baseadas em inteligência artificial generativa podem influenciar a utilidade percebida e a proposta de valor das plataformas low-code e no-code, considerando os resultados obtidos na avaliação realizada?*

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral deste trabalho é conduzir uma avaliação prática sobre o uso de plataformas *low-code* e *no-code* no desenvolvimento de software, com base em um sistema de métricas avaliativas aplicado à implementação de uma prova de conceito, a fim de identificar ganhos, limitações e desafios associados a essas estratégias.

1.4.2 Objetivos Específicos

Para atingir o objetivo geral, esta pesquisa busca:

- Definir um conjunto de critérios avaliativos para analisar plataformas *low-code* e *no-code* no desenvolvimento de software;
- Desenvolver uma prova de conceito representativa para apoiar a avaliação prática das plataformas selecionadas;
- Implementar a prova de conceito em pelo menos duas plataformas low-code/no-code, observando esforço de desenvolvimento, limitações, integrações, customização

e experiência de uso;

- Avaliar as plataformas selecionadas com base no sistema de métricas definido, registrando notas e justificativas qualitativas para cada critério;
- Discutir os resultados obtidos considerando o uso de plataformas low-code/no-code em um cenário no qual ferramentas de IA generativa também vêm sendo utilizadas como estratégia de desenvolvimento de software;

1.5 Metodologia

O processo metodológico adotado nesta pesquisa foi organizado em quatro etapas principais, que estruturam desde a revisão da literatura até a análise crítica dos resultados. O objetivo dessas etapas é fornecer um percurso claro e fundamentado para responder às questões de pesquisa previamente apresentadas na Seção 1.3. A descrição detalhada da metodologia de pesquisa deste trabalho será apresentada no Capítulo 4.

A primeira etapa consistiu na elaboração de estratégias de busca em inglês e em português, com o objetivo de identificar estudos relevantes sobre plataformas *low-code* e *no-code*. Essas estratégias foram posteriormente aplicadas em buscas realizadas nas bases *Google Scholar* e *Scite*. Os detalhes desse processo, incluindo as *strings* utilizadas e as etapas de seleção dos estudos, estão descritos no Capítulo 4, especialmente na subseção 4.1.2. O resultado desse processo foi a seleção de publicações que fundamentam o Capítulo 3.

A segunda etapa envolveu a identificação, organização e sistematização dos critérios analíticos relevantes para avaliação das plataformas *low-code* e *no-code*. Foram definidos critérios como flexibilidade, escalabilidade, suporte a integrações, governança, curva de aprendizado, entre outros. Também foi estruturada uma escala de complexidade para classificar cenários de desenvolvimento e analisar a viabilidade prática dessas tecnologias em diferentes níveis de demanda.

Na terceira etapa, foi conduzido um estudo de caráter prático. Para isso, foram selecionadas as plataformas *Bubble*, como representação do ecossistema *no-code*, e *Budibase*, como representação do ecossistema *low-code*. Nessas plataformas, foi desenvolvida a Prova de Conceito (PoC) de um sistema de agendamento de consultas médicas. Esta prova

de conceito consistiu na implementação de um fluxo funcional básico e na observação dos desafios técnicos, limitações e facilidades de cada plataforma. Essa etapa permitiu testar, em um ambiente controlado, como essas tecnologias se comportam diante de demandas reais de desenvolvimento.

Por fim, na quarta etapa, foi realizada a análise crítica dos resultados obtidos. As PoCs desenvolvidas foram avaliadas com base nos critérios previamente estabelecidos, permitindo comparar expectativas teóricas com evidências práticas e consolidar recomendações aplicáveis ao uso dessas estratégias em projetos reais de desenvolvimento de software.

2 Fundamentação Teórica

Este capítulo apresenta os principais conceitos necessários para compreender o contexto desta pesquisa. Inicialmente, são discutidos fundamentos da Engenharia de Software, incluindo processos de desenvolvimento, especificação, modelagem e metodologias ágeis. Em seguida, são apresentadas as plataformas *low-code* e *no-code*, destacando suas características, benefícios e limitações. Por fim, discute-se o uso de inteligência artificial generativa na Engenharia de Software e sua relação com o desenvolvimento apoiado por plataformas visuais.

2.1 Engenharia de software

A Engenharia de Software pode ser compreendida como uma área voltada à aplicação de métodos, processos, técnicas e ferramentas para o desenvolvimento, manutenção e evolução de sistemas de software. Diferentemente da programação entendida apenas como escrita de código, a Engenharia de Software envolve atividades como levantamento e análise de requisitos, projeto, implementação, testes, implantação, manutenção e gestão do processo de desenvolvimento (SOMMERVILLE, 2016; PRESSMAN; MAXIM, 2020).

O desenvolvimento de software é, portanto, uma atividade técnica e organizacional. Ele exige compreender problemas do mundo real, traduzi-los em requisitos, projetar soluções adequadas e implementar sistemas que possam ser mantidos e evoluídos ao longo do tempo. Por esse motivo, processos de software são utilizados para organizar o desenvolvimento e orientar a execução das atividades necessárias para transformar necessidades de usuários e organizações em sistemas funcionais.

Modelos tradicionais de desenvolvimento, como o modelo em cascata, organizam o processo em etapas sequenciais, geralmente envolvendo especificação de requisitos, projeto, implementação, testes e manutenção. Esse tipo de abordagem pode ser adequado em contextos nos quais os requisitos são relativamente estáveis e bem conhecidos desde o início. No entanto, em cenários sujeitos a mudanças frequentes, ciclos longos e rígidos

podem dificultar a adaptação do produto às necessidades dos usuários e do negócio (SOMMERVILLE, 2016; VALENTE, 2020).

Nesse contexto, as metodologias ágeis surgem como uma alternativa aos processos mais prescritivos. O Manifesto Ágil valoriza indivíduos e interações, software em funcionamento, colaboração com o cliente e resposta a mudanças (BECK et al., 2001). Em vez de concentrar o desenvolvimento em grandes entregas ao final de um ciclo longo, as abordagens ágeis favorecem entregas incrementais, feedback contínuo e adaptação ao longo do processo.

Essa discussão é relevante para este trabalho porque plataformas *low-code* e *no-code* se inserem em um cenário no qual organizações buscam acelerar entregas, reduzir esforço manual e aproximar profissionais de negócio do desenvolvimento de software. A promessa dessas plataformas dialoga com parte dos objetivos das metodologias ágeis, especialmente em relação à entrega rápida de soluções e à possibilidade de adaptação contínua. No entanto, a agilidade no desenvolvimento não depende apenas da velocidade de criação inicial da aplicação, mas também da facilidade de manutenção, evolução, integração, validação e controle técnico do sistema.

Outro aspecto central da Engenharia de Software é a especificação do software. Antes da implementação, é necessário compreender o problema a ser resolvido e definir quais funcionalidades, restrições e comportamentos o sistema deve apresentar. Essa etapa envolve a identificação de requisitos funcionais, que descrevem o que o sistema deve fazer, e requisitos não funcionais, que tratam de aspectos como desempenho, segurança, usabilidade, disponibilidade e manutenibilidade (PRESSMAN; MAXIM, 2020; SOMMERVILLE, 2016).

A modelagem também desempenha papel importante nesse processo. Modelos permitem representar partes do sistema de forma abstrata, facilitando a comunicação entre as pessoas envolvidas no projeto e apoiando a análise da solução antes ou durante sua implementação. A *Unified Modeling Language* (UML), por exemplo, oferece diferentes tipos de diagramas para representar aspectos estruturais e comportamentais de sistemas. Diagramas de casos de uso podem ser utilizados para representar funcionalidades percebidas pelos usuários e suas interações com o sistema, enquanto diagramas de

classes ou modelos de dados ajudam a representar entidades, atributos e relacionamentos (VALENTE, 2020).

Pacheco et al. (PACHECO et al., 2025) também destacam que o processo de Engenharia de Software costuma envolver etapas como planejamento, projeto, codificação e testes. Segundo os autores, requisitos, histórias de usuário e casos de uso são exemplos de artefatos frequentemente escritos em linguagem natural, enquanto a etapa de projeto pode utilizar diagramas, como os da UML, para representar modelos do sistema. Essa perspectiva reforça a importância de artefatos de especificação e modelagem mesmo em contextos nos quais o desenvolvimento busca ser mais rápido ou automatizado.

No contexto desta pesquisa, esses conceitos são importantes porque a Prova de Conceito (PoC) desenvolvida não foi construída apenas como um conjunto de telas, mas como um pequeno sistema com usuários, entidades, funcionalidades, fluxos e integrações. A definição prévia de casos de uso, entidades e critérios avaliativos permitiu observar as plataformas de forma mais estruturada, considerando não apenas a aparência da aplicação gerada, mas também sua capacidade de representar dados, executar operações de cadastro e edição, configurar automações e integrar-se a serviços externos.

2.2 Plataformas *low-code* e *no-code*

As plataformas *low-code* e *no-code* são ferramentas que buscam reduzir a quantidade de codificação manual necessária para o desenvolvimento de aplicações, utilizando interfaces gráficas, componentes pré-construídos, configurações visuais e mecanismos de automação (MARTINS; BRANCO; MAMEDE, 2023; BRUHIN et al., 2024; LUO et al., 2021). Em geral, essas plataformas oferecem interfaces visuais, componentes reutilizáveis, editores de fluxos, mecanismos de geração automática de telas, conectores com bases de dados e integração com serviços externos. A partir desses recursos, parte da construção do sistema é realizada por meio de configuração, composição de componentes e modelagem visual, em vez da escrita direta de código-fonte.

Do ponto de vista histórico, embora práticas relacionadas à programação visual, geração automática de código e desenvolvimento orientado a modelos já existissem anteriormente, o termo *low-code* passou a ser utilizado de forma mais consolidada a partir de

2014. Bock e Frank (BOCK; FRANK, 2021) afirmam que a expressão provavelmente foi cunhada pela *Forrester*[1], naquele ano. A partir disso, o termo passou a organizar um conjunto de plataformas voltadas à construção de aplicações com menor dependência de codificação manual, frequentemente associadas a interfaces visuais, componentes reutilizáveis e maior participação de usuários não necessariamente especializados em programação.

Embora os termos *low-code* e *no-code* sejam frequentemente utilizados em conjunto, eles não representam exatamente a mesma abordagem. Plataformas *low-code* costumam pressupor algum nível de conhecimento técnico e permitem maior personalização por meio de scripts, expressões, consultas, APIs, plugins ou configurações avançadas. Já plataformas *no-code* procuram abstrair ainda mais a complexidade técnica, buscando permitir que usuários com pouca ou nenhuma experiência em programação construam aplicações por meio de interfaces visuais e fluxos configuráveis.

Essa distinção, contudo, não deve ser interpretada de forma absoluta. Na prática, existe um espectro entre ferramentas com maior ou menor exigência técnica. Algumas plataformas classificadas como *no-code* podem exigir compreensão de lógica, dados, eventos e integração com APIs quando a aplicação ultrapassa funcionalidades simples. Da mesma forma, plataformas classificadas como *low-code* podem oferecer recursos automáticos que reduzem significativamente o esforço inicial de desenvolvimento. Por isso, a classificação de uma ferramenta como *low-code* ou *no-code* não garante, por si só, menor complexidade, maior produtividade ou menor necessidade de conhecimento técnico.

Martins, Branco e Mamede (MARTINS; BRANCO; MAMEDE, 2023) descrevem o desenvolvimento *low-code* como uma abordagem que reduz a codificação manual por meio de interfaces gráficas, elementos pré-construídos, objetos de dados, componentes de negócio e configuração de lógica. Os autores também destacam que essas plataformas ganharam relevância em razão da demanda por maior agilidade no desenvolvimento e da escassez de profissionais especializados. Essa perspectiva ajuda a compreender por que tais ferramentas têm sido apresentadas como alternativas para acelerar a entrega de soluções digitais.

De forma semelhante, Bruhin et al. (BRUHIN et al., 2024) caracterizam plata-

[1]Empresa de pesquisa e consultoria focada em tecnologia, negócios e experiência do cliente.

formas *low-code* como ambientes que utilizam abstração, automação, notações visuais e agilidade para apoiar o desenvolvimento de aplicações. Segundo os autores, essas plataformas reduzem a necessidade de programação manual e podem permitir que profissionais com menor conhecimento técnico participem do desenvolvimento de soluções. Ao mesmo tempo, essa redução de barreiras não elimina a necessidade de compreender a estrutura do sistema, seus dados, suas regras e suas integrações.

Entre os benefícios associados a plataformas *low-code* e *no-code*, destacam-se a aceleração da construção de aplicações, a reutilização de componentes, a redução de tarefas repetitivas, a aproximação entre equipes técnicas e áreas de negócio e a possibilidade de desenvolvimento por usuários que não são programadores profissionais. Em contextos organizacionais, essas ferramentas podem apoiar iniciativas de transformação digital, principalmente quando há demanda por sistemas internos, automações administrativas, painéis, formulários, fluxos de aprovação e aplicações de baixa ou média complexidade (LUO et al., 2021; FRANK; MAIER; BOCK, 2021).

No entanto, a literatura também aponta limitações relevantes. Entre elas estão a dependência da plataforma fornecedora (*vendor lock-in*), limitações de customização, dificuldades de integração, riscos de segurança, problemas de governança, restrições de escalabilidade, documentação insuficiente e necessidade de conhecimento técnico para lidar com cenários mais complexos (LUO et al., 2021; FRANK; MAIER; BOCK, 2021; MARTINS; BRANCO; MAMEDE, 2023). Esses desafios indicam que o ganho de produtividade prometido pelas plataformas não é automático, pois depende do tipo de aplicação, da maturidade da ferramenta, do nível de personalização exigido, das integrações necessárias e da experiência dos usuários envolvidos.

Outro ponto importante é que a adoção de plataformas *low-code* e *no-code* pode alterar a dinâmica do processo de desenvolvimento. Ao permitir que usuários de negócio participem mais ativamente da criação de soluções, essas ferramentas podem reduzir a distância entre quem demanda o sistema e quem o implementa. Por outro lado, essa ampliação da participação exige mecanismos de controle, documentação, padrões de desenvolvimento e governança, especialmente para evitar a criação de sistemas difíceis de manter, inseguros ou desconectados da arquitetura tecnológica da organização.

No contexto deste trabalho, a análise de plataformas *low-code* e *no-code* é realizada por meio de uma PoC aplicada a um sistema de agendamento de consultas médicas. A escolha desse tipo de sistema permite observar funcionalidades comuns em aplicações administrativas, como cadastro de registros, visualização de dados, relacionamentos entre entidades, automações e integração com serviços externos. Dessa forma, as plataformas são avaliadas não apenas pela promessa de rapidez, mas também por critérios relacionados à modelagem de dados, criação de interfaces, customização, integrações, automações, dependência da plataforma, documentação e necessidade de conhecimento técnico.

2.3 IA na Engenharia de Software

Nos últimos anos, ferramentas baseadas em inteligência artificial generativa passaram a ocupar espaço crescente no desenvolvimento de software. Em especial, os *Large Language Models* (LLMs), ou modelos de linguagem de grande porte, tornaram-se capazes de interpretar instruções em linguagem natural, gerar textos, explicar trechos de código, sugerir implementações, apoiar depuração, produzir documentação e auxiliar em decisões de projeto.

Zhao et al. (ZHAO et al., 2023) explicam que LLMs são modelos de linguagem em larga escala, geralmente baseados em arquiteturas *Transformer* e treinados sobre grandes volumes de dados textuais. Esses modelos apresentam capacidades de compreensão e geração de linguagem natural, além de habilidades emergentes que se tornam mais evidentes com o aumento de escala, como aprendizagem em contexto e execução de tarefas a partir de instruções. Essas características explicam por que ferramentas baseadas em LLMs passaram a ser utilizadas em diferentes atividades de desenvolvimento de software.

Na Engenharia de Software, o uso de IA generativa pode apoiar diversas etapas do ciclo de vida do desenvolvimento. Na fase de requisitos, pode auxiliar na escrita e refinamento de histórias de usuário, critérios de aceitação e descrições funcionais. Na etapa de projeto, pode sugerir estruturas de solução, padrões arquiteturais e alternativas de modelagem. Durante a implementação, pode gerar código, completar funções, propor refatorações e explicar erros. Nos testes, pode auxiliar na criação de casos de teste, dados de entrada e cenários de validação. Também pode contribuir para documentação, revisão

de código e manutenção (PACHECO et al., 2025; SAUVOLA et al., 2024).

Pacheco et al. (PACHECO et al., 2025) mostram que o Processamento de Linguagem Natural tem sido aplicado em diferentes artefatos e tarefas da Engenharia de Software, como requisitos, documentação, código-fonte e testes. Essa relação ocorre porque muitos artefatos do desenvolvimento de software são produzidos ou descritos em linguagem natural, o que permite o uso de técnicas de PLN para apoiar atividades como análise, classificação, extração de informações e geração de artefatos.

No contexto mais recente, os Large Language Models (LLMs) ampliam essa discussão ao permitir interações mais flexíveis baseadas em linguagem natural. Os autores também destacam o crescimento do uso de LLMs em tarefas como elicitação de requisitos, geração de código, correção de defeitos e testes de software (PACHECO et al., 2025). Dessa forma, os modelos de linguagem podem ser compreendidos como uma evolução relevante dentro do uso de PLN na Engenharia de Software, atuando não apenas como geradores de texto, mas também como ferramentas de apoio a atividades técnicas do desenvolvimento.

Sauvola et al. (SAUVOLA et al., 2024) argumentam que a IA generativa representa uma mudança relevante para o desenvolvimento de software, pois pode apoiar tarefas criativas e também substituir ou automatizar atividades repetitivas e manuais. Os autores destacam, entretanto, que esse avanço traz oportunidades e riscos, exigindo compreensão de suas limitações, diretrizes de uso e reflexão sobre impactos técnicos, organizacionais e éticos.

Um aspecto central no uso de LLMs é a formulação dos comandos fornecidos ao modelo. White et al. (WHITE et al., 2023) definem *prompts* como instruções fornecidas a um LLM para estabelecer regras, automatizar processos e direcionar as características da saída gerada. Os autores argumentam que *prompts* podem ser compreendidos como uma forma de programação, pois configuram o comportamento do modelo e orientam a interação com ele. Assim, a qualidade da resposta gerada por uma ferramenta de IA depende não apenas do modelo utilizado, mas também da clareza, completude e estrutura da instrução fornecida.

Apesar desse potencial, ferramentas de IA generativa não eliminam a necessidade

de conhecimento técnico. O usuário ainda precisa formular bons *prompts*, como indica White et al. (WHITE et al., 2023), avaliar a qualidade das respostas, validar o código produzido, compreender riscos de segurança, corrigir erros e garantir que a solução atenda aos requisitos do sistema. Além disso, modelos de IA podem gerar respostas incorretas, incompletas ou aparentemente plausíveis, exigindo revisão crítica e validação humana. Dessa forma, a IA generativa pode reduzir esforço em algumas tarefas, mas também desloca parte da responsabilidade para a verificação, curadoria e integração das soluções geradas ao contexto real do projeto.

A relação entre IA generativa e plataformas *low-code/no-code* é particularmente relevante para este trabalho. Por um lado, essas plataformas passaram a incorporar recursos de IA para gerar telas, sugerir estruturas de dados, automatizar fluxos ou apoiar configurações. Por outro lado, ferramentas externas de IA generativa também passaram a permitir a criação de protótipos e aplicações a partir de instruções em linguagem natural, disputando parte da proposta de valor tradicionalmente associada às plataformas *low-code* e *no-code*: acelerar o desenvolvimento e reduzir a necessidade de codificação manual.

Bruhin et al. (BRUHIN et al., 2024) investigam especificamente o impacto da IA generativa em plataformas *low-code*, destacando benefícios potenciais como aumento de eficiência e redução de erros, mas também enfatizando a importância da supervisão humana e da colaboração entre perfis técnicos e não técnicos. Essa perspectiva indica que a IA não substitui automaticamente as plataformas visuais, mas pode modificar a forma como elas são utilizadas e avaliadas.

Martins, Branco e Mamede (MARTINS; BRANCO; MAMEDE, 2023) também discutem a combinação entre ChatGPT e desenvolvimento *low-code*, propondo um modelo no qual o LLM atua como apoio ao processo de construção de aplicações. O estudo reforça a tendência de integração entre interfaces configuráveis e interação em linguagem natural, aproximando o desenvolvimento *low-code* de abordagens *no-code* baseadas em *prompts*.

Além disso, Monteiro et al. (MONTEIRO et al., 2025) apresentam o NoCodeGPT, uma interface *no-code* voltada à construção de pequenas aplicações web com modelos de linguagem. O estudo mostra que a interface padrão de chat não é necessariamente adequada para usuários com pouca experiência em desenvolvimento web, espe-

cialmente quando é necessário organizar arquivos, manter arquitetura, gerenciar versões e corrigir saídas incorretas. A proposta do NoCodeGPT busca abstrair essas decisões técnicas, permitindo que o usuário concentre seus prompts em requisitos funcionais.

Esses estudos mostram que a relação entre IA generativa e plataformas *low-code/no-code* ainda está em consolidação. A existência de IA generativa não significa que ferramentas *low-code* e *no-code* tenham perdido sua utilidade, pois elas ainda oferecem ambientes integrados, componentes prontos, infraestrutura gerenciada, conectores e mecanismos visuais de configuração. No entanto, o avanço da IA modifica os parâmetros de comparação. Se antes a principal alternativa era o desenvolvimento manual tradicional, agora parte da comparação envolve também ferramentas capazes de gerar código, interfaces e protótipos rapidamente a partir de descrições textuais.

Entre as ferramentas recentes baseadas em IA generativa voltadas ao desenvolvimento de aplicações, destaca-se a *Lovable*, uma plataforma que permite construir aplicações, sites e produtos digitais a partir de instruções em linguagem natural. Segundo sua documentação institucional, a ferramenta tem como proposta apoiar a criação de aplicações por meio de uma plataforma baseada em IA, sem exigir conhecimento profundo de programação (LOVABLE, 2026). Diferentemente de plataformas *low-code* e *no-code* tradicionais, que geralmente estruturam o desenvolvimento em torno de componentes visuais, fluxos configuráveis e modelagem manual de dados, ferramentas como a *Lovable* partem de uma interação baseada em *prompts*, na qual o usuário descreve a aplicação desejada e a ferramenta gera uma versão inicial do sistema que pode ser incrementada via *prompts* novamente. Dessa forma, a *Lovable* representa uma abordagem relevante para contextualizar o cenário atual de desenvolvimento apoiado por IA generativa, especialmente em relação à velocidade de prototipação e à geração inicial de interfaces.

Neste trabalho, a IA generativa é considerada de duas formas. Primeiro, como recurso interno das plataformas avaliadas, quando disponível, como ocorreu na geração inicial da aplicação na *Bubble* e no apoio à estruturação de dados na *Budibase*. Segundo, como elemento complementar de comparação, por meio da geração de uma versão alternativa da PoC em uma ferramenta de IA generativa. Essa comparação não busca esgotar a discussão sobre IA no desenvolvimento de software, mas contextualizar os resultados

obtidos e indicar como a proposta de valor das plataformas *low-code* e *no-code* pode ser impactada por esse novo cenário.

3 Trabalhos relacionados

Neste capítulo, da Seção 3.1 até a Seção 3.5 serão apresentados cinco (05) trabalhos relacionados que foram selecionados por sua relevância ao tema desta pesquisa. A análise desses estudos tem como propósito evidenciar diferentes perspectivas e contribuições já existentes na literatura, oferecendo um panorama que auxilia na compreensão do assunto investigado.

A exposição desses estudos busca, portanto, oferecer uma visão sintética das contribuições encontradas na literatura, descrevendo o objetivo de cada pesquisa, a forma como foi conduzida e as principais conclusões alcançadas. Dessa maneira, pretende-se contextualizar o tema em estudo e, ao final, na Seção 3.6 destacar como esses estudos se relacionam com a proposta deste trabalho.

3.1 Uma análise conceitual

O estudo denominado "Uma análise das principais plataformas low-code do mercado" (FERREIRA, 2024) está relacionado ao tema da presente pesquisa, visto que possui como objetivo principal investigar três das principais plataformas low-code disponíveis atualmente. Com o intuito de oferecer uma visão geral sobre elas, apresenta a perspectiva do autor quanto às vantagens e desvantagens de cada ferramenta, servindo como base de entendimento para profissionais e pesquisadores interessados nesse tipo de tecnologia.

A metodologia adotada pelo autor consistiu em uma pesquisa de caráter bibliográfico, na qual foram consultadas fontes teóricas e documentações técnicas das plataformas escolhidas. O trabalho foca especificamente em *Mendix*, *Outsystems* e *Power Apps*, analisando suas funcionalidades, pontos fortes e limitações. Por meio dessa abordagem, buscou-se fornecer um panorama comparativo que auxiliasse na compreensão de como essas soluções se posicionam no mercado.

Como resultado, o estudo concluiu que as plataformas avaliadas compartilham a característica central de acelerar o processo de desenvolvimento, mas cada uma delas apre-

senta diferentes níveis de complexidade, custos e públicos-alvo. Além disso, foi destacado que, apesar dos benefícios em termos de produtividade, ainda existem barreiras como a dependência do fornecedor e limitações na personalização de sistemas mais robustos.

A pesquisa apresenta limitações por não se aprofundar no estudo prático das plataformas, já que se baseia apenas em literatura, o que impede de entrar em detalhes sobre a real implementação da tecnologia.

3.2 Estudo comparativo entre dez sistemas

O relatório de pesquisa intitulado *Low code platforms: Promises, concepts and prospects. A comparative study of ten systems*, de (FRANK; MAIER; BOCK, 2021), amplia a discussão sobre plataformas *low-code* ao analisar dez sistemas distintos. Os autores abordam o tema sob uma perspectiva conceitual e comparativa, partindo do pressuposto de que ainda não há uma definição unificada para o termo *low-code*. Essa ausência de unificação decorre, segundo os autores, da diversidade de produtos comercializados sob esse rótulo e do uso frequente de descrições marcadas por jargões de mercado, promessas de fornecedores e interesses comerciais. Assim, ferramentas classificadas como *low-code* podem apresentar propostas bastante distintas, variando quanto ao tipo de abstração oferecida, ao grau de redução de código, ao público-alvo, à ênfase em modelagem visual, automação de processos, integração ou desenvolvimento de aplicações de negócio. Diante disso, o estudo busca compreender o que caracteriza uma plataforma *low-code* e quais elementos permitem diferenciá-la de outras abordagens de desenvolvimento.

O método adotado por (FRANK; MAIER; BOCK, 2021) é guiado por dois instrumentos da própria pesquisa: (1) uma *estrutura conceitual*, utilizada para descrever e comparar características comuns e específicas das plataformas analisadas, e (2) um *modelo de processo da investigação*, que organiza a sequência de etapas seguidas pelos autores durante o estudo. Nesse caso, o modelo de processo não se refere aos processos internos das plataformas avaliadas, mas ao procedimento metodológico utilizado pelos pesquisadores, que inclui etapas como estudo piloto, seleção das plataformas, preparação do ambiente de teste, análise das plataformas, conceitualização/caracterização e reflexão. Essa combinação contribui para tornar a comparação mais consistente e reduz o risco de que a

análise fique restrita a descrições subjetivas, comerciais ou meramente mercadológicas das ferramentas.

As conclusões do estudo indicam que as plataformas *low-code* não apresentam inovações radicais em termos de componentes, mas sim a síntese de elementos já conhecidos em ambientes que reduzem o esforço em tarefas rotineiras de desenvolvimento. Os autores destacam que os ganhos de produtividade são condicionais, manifestando-se de forma mais clara apenas quando os projetos se adequam ao esqueleto pré-definido das plataformas e quando existem condições técnicas e econômicas favoráveis. Além disso, apontam caminhos relevantes para futuras pesquisas em ciência da informação, com foco na retomada da modelagem conceitual, na avaliação crítica da própria terminologia "*low-code*" e na investigação das condições em que esse tipo de abordagem pode gerar resultados mais efetivos.

3.3 Aplicação em contexto de saúde pública

O trabalho "Farmácia cidadã – desenvolvimento de um sistema de gestão de distribuição de medicamentos utilizando ferramentas low-code"de (ALENCAR, 2023) apresenta como objetivo geral demonstrar a aplicação prática das abordagens *low-code* e *no-code* no desenvolvimento de um sistema para a farmácia estadual do Espírito Santo.

A metodologia adotada incluiu o levantamento dos requisitos que precisariam ser implementados posteriormente, a seleção de plataformas *low/no-code*, revisão de literatura sobre desenvolvimento de software e o conceito do *low-code*. A autora também elaborou a definição do design do sistema proposto, incluindo diagramas e demais especificações. Além da implementação e avaliação das ferramentas escolhidas (*Bubble*, *App Gyver* e *Budibase*). Vale ressaltar que o trabalho não contemplou a implantação e testes práticos junto aos usuários finais. O estudo também nos apresenta o funcionamento interno dos sistemas e a definição de arquitetura das plataformas selecionadas.

A conclusão enfatiza os benefícios potenciais da abordagem *low-code* no contexto público, sobretudo em termos de agilidade de desenvolvimento e melhoria no atendimento ao cidadão. Entretanto, destaca-se a necessidade de etapas posteriores envolvendo a integração com os demais procedimentos e sistemas pré-existentes do Governo Estadual onde

o escopo da aplicação é definido, para que o sistema possa ser efetivamente operacionalizado.

3.4 Percepção empírica sobre produtividade

A dissertação "Avaliação de plataformas *low code* e *no code* como estratégia para aumento da produtividade no desenvolvimento de software"de (SILVA, 2024) tem como foco principal a questão da produtividade. A metodologia envolve duas fases: (1) uma revisão multivocal da literatura sobre os benefícios e desafios dessas plataformas; e (2) o desenvolvimento de um protótipo de sistema para coleta seletiva de resíduos com perfis diferenciados (coletor, produtor e reciclador) implementado nas plataformas *OutSystems* e *Bubble*.

O trabalho parte da premissa de que a crescente demanda por soluções de software tem intensificado a busca por estratégias capazes de melhorar a produtividade no desenvolvimento. Nesse contexto, o autor investiga plataformas *low-code* e *no-code* como alternativas que prometem reduzir a necessidade de codificação manual, acelerar a construção de aplicações e ampliar a participação de usuários com diferentes níveis de conhecimento técnico. A pesquisa, portanto, não se limita a descrever essas plataformas de forma conceitual, mas busca verificar se os benefícios frequentemente atribuídos a elas, como redução do tempo de desenvolvimento, abstração de código, reutilização de componentes e maior facilidade de prototipação, também podem ser observados em um cenário prático de desenvolvimento (SILVA, 2024).

Nos resultados, o estudo identifica diversas vantagens associadas ao uso das plataformas, como abstração de código, reutilização de componentes e simplificação da colaboração entre integrantes da equipe. Por outro lado, são apontados desafios como dependência de plataforma, exigência de conhecimento técnico, limitações funcionais e preocupações com a segurança de dados.

3.5 A perspectiva dos profissionais

O artigo "Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective"de (LUO et al., 2021), traz uma contribuição singular ao abordar diretamente a visão de profissionais que utilizam plataformas low-code em seu cotidiano, afastando-se de comparações técnicas entre ferramentas específicas.

Para tanto, os autores coletaram dados em comunidades online amplamente utilizadas por desenvolvedores, como *Stack Overflow* e *Reddit*. A partir de um processo sistemático de filtragem e seleção, foram analisadas publicações que discutiam efetivamente o tema, as quais serviram de base para responder a nove questões de pesquisa propostas no estudo. Essas questões abrangem desde a compreensão conceitual do desenvolvimento low-code até os benefícios, limitações, plataformas populares, tecnologias de apoio e tipos de aplicações construídas, permitindo compor um panorama abrangente sobre como a prática é percebida por engenheiros de software.

Esse enfoque imprime uma perspectiva concreta, complementando pesquisas de caráter mais descritivo ou experimental com a voz de usuários reais das plataformas. Essa proximidade com a prática torna o trabalho particularmente relevante, ao destacar tanto os ganhos relatados (como maior rapidez de entrega e redução de custos) quanto os desafios enfrentados (como limitações técnicas, dificuldades de integração e preocupações de manutenção).

O estudo ainda dialoga com previsões de mercado apresentadas por consultorias como o *Gartner*[2], que apontam para o crescimento acelerado das plataformas low-code. Sua principal contribuição está em fornecer uma visão realista ancorada na experiência de profissionais, oferecendo insumos práticos que complementam abordagens conceituais e enriquecem a compreensão dos impactos, potenciais e limitações dessas tecnologias no desenvolvimento de software.

[2]Empresa global de pesquisa e consultoria focada em tecnologia https://www.gartner.com.br

3.6 Síntese dos estudos relacionados

A análise conjunta dos trabalhos referenciados anteriormente oferece tanto fundamentos conceituais quanto evidências práticas que se conectam diretamente ao tema desta pesquisa.

O estudo de (FERREIRA, 2024) contribui com um panorama inicial de três das principais plataformas do mercado, destacando prós e contras de cada uma. Embora de caráter essencialmente bibliográfico, sua contribuição é relevante por sistematizar informações que ajudam a compreender o posicionamento dessas ferramentas no cenário atual. Para o presente TCC, esse estudo funciona como ponto de partida, permitindo confrontar a teoria com resultados práticos por meio da implementação de uma Prova de Conceito (PoC).

O relatório de (FRANK; MAIER; BOCK, 2021) expande a discussão ao propor uma abordagem conceitual e comparativa, baseada em um quadro conceitual e em um modelo de processo que asseguram consistência metodológica na análise de dez plataformas distintas. Suas conclusões indicam que o low-code não representa uma ruptura tecnológica, mas sim a síntese de práticas já conhecidas em um novo arranjo que reduz esforços em tarefas rotineiras. Esse estudo fornece elementos teóricos importantes para este trabalho, permitindo compreender limites e potencialidades das plataformas além da visão mercadológica.

No caso de (ALENCAR, 2023), a pesquisa se encontra fortemente relacionada ao tema, vista a ênfase na aplicação prática da tecnologia em um sistema pensado para sanar uma dor real. O trabalho demonstra como a abordagem pode ser empregada em um contexto socialmente relevante, evidenciando benefícios e desafios operacionais ao fazê-lo. Para este TCC, esse estudo reforça a importância de entender o processo do desenvolvimento de software, nos guiando pelas etapas de construção de um estudo de caso e também o impacto da tecnologia em questão no cenário real.

A dissertação de (SILVA, 2024), por sua vez, aborda a avaliação de diferentes plataformas *low-code* e *no-code* sob a perspectiva da produtividade, investigando de que maneira essas ferramentas podem contribuir para acelerar entregas sem comprometer a qualidade. O trabalho traz reflexões sobre métricas e critérios de avaliação, o que auxilia

na construção de uma visão crítica acerca do real potencial dessas tecnologias, sendo, portanto, uma contribuição significativa para estudos que buscam equilibrar ganhos de produtividade com riscos inerentes à adoção.

Por fim, o artigo de (LUO et al., 2021) contribui para esta pesquisa ao sistematizar características e desafios associados ao uso de plataformas *low-code* a partir de discussões em comunidades online. Embora o presente trabalho não tenha como objetivo analisar diretamente a percepção de engenheiros de software ou de usuários dessas comunidades, o estudo é relevante por evidenciar problemas recorrentes no uso prático dessas ferramentas, como limitações de customização, dificuldades de integração, dependência da plataforma e necessidade de conhecimento técnico em determinados cenários. Dessa forma, ele complementa os demais trabalhos relacionados ao oferecer subsídios para compreender limitações que também podem ser observadas em avaliações práticas de plataformas *low-code/no-code*.

4 Metodologia

Este capítulo apresenta detalhadamente o percurso metodológico adotado para o desenvolvimento desta pesquisa, previamente introduzido na Seção 1.5. Sua finalidade é esclarecer a abordagem científica escolhida, bem como os procedimentos empregados para responder às questões de pesquisa estabelecidas.

As seções a seguir descrevem a classificação da pesquisa e as quatro etapas principais que compõem a metodologia adotada: (1) a revisão e seleção da literatura de base, (2) a sistematização dos critérios analíticos e da escala de complexidade, (3) a condução do estudo prático em plataformas *low-code* e *no-code*, e (4) a avaliação comparativa dos resultados obtidos.

4.1 Metodologia Científica

4.1.1 Classificação da Pesquisa

A presente pesquisa caracteriza-se como exploratória, aplicada, bibliográfica e de abordagem predominantemente qualitativa, com apoio de dados quantitativos descritivos. Essa classificação considera os objetivos do estudo, sua natureza, os procedimentos técnicos adotados e a forma de abordagem do problema.

Quanto aos objetivos, a pesquisa é classificada como exploratória, pois busca ampliar a compreensão sobre o uso de plataformas *low-code* e *no-code* no desenvolvimento de software, tema ainda em consolidação na literatura científica, especialmente quando analisado em conjunto com os impactos recentes das ferramentas baseadas em inteligência artificial generativa. Segundo Lakatos e Marconi (LAKATOS; MARCONI, 2017), investigações exploratórias buscam aumentar a familiaridade com um fenômeno ainda pouco estruturado, fornecendo subsídios para compreendê-lo de maneira mais clara.

Em relação à natureza, trata-se de uma pesquisa aplicada, uma vez que o estudo não se limita à discussão teórica sobre o tema, mas propõe e utiliza um sistema de

critérios avaliativos para analisar, na prática, plataformas *low-code* e *no-code* a partir da implementação de uma prova de conceito. Dessa forma, busca-se produzir conhecimentos com potencial de aplicação em avaliações futuras e no apoio à tomada de decisão sobre a adoção dessas ferramentas em contextos de desenvolvimento de software.

Quanto aos procedimentos técnicos, a pesquisa possui caráter bibliográfico, pois se fundamenta em artigos, estudos e materiais científicos já publicados sobre desenvolvimento *low-code*, *no-code*, produtividade, desafios de adoção e limitações dessas plataformas. Esse levantamento serviu de base para a construção do referencial teórico e para a definição dos critérios utilizados na avaliação prática.

Por fim, quanto à forma de abordagem do problema, a pesquisa é classificada como predominantemente qualitativa, com apoio de dados quantitativos descritivos. O aspecto qualitativo está presente na análise interpretativa das vantagens, limitações, desafios e implicações práticas observadas durante o uso das plataformas. Já o apoio quantitativo aparece na atribuição de notas aos critérios avaliativos, na comparação entre as plataformas e na apresentação dos resultados por meio de tabelas e gráficos. Embora o estudo utilize medidas numéricas, seu objetivo não é realizar inferências estatísticas, mas organizar e apoiar a análise comparativa das plataformas avaliadas.

Dessa forma, a classificação adotada sustenta as decisões metodológicas do trabalho, articulando a revisão da literatura com uma avaliação prática baseada em critérios previamente definidos.

4.1.2 Revisão e Seleção da Literatura de Base

A primeira etapa da metodologia consistiu na seleção e análise da literatura sobre plataformas *low-code* e *no-code*, com o objetivo de compreender seus benefícios, desafios, limitações e impactos no processo de desenvolvimento de software. Diferentemente de uma Revisão Sistemática da Literatura, adotou-se uma busca bibliográfica orientada por critérios previamente definidos, porém adaptada ao escopo e ao tempo disponíveis.

Busca exploratória inicial

A etapa inicial consistiu em uma busca exploratória ampla, com o objetivo de identificar como as plataformas *low-code* e *no-code* vêm sendo discutidas na literatura recente e quais temas recorrentes emergem dessas discussões. Essa busca preliminar também auxiliou na definição dos critérios de inclusão e exclusão aplicados posteriormente.

Foram elaboradas duas *strings* de busca, aplicadas no *Google Scholar* e no *Scite*, conforme apresentado a seguir:

1. `("no code platforms"OR "low code platforms"OR "low-code development")`
   `AND ("advantages"OR "disadvantages"OR "challenges")`
   `AND ("software development")`

2. `("no code"OR "low code") AND "desenvolvimento de software"`
   `AND ("produtividade"OR "vantagens"OR "desvantagens"`
   `OR "desafios"OR "adoção")`

A busca retornou 3.595 registros iniciais, o que demandou uma estratégia de filtragem progressiva focada nos 100 trabalhos mais visualmente aderentes ao tema.

Foram definidos critérios explícitos para orientar a seleção dos estudos.

Critérios de Inclusão:

- Estudos que discutem diretamente o uso de plataformas *low-code* e/ou *no-code* no desenvolvimento de software;
- Trabalhos que abordam vantagens, desvantagens, desafios e/ou adoção dessa tecnologia;
- Textos disponíveis integralmente e gratuitamente;
- Publicações em português ou inglês.

Critérios de Exclusão:

- Estudos que não tratam diretamente de *low-code*/*no-code*;
- Materiais não científicos (blogs, vídeos, marketing comercial);
- Trabalhos indisponíveis integralmente.

Processo de triagem

A filtragem dos estudos ocorreu em quatro etapas:

- **Leitura de títulos**: dos 3.595 resultados, foram selecionadas as 100 publicações com títulos mais aderentes ao tema e que foram escritos em Português ou Inglês.
- **Validação de acesso**: nessa etapa foram eliminados aqueles que não estavam disponíveis para leitura gratuitamente.
- **Leitura de resumos**: após a leitura dos resumos/abstracts, o conjunto foi reduzido para 13 publicações.
- **Leitura completa**: por fim, após a leitura completa, 01 artigo foi eliminado por fugir do tema da pesquisa. Resultando na inclusão final de 12 publicações.

Após esse processo, cinco desses estudos foram aprofundados e discutidos no capítulo de Trabalhos Relacionados (ver Capítulo 3), por sua relevância conceitual e metodológica.

4.2 Avaliação prática

4.2.1 Ferramentas selecionadas

Para fins desta pesquisa, *Budibase* e *Bubble* foram utilizadas como plataformas de referência para avaliação prática, representando abordagens distintas dentro do espectro *low-code* e *no-code*.

A seleção das plataformas foi realizada de forma intencional, considerando a aderência ao objetivo da prova de conceito, a viabilidade de experimentação no período disponível, a disponibilidade de recursos suficientes para implementação do sistema proposto e a possibilidade de observar diferenças entre uma abordagem *low-code* e uma abordagem *no-code*.

A *Bubble* foi selecionada como representante da abordagem *no-code*, por propor a criação de aplicações web por meio de recursos visuais, fluxos de trabalho e modelagem de dados com baixa exigência de codificação manual. A *Budibase*, por sua vez, foi selecionada

como representante da abordagem *low-code*, por combinar recursos visuais com maior proximidade de conceitos técnicos.

Durante o levantamento inicial, outras plataformas também foram consideradas, como *Mendix*, *Power Apps* e *OutSystems*. No entanto, optou-se por não incluí-las no escopo prático da pesquisa devido às limitações de tempo, ao esforço necessário para implementar a mesma prova de conceito em múltiplas ferramentas e à necessidade de manter a análise concentrada em dois casos representativos. Embora essas plataformas também sejam relevantes no mercado de desenvolvimento *low-code*, elas apresentam características que poderiam ampliar excessivamente o escopo do estudo. O *Mendix* e o *OutSystems*, por exemplo, estão associados a ecossistemas corporativos mais robustos, voltados a aplicações empresariais de maior escala. Já o *Power Apps* possui forte integração com o ecossistema Microsoft, o que poderia introduzir dependências específicas desse ambiente na avaliação.

Assim, a escolha de *Bubble* e *Budibase* não busca afirmar que essas plataformas representam todas as possibilidades existentes no mercado, mas utilizá-las como casos práticos e viáveis para observar os critérios definidos na Subseção 4.2.2 e associados às estratégias *low-code* e *no-code*. Essa delimitação permitiu conduzir uma análise compatível com os objetivos da pesquisa, mantendo o foco na comparação entre duas abordagens distintas de desenvolvimento visual de software.

Budibase

Budibase é uma plataforma voltada ao desenvolvimento de aplicações web e mobile por meio de componentes reutilizáveis e fluxos de automação configuráveis pelo usuário. Seu ambiente de construção permite criar interfaces visuais, definir modelos de dados e estabelecer regras de negócio com baixa necessidade de codificação manual. Segundo o seu site oficial, "*Budibase* is an open-source workflow platform that saves engineers time and energy building apps that integrate with any data source and speed-up any process" (BUDIBASE, 2025). No contexto deste trabalho, *Budibase* foi utilizada como exemplo de abordagem *low-code*, permitindo observar o processo de construção de funcionalidades com algum grau de flexibilidade técnica e personalização.

Bubble

Bubble é uma plataforma de desenvolvimento de aplicações web e mobile orientada ao uso de elementos visuais e fluxos lógicos configurados por meio de interface gráfica. A ferramenta integra recursos para construção de telas, gerenciamento de dados, autenticação e publicação de aplicações, priorizando a abstração do código. Conforme descrito pelo site da plataforma, "Bubble is an AI app platform that lets anyone build powerful web and mobile apps visually, without code" (BUBBLE, 2025). Neste estudo, *Bubble* foi utilizada como exemplo de abordagem *no-code*, permitindo observar como funcionalidades completas podem ser construídas com foco em automação e minimização de esforço técnico.

4.2.2 Critérios avaliativos

Com base nos objetivos definidos e nos temas recorrentes identificados na literatura consultada, foram estabelecidos critérios para orientar a avaliação das plataformas durante o desenvolvimento da prova de conceito. Esses critérios foram organizados em seis grupos: agilidade e evolução, desenvolvimento e flexibilidade, integrações, limitações e riscos, usabilidade, e documentação e suporte. A divisão em grupos teve como objetivo facilitar a comparação entre as plataformas e permitir que a análise não se restringisse apenas ao tempo de desenvolvimento, mas também considerasse fatores como manutenção, customização, dependência da plataforma e necessidade de conhecimento técnico.

Os critérios avaliativos foram definidos a partir de duas fontes principais. Em primeiro lugar, foram consideradas dimensões recorrentes nos trabalhos relacionados sobre plataformas *low-code* e *no-code*, como velocidade de desenvolvimento, modelagem de dados, criação de interfaces, integração com serviços externos, automação de fluxos, dependência da plataforma, necessidade de conhecimento técnico, documentação e suporte. Esses aspectos foram considerados por sua relevância para a avaliação prática de plataformas *low-code* e *no-code* no desenvolvimento de software. Em segundo lugar, outros critérios foram definidos a partir da experiência prática de implementação, com o objetivo de capturar aspectos observados durante o uso das plataformas, como facilidade de início, facilidade de edição pós-geração, facilidade de *debug* e qualidade da aplicação

gerada automaticamente. Dessa forma, a matriz avaliativa não corresponde à reprodução direta de critérios já existentes, mas a uma estrutura própria, construída a partir de aspectos recorrentes nos trabalhos relacionados e de dimensões observáveis no processo de desenvolvimento com plataformas *low-code* e *no-code*.

Os critérios avaliativos definidos são apresentados a seguir.

1. **Agilidade e evolução**: grupo relacionado à rapidez para iniciar, gerar e evoluir a aplicação após sua criação inicial.

    - **Velocidade de desenvolvimento**: avalia a rapidez para evoluir o sistema após o início da implementação, considerando a criação de telas, entidades, fluxos e funcionalidades necessárias à aplicação avaliada.
    - **Qualidade da aplicação gerada automaticamente**: avalia a qualidade estrutural e funcional do sistema gerado por recursos automáticos da plataforma, como IA, templates ou geração de telas a partir de dados.
    - **Facilidade de ajuste e evolução pós-geração**: avalia a facilidade de modificar, corrigir e evoluir o sistema após a geração inicial, considerando ajustes visuais, funcionais e estruturais. Esse critério não se limita à manutenção de um sistema já finalizado, pois também envolve adaptações necessárias durante a construção e refinamento da aplicação gerada.

2. **Desenvolvimento e flexibilidade**: grupo relacionado à criação da estrutura interna da aplicação, das telas e das possibilidades de adaptação.

    - **Modelagem de dados**: avalia a facilidade para criar, organizar e manter a estrutura de dados da aplicação, incluindo tabelas, campos, entidades e relacionamentos.
    - **Criação de interface (UI)**: avalia a facilidade de criação e ajuste das telas da aplicação, considerando organização visual, responsividade, formulários, listagens e componentes de interação.
    - **Flexibilidade para customizações**: avalia a capacidade da plataforma de permitir adaptações além do padrão inicialmente oferecido, incluindo alterações

de layout, regras específicas, plugins, scripts, extensões ou configurações avançadas.

3. **Integrações**: grupo relacionado à comunicação com serviços externos e à automação de processos.

    - **Integração com APIs externas**: avalia a facilidade para conectar a aplicação a serviços externos, configurar requisições, autenticação, parâmetros e troca de dados com outras ferramentas.
    - **Configuração de automações**: avalia a facilidade de entender, configurar e manter automações dentro da plataforma, como envio de mensagens, execução de fluxos a partir de eventos e acionamento de serviços externos.

4. **Limitações e riscos**: grupo relacionado a fatores que podem dificultar a manutenção, evolução, controle técnico ou adoção da aplicação.

    - **Dependência da plataforma (*vendor lock-in*)**: avalia o grau de dependência da plataforma para funcionamento, manutenção, publicação e continuidade do sistema, considerando limitações de portabilidade e dependência de infraestrutura proprietária.
    - **Necessidade de conhecimento técnico**: avalia o nível de conhecimento técnico necessário para desenvolver, integrar e manter a aplicação. Esse critério foi interpretado de acordo com a proposta declarada de cada plataforma: na *Bubble*, considerou-se a expectativa de menor exigência técnica associada a ferramentas *no-code*; na *Budibase*, considerou-se que plataformas *low-code* pressupõem maior participação técnica do usuário.
    - **Facilidade de debug**: avalia a facilidade de identificar, compreender e corrigir erros durante o desenvolvimento, considerando mensagens de erro, alertas, logs, mecanismos de inspeção e clareza das informações fornecidas pela plataforma.

5. **Usabilidade**: grupo relacionado à facilidade de uso da plataforma durante o início e a condução do desenvolvimento da PoC.

- **Facilidade de início**: avalia a facilidade para começar o desenvolvimento do sistema, incluindo configuração inicial, criação do primeiro protótipo e compreensão dos primeiros passos dentro da plataforma.
- **UX da plataforma**: avalia a experiência de uso da ferramenta durante o desenvolvimento, incluindo organização dos menus, previsibilidade dos fluxos, facilidade de localização dos recursos, clareza das configurações e consistência da interface.

6. **Documentação e suporte**: grupo relacionado aos materiais e canais disponíveis para apoiar o uso da plataforma durante o desenvolvimento.

   - **Documentação**: avalia a qualidade, atualidade, completude e utilidade da documentação oficial, tutoriais, exemplos e materiais de apoio disponíveis para auxiliar o desenvolvimento.
   - **Suporte**: avalia a disponibilidade e utilidade dos canais de suporte, comunidades, fóruns, central de ajuda ou outros meios de resolução de dúvidas e problemas encontrados durante o uso da plataforma.

Esses critérios serviram como base para a construção do quadro avaliativo aplicado às duas plataformas. A avaliação considerou não apenas a nota atribuída a cada critério, mas também as justificativas qualitativas registradas durante o desenvolvimento da PoC. Dessa forma, as notas não foram utilizadas como um ranking absoluto entre ferramentas, mas como um recurso de apoio para organizar e comparar as observações práticas.

A escolha desses critérios também permitiu verificar se a promessa de agilidade e simplicidade associada às plataformas *low-code* e *no-code* se confirmou no cenário analisado. Assim, a avaliação buscou observar não apenas se as plataformas permitiam construir a aplicação, mas também quanto esforço foi necessário para ajustar, integrar, corrigir e evoluir o sistema após sua criação inicial.

4.2.3 Construção da Escala de Avaliação

Para avaliar as plataformas, adotou-se uma escala Likert de cinco níveis, acompanhada da categoria adicional "Não se aplica" (N/A). As escalas Likert são amplamente utilizadas em pesquisas qualitativas por permitirem mensurar a intensidade com que um critério é atendido, mantendo a consistência na apreciação entre múltiplos itens (LIKERT, 1932; JOSHI et al., 2015).

Os níveis da escala foram definidos da seguinte forma:

- **N/A – Não se aplica**: o aspecto não se aplica ao contexto da PoC ou não pôde ser avaliado. Também foi utilizado quando não havia evidências suficientes para atribuição de nível.
- **1 – Muito insatisfatório**: o aspecto não é atendido ou apresenta limitações severas. Quando aplicável a critérios relacionados a custo, esforço, exigência ou impacto, indica nível mínimo de exigência ou impacto.
- **2 – Insatisfatório**: o aspecto é atendido parcialmente, mas apresenta limitações relevantes. Quando aplicável a critérios relacionados a custo, esforço, exigência ou impacto, indica baixo nível de exigência ou impacto.
- **3 – Neutro**: o aspecto é atendido de maneira mediana, sem destaque positivo ou negativo significativo. Quando aplicável a critérios relacionados a custo, esforço, exigência ou impacto, indica nível moderado de exigência ou impacto.
- **4 – Satisfatório**: o aspecto é atendido de forma consistente, com poucas limitações. Quando aplicável a critérios relacionados a custo, esforço, exigência ou impacto, indica alto nível de exigência ou impacto.
- **5 – Excelente**: o aspecto é atendido de forma excelente, sem limitações relevantes. Quando aplicável a critérios relacionados a custo, esforço, exigência ou impacto, indica nível muito alto de exigência ou impacto.

Além da pontuação, cada avaliação deverá conter uma justificativa qualitativa, contribuindo para maior transparência e confiabilidade dos resultados (MOHAJAN, 2017).

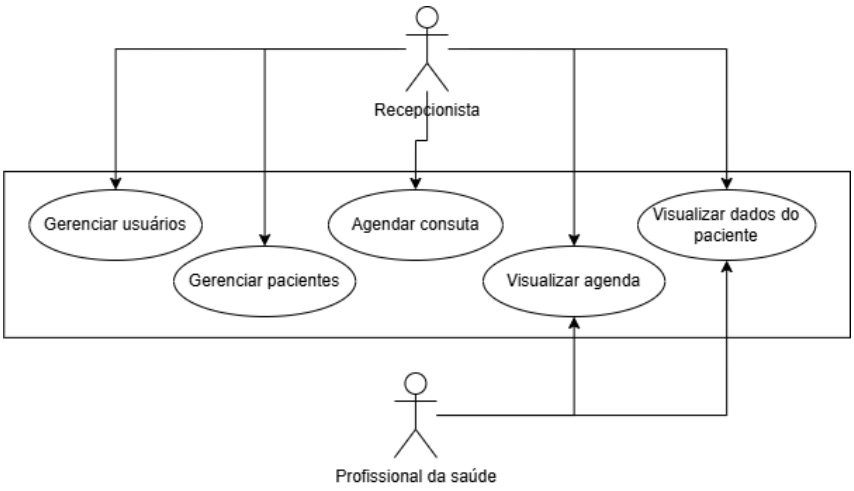

Figura 4.1: Diagrama de casos de uso da prova de conceito.

4.2.4 Prova de Conceito

A prova de conceito desenvolvida neste trabalho consistiu na implementação de um sistema de agendamento de consultas para uma clínica. O objetivo da PoC foi criar um cenário funcional comum às plataformas avaliadas, permitindo observar o processo de desenvolvimento.

O sistema considerado na PoC contemplou dois perfis principais de usuários: recepcionista e profissional da saúde. A recepcionista é responsável por atividades administrativas, como gerenciamento de pacientes, agendamento de consultas, visualização de agendas e gerenciamento de usuários. O profissional da saúde, por sua vez, pode visualizar sua agenda e consultar dados dos pacientes.

A Figura 4.1 apresenta o diagrama de casos de uso definido para a prova de conceito. Esse diagrama representa o escopo funcional utilizado como referência para a implementação nas plataformas avaliadas.

A partir desse escopo, foram consideradas entidades básicas como pacientes, profissionais, agendamentos e mensagens. Entre as funcionalidades previstas estavam o cadastro e a edição de registros, a visualização de informações em telas do tipo *Create, Read, Update and Delete* (CRUD), o registro de consultas com data, horário, paciente e profissional associado, além da possibilidade de acionar um serviço externo para envio de lembretes por *WhatsApp*.

A proposta não foi construir um sistema final pronto para produção, nem repre-

sentar integralmente a complexidade de um sistema real de gestão em saúde. A PoC foi concebida como um protótipo funcional e delimitado, com um conjunto de funcionalidades suficiente para observar o comportamento das plataformas *low-code* e *no-code* durante o desenvolvimento de uma aplicação. Assim, o objetivo não foi atender a todos os requisitos de um software completo, mas criar um cenário de teste que permitisse avaliar aspectos como esforço de desenvolvimento, facilidade de configuração, flexibilidade, integrações e limitações práticas.

Dessa forma, a PoC contemplou funcionalidades comuns em sistemas de agendamento, como cadastro, edição e visualização de registros, mas também incluiu elementos que exigem maior configuração das plataformas, como relacionamentos entre entidades, automações e integração com serviço externo. Esse recorte permitiu testar recursos relevantes das ferramentas sem ampliar excessivamente o escopo da aplicação, o que poderia deslocar o foco da pesquisa para a construção de um sistema mais complexo, e não para a avaliação das plataformas.

4.2.5 Procedimento de análise dos resultados

Como apresentado em 4.2.2, a avaliação foi conduzida com base nos critérios definidos para o estudo, organizados em seis grupos: agilidade e evolução, desenvolvimento e flexibilidade, integrações, limitações e riscos, usabilidade, e documentação e suporte. A interpretação das notas considerou também a natureza de cada plataforma. Assim, a *Bubble* foi avaliada considerando a expectativa de maior simplicidade associada a ferramentas *no-code*, enquanto a *Budibase* foi avaliada considerando que plataformas *low-code* pressupõem maior conhecimento técnico por parte do usuário.

A avaliação das plataformas foi realizada com base nos critérios definidos na Subseção 4.2.2 e na escala apresentada na Subseção 4.2.3. As notas atribuídas à Bubble e à Budibase são apresentadas de forma resumida no capítulo de resultados (Capítulo 5), acompanhadas de uma análise dos principais resultados observados durante a implementação da PoC.

Os quadros completos de avaliação, contendo a nota e a justificativa detalhada atribuída a cada critério em cada plataforma, também são apresentados no Capítulo 5

como Tabela 5.1 e Tabela 5.2.

5 Resultados

Neste capítulo, são apresentados os resultados desta pesquisa, que incluem a implementação da prova de conceito (PoC) de um sistema de agendamento de consultas nas plataformas *Bubble* e *Budibase*, comparação com uma aplicação gerada por IA generativa através da ferramenta Lovable, e a discussão dos resultados. Inicialmente, são discutidas as experiências individuais em cada plataforma, considerando os critérios avaliativos definidos na metodologia. Em seguida, é apresentada uma comparação complementar com uma aplicação gerada por IA generativa. Por fim, os resultados são consolidados em uma visão geral das notas atribuídas e em uma síntese interpretativa dos principais achados.

5.1 Avaliação da plataforma *Bubble*

O desenvolvimento da PoC na *Bubble* foi iniciado a partir do recurso de geração de aplicação por inteligência artificial da própria plataforma. Essa decisão decorreu do próprio fluxo de uso apresentado pela ferramenta durante o início da criação da aplicação, uma vez que o *onboarding* da plataforma direciona o usuário para a utilização desse recurso como ponto de partida. Dessa forma, optou-se por seguir o caminho sugerido pela própria *Bubble*, com o objetivo de observar a experiência prática oferecida a um novo usuário e avaliar se a geração inicial por IA contribuiria efetivamente para acelerar o desenvolvimento da prova de conceito. Essa escolha também se mostrou coerente com os critérios avaliativos definidos neste trabalho, especialmente aqueles relacionados à facilidade de início, qualidade da aplicação gerada automaticamente e facilidade de edição pós-geração. Assim, o uso inicial da IA não foi tratado como uma comparação independente com ferramentas externas de inteligência artificial generativa, mas como parte da experiência nativa de uso da plataforma *Bubble*.

A Tabela 5.1 apresenta o quadro completo de avaliação da plataforma *Bubble*, contendo o grupo avaliativo, o critério analisado, a nota atribuída e a justificativa correspondente. A descrição conceitual dos critérios não é repetida nesta seção, pois já foi

apresentada na Subseção 4.2.2. Na sequência, esses resultados são discutidos com apoio das evidências observadas durante a implementação da PoC.

Tabela 5.1: Quadro completo de avaliação da plataforma *Bubble*.

Grupo	**Critério**	**Nota**	**Justificativa**
Usabilidade	Facilidade de início	3 – Neutro	A IA gerou rapidamente uma base visual, mas a aplicação não saiu realmente funcional e exigiu ajustes antes de avançar. Para uma plataforma *no-code*, o resultado inicial ficou abaixo do esperado.
Agilidade e evolução	Velocidade de desenvolvimento	2 – Insatisfatório	Apesar do início rápido, a evolução exigiu muito trabalho manual, ajustes de layout, correções e configuração de lógica.
Agilidade e evolução	Qualidade da aplicação gerada automaticamente	1 – Muito insatisfatório	A base gerada era visualmente agradável, mas apresentou erros, funcionalidades incompletas e necessidade relevante de reorganização.
Agilidade e evolução	Facilidade de ajuste e evolução pós-geração	2 – Insatisfatório	A edição foi possível, mas demandou esforço manual frequente e organização prévia da estrutura da aplicação. O critério considera ajustes, correções e evolução após a geração inicial, não apenas manutenção de um sistema finalizado.
Desenvolvimento e flexibilidade	Modelagem de dados	3 – Neutro	A estrutura atendeu à PoC, mas não apresentou ganho expressivo em relação ao esforço exigido.
Desenvolvimento e flexibilidade	Criação de interface (UI)	2 – Insatisfatório	Houve necessidade frequente de ajustes manuais de posição, responsividade, modais e elementos visuais.
Desenvolvimento e flexibilidade	Flexibilidade para customizações	3 – Neutro	Há plugins e possibilidades de customização, mas isso exige configuração manual, conhecimento técnico e, em alguns casos, assinatura paga.
Integrações	Integração com APIs externas	2 – Insatisfatório	A integração foi possível via API Connector, mas exigiu conhecimento técnico e não se mostrou simples para uma proposta *no-code*.
Integrações	Configuração de automações	2 – Insatisfatório	A configuração depende bastante da organização dos elementos da interface e exige entendimento de lógica, APIs e eventos.

Grupo	**Critério**	**Nota**	**Justificativa**
Limitações e riscos	Dependência da plataforma (*vendor lock-in*)	1 – Muito insatisfatório	Há forte dependência da infraestrutura da plataforma, dos plugins e do ambiente proprietário.
Limitações e riscos	Necessidade de conhecimento técnico	1 – Muito insatisfatório	Por ser *no-code*, esperava-se menor exigência técnica. No entanto, integrações e automações exigiram conhecimento de APIs, lógica e estruturação da aplicação.
Limitações e riscos	Facilidade de debug	2 – Insatisfatório	A aba de erros e alertas existe, mas o volume de mensagens dificulta localizar problemas específicos.
Usabilidade	UX da plataforma	3 – Neutro	A interface é utilizável, mas a curva de aprendizado e o esforço manual são maiores do que o esperado para uma ferramenta *no-code*.
Documentação e suporte	Documentação e suporte	3 – Neutro	A documentação auxilia, mas não foi suficiente para eliminar dificuldades práticas encontradas durante integrações e ajustes.

A partir desse fluxo inicial, foi submetido um *prompt* descrevendo o objetivo do sistema, os usuários, os requisitos de autenticação, as entidades principais, os campos necessários, as funcionalidades de *CRUD* e a possibilidade de integração com serviço externo para envio de lembretes por *WhatsApp*. A ferramenta gerou uma base visual inicial para a aplicação, o que contribuiu para reduzir o esforço de configuração inicial. Entretanto, a avaliação desse primeiro resultado foi classificada como neutra, no sentido definido na escala da Subseção 4.2.3, isto é, como um resultado mediano, sem destaque positivo ou negativo significativo.

A Figura 5.1 ilustra uma das telas iniciais geradas pela inteligência artificial da *Bubble* para o cadastro de agendamentos. A figura evidencia problemas associados aos critérios de qualidade da aplicação gerada automaticamente e criação de interface, pois a tela apresenta inconsistências visuais e funcionais mesmo após ajustes manuais. Entre eles, destacam-se a presença de textos em idiomas diferentes na mesma interface, como rótulos em inglês combinados com informações configuradas em português, desalinhamentos e proporções inadequadas em componentes visuais, além de campos de entrada que não estavam corretamente associados aos dados esperados. Também foram identificados botões e ações sem funcionamento completo, exigindo reconfiguração manual dos fluxos

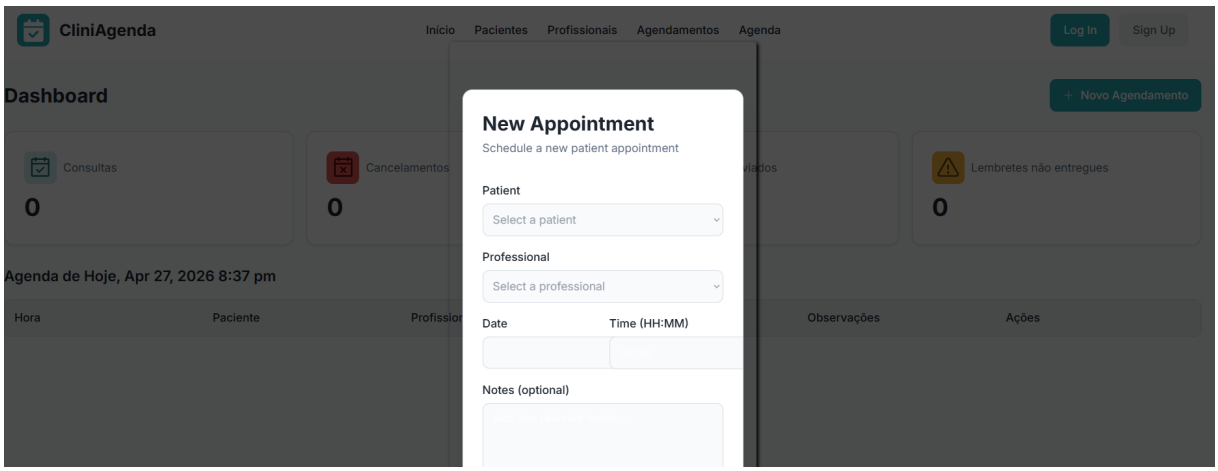

Figura 5.1: Tela de agendamento gerada pela inteligência artificial da *Bubble*, com inconsistências visuais, mistura de idiomas e necessidade de ajustes manuais em campos e ações.

para que as operações de cadastro e navegação pudessem ser executadas corretamente.

Apesar da proposta *no-code* da plataforma, a experiência prática demonstrou que a evolução da aplicação exigiu uma quantidade significativa de trabalho manual. Foram necessários ajustes no formato do campo de telefone, correções em limites de caracteres, alterações de largura em modais, tradução de textos gerados em inglês, adequações de espaçamento e alinhamento em diferentes elementos da interface, além da revisão de botões e campos que não executavam corretamente as ações esperadas. Esses ajustes envolveram tanto aspectos visuais, relacionados à apresentação da interface, quanto aspectos funcionais, relacionados à associação entre campos, dados e workflows. Por esse motivo, a velocidade de desenvolvimento e a facilidade de edição pós-geração foram avaliadas como insatisfatórias. A geração inicial acelerou o começo da implementação, mas a produtividade reduziu à medida que o sistema precisou ser refinado.

A qualidade da aplicação gerada automaticamente também foi avaliada como muito insatisfatória. Embora a base inicial fosse visualmente tolerável, ela apresentava erros e funcionalidades incompletas. A interface gerada não estava totalmente conectada à estrutura de dados necessária para a PoC, alguns campos não executavam corretamente o cadastro ou a edição de informações, e determinadas ações exigiam reconfiguração manual dos workflows. Além disso, a mistura de idiomas e a necessidade de corrigir componentes visuais indicaram que a geração automática produziu uma base inicial útil apenas como ponto de partida, mas insuficiente para ser considerada uma aplicação funcional. Também

foi observada a necessidade de reorganização da estrutura interna do projeto. Embora a Bubble não utilize uma arquitetura baseada em código-fonte tradicional, a aplicação gerada apresentou baixa clareza na organização dos elementos, páginas, componentes e fluxos de ação. Do ponto de vista de uma desenvolvedora acostumada com projetos de software organizados em estruturas de código mais definidas, a navegação interna do projeto se mostrou pouco intuitiva, dificultando a localização dos elementos que precisavam ser gerenciados ou alterados. Esse aspecto contribuiu para aumentar o esforço de manutenção e edição após a geração inicial.

Esse resultado é importante porque a *Bubble*, por ser uma plataforma *no-code*, tende a criar a expectativa de que usuários com menor conhecimento técnico consigam desenvolver aplicações com baixa complexidade operacional. No entanto, a PoC indicou que essa expectativa não se confirmou integralmente no cenário analisado.

No grupo de desenvolvimento e flexibilidade, a modelagem de dados foi avaliada como neutra. A estrutura criada atendeu à PoC, mas não apresentou ganho expressivo em relação ao esforço exigido. A criação da interface recebeu uma avaliação insatisfatória, pois houve necessidade frequente de ajustes manuais de posição, responsividade, modais e elementos visuais. A flexibilidade para customizações foi considerada neutra, uma vez que a plataforma oferece plugins e alternativas de customização, mas essas possibilidades dependem de configuração manual, conhecimento técnico e, em alguns casos, assinatura paga.

A etapa de integração evidenciou uma das principais limitações da experiência na *Bubble*. Inicialmente, foi testada a configuração de envio de mensagens por *WhatsApp* com plugin, mas a alternativa apresentou dificuldades, incluindo erro relacionado ao domínio do plugin e necessidade de contratação de plano pago. Posteriormente, foi testada uma alternativa gratuita com *API Collections*, funcionalidade nativa da plataforma, e *Twilio*, ferramenta externa com *API*, o que permitiu avançar na integração.

A Figura 5.2 evidencia que a integração não ocorreu apenas por seleção visual de componentes, pois exigiu a configuração manual de uma requisição HTTP, incluindo definição de método, URL, parâmetros, autenticação e corpo da chamada. Esse processo impactou o critério de integração com APIs externas, uma vez que demandou conhe-

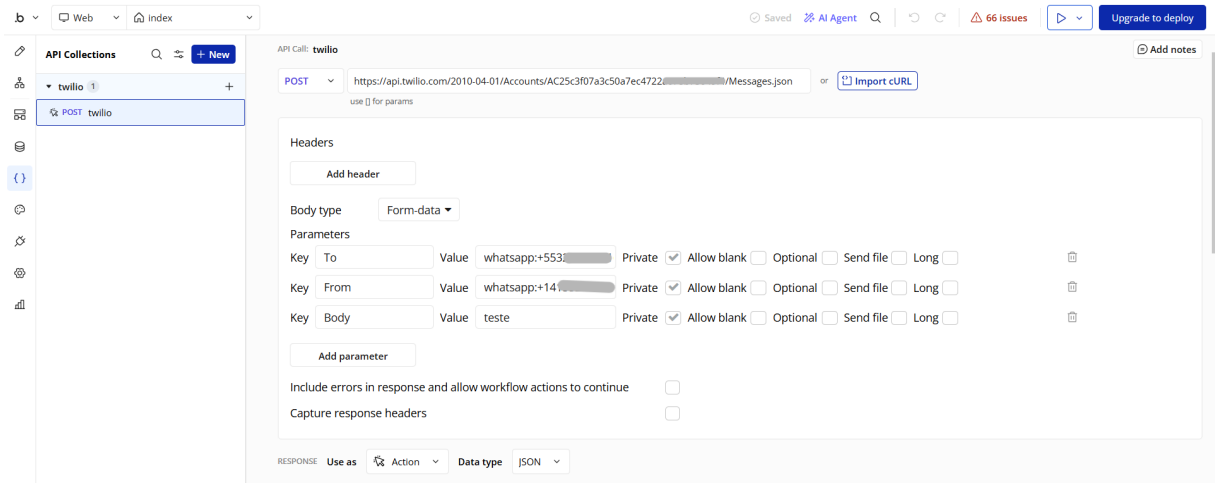

Figura 5.2: Configuração manual de requisição para consumo da API do *Twilio* na *Bubble*.

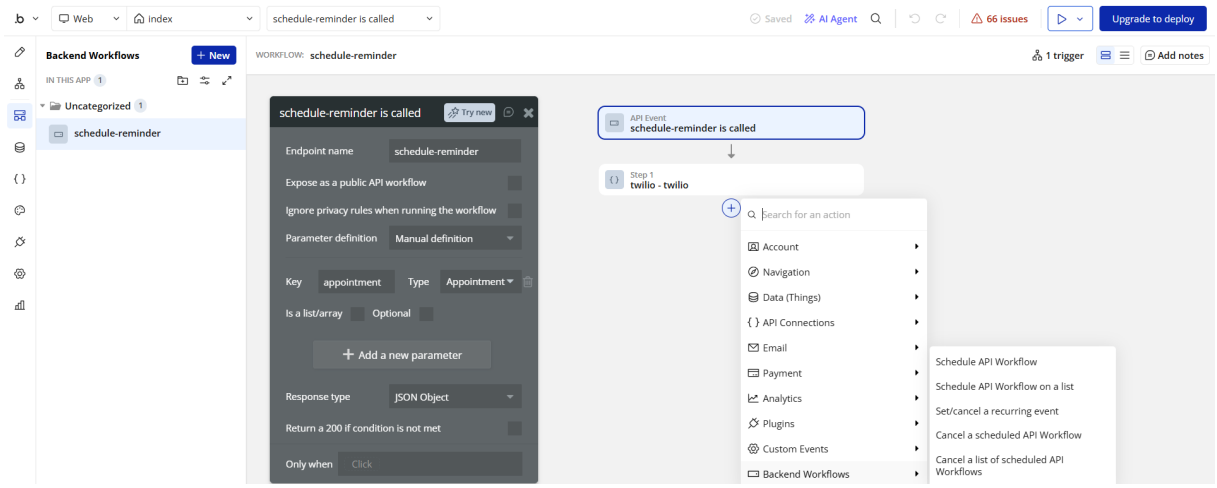

Figura 5.3: *Workflow* back-end configurado na *Bubble* para acionar a integração externa com o serviço de envio de mensagens.

cimento sobre consumo de APIs e estrutura de requisições. Já a Figura 5.3 mostra o *workflow* back-end necessário para acionar essa integração a partir da aplicação. A necessidade de relacionar a chamada externa a eventos e dados da aplicação também impactou o critério de configuração de automações, pois exigiu compreensão da lógica de execução dos fluxos. Dessa forma, embora a integração tenha sido possível, ela não correspondeu plenamente à expectativa de simplicidade associada a uma ferramenta *no-code*.

A facilidade de debug foi considerada insatisfatória. A Figura 5.4 mostra a aba de erros e alertas da plataforma durante o desenvolvimento da PoC. Embora a existência desse recurso seja positiva, a quantidade de mensagens exibidas dificultou a identificação precisa dos problemas que impediam o funcionamento correto da aplicação. Assim, a figura evidencia que o problema não foi a ausência de mecanismo de alerta, mas a difi-

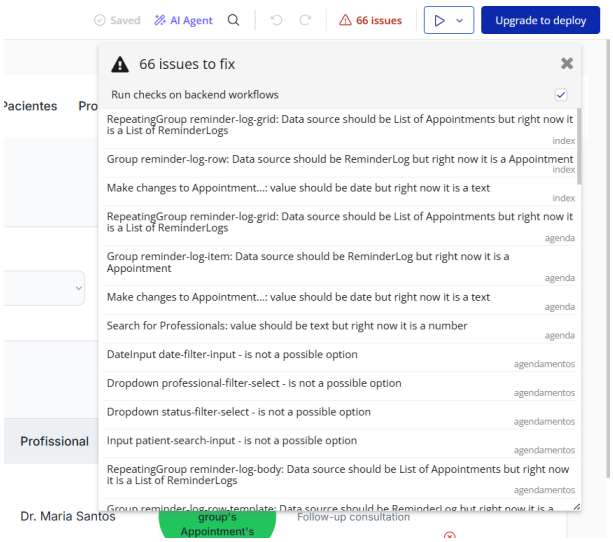

Figura 5.4: Aba de erros e alertas da *Bubble*, indicando o volume de mensagens geradas durante o desenvolvimento da PoC.

culdade de interpretar, priorizar e relacionar as mensagens apresentadas aos elementos específicos que precisavam ser corrigidos.

A configuração de automações também foi avaliada como insatisfatória. Embora a *Bubble* permita construir *workflows*, a experiência dependeu fortemente da organização prévia dos elementos da interface e da compreensão de eventos, lógica condicional e chamadas externas. A Figura 5.5 exemplifica esse processo ao mostrar a configuração manual da ação de um botão no front-end. A figura evidencia que, mesmo para uma interação simples da interface, foi necessário selecionar o evento correspondente, definir a ação a ser executada e relacioná-la aos elementos da aplicação. Esse comportamento impactou o critério de configuração de automações, pois a criação dos fluxos exigiu atenção à lógica de execução e à associação correta entre interface, dados e ações.

Por fim, a dependência da plataforma foi avaliada como muito insatisfatória, devido à forte dependência da infraestrutura proprietária da *Bubble*, dos plugins e do ambiente da própria ferramenta. A necessidade de conhecimento técnico também foi avaliada como muito insatisfatória, não porque conhecimento técnico seja negativo em si, mas porque ele se mostrou maior do que o esperado para uma plataforma *no-code*. Assim, a experiência prática indicou que a *Bubble* oferece uma entrada inicial rápida, mas perde eficiência quando a aplicação exige refinamento, integração externa e controle mais detalhado da lógica do sistema.

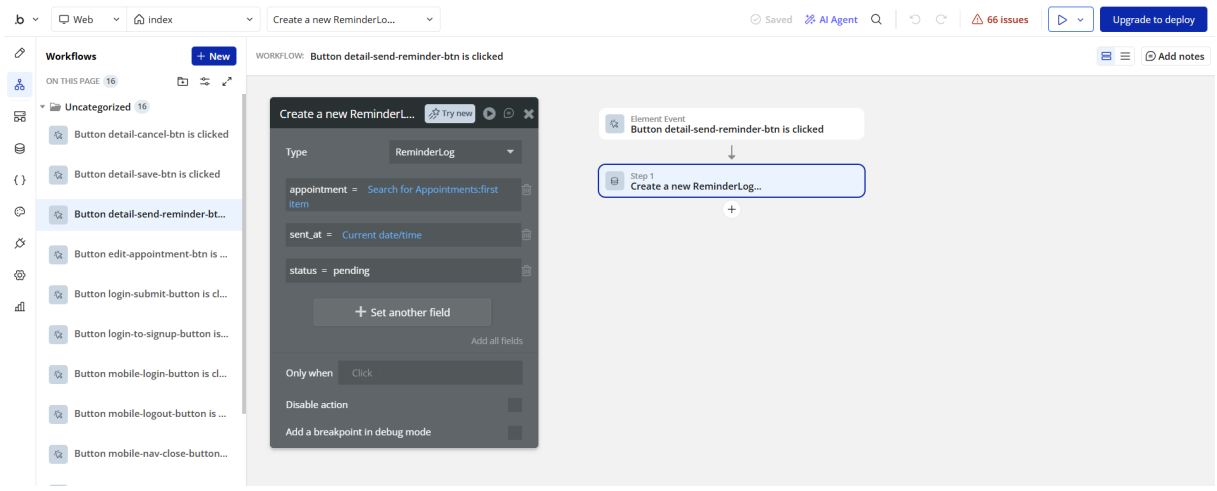

Figura 5.5: Configuração de *workflow* no front-end da *Bubble*, exemplificando a definição manual da ação de um botão.

5.2 Análise da plataforma *Budibase*

A implementação da PoC na *Budibase* apresentou uma experiência diferente da observada na *Bubble*. A plataforma não gerou a aplicação completa por inteligência artificial a partir de um único *prompt*. Durante o início do desenvolvimento, foram observadas diferentes possibilidades para criação da base da aplicação, incluindo a criação manual de tabelas, o uso de dados de exemplo, a importação de arquivos, a conexão com fontes externas e o apoio de IA para estruturação de dados. Neste estudo, o uso de IA esteve mais associado à criação de tabelas, campos e relacionamentos na seção de dados da plataforma, enquanto as demais etapas foram conduzidas por meio dos recursos visuais e configuráveis da própria *Budibase*.

A Tabela 5.2 apresenta o quadro completo de avaliação da plataforma *Budibase*, contendo o grupo avaliativo, o critério analisado, a nota atribuída e a justificativa correspondente. A descrição conceitual dos critérios não é repetida nesta seção, pois já foi apresentada na Subseção 4.2.2. Na sequência, esses resultados são discutidos com apoio das evidências observadas durante a implementação da PoC.

Tabela 5.2: Quadro completo de avaliação da plataforma *Budibase*.

Grupo	**Critério**	**Nota**	**Justificativa**
Usabilidade	Facilidade de início	4 – Satisfatório	Apesar de não gerar a aplicação inteira por IA, a criação das tabelas e telas a partir da estrutura de dados foi direta e coerente com a proposta *low-code*.
Agilidade e evolução	Velocidade de desenvolvimento	4 – Satisfatório	A geração automática de CRUD a partir das tabelas acelerou o processo de criação do sistema e reduziu bastante o trabalho manual.
Agilidade e evolução	Qualidade da aplicação gerada automaticamente	4 – Satisfatório	A geração automática foi funcional, porém básica e limitada à estrutura de dados. Não apresentou bugs relevantes durante a PoC.
Agilidade e evolução	Facilidade de ajuste e evolução pós-geração	4 – Satisfatório	A estrutura da plataforma é mais organizada, o que facilita ajustes, correções e evolução da aplicação após a geração inicial.
Desenvolvimento e flexibilidade	Modelagem de dados	4 – Satisfatório	O suporte com IA para tabelas e relacionamentos ajudou bastante, embora ainda exija prompt bem elaborado e possíveis ajustes manuais.
Desenvolvimento e flexibilidade	Criação de interface (UI)	3 – Neutro	A interface gerada é funcional, mas visualmente básica e exige personalização para um resultado mais refinado.
Desenvolvimento e flexibilidade	Flexibilidade para customizações	3 – Neutro	Há limite de personalização devido aos widgets disponíveis, embora a plataforma ofereça possibilidades técnicas úteis para a PoC.
Integrações	Integração com APIs externas	4 – Satisfatório	Possui recursos adequados para APIs, ampla biblioteca de plugins e conexões externas, embora exija conhecimento técnico, o que é compatível com a proposta *low-code*.
Integrações	Configuração de automações	3 – Neutro	O processo é baseado em eventos, como inserção em tabelas, o que torna a configuração estruturada. Ainda assim, requer conhecimento técnico.
Limitações e riscos	Dependência da plataforma (*vendor lock-in*)	1 – Muito insatisfatório	Há dependência relevante do ambiente da plataforma e limitações para transportar o sistema para fora dela.
Limitações e riscos	Necessidade de conhecimento técnico	4 – Satisfatório	A plataforma exige conhecimento técnico, mas isso é esperado em uma ferramenta *low-code*. A exigência não contradiz a proposta da ferramenta.

Grupo	**Critério**	**Nota**	**Justificativa**
Limitações e riscos	Facilidade de debug	N/A – Não se aplica	Não houve cenário relevante de debug durante a PoC que permitisse avaliar esse critério com segurança.
Usabilidade	UX da plataforma	4 – Satisfatório	A plataforma se mostrou organizada, consistente e mais previsível durante o desenvolvimento.
Documentação e suporte	Documentação e suporte	2 – Insatisfatório	Parte da documentação parece desatualizada em relação à versão atual da plataforma.

Embora a *Budibase* não tenha realizado a geração inicial da aplicação a partir de um único *prompt*, a facilidade de início foi avaliada como satisfatória, pois a criação da estrutura inicial ocorreu de forma direta e coerente com a proposta *low-code*.

A Figura 5.6 ilustra a etapa inicial de criação da base de dados na *Budibase*. A figura mostra que a plataforma oferece diferentes caminhos para iniciar a modelagem dos dados, como criação manual de tabelas, uso de dados de exemplo, importação de arquivos e conexão com fontes externas, como bancos relacionais e serviços externos. Esse comportamento contribuiu positivamente para o critério de facilidade de início, pois a plataforma apresentou alternativas claras para iniciar a construção da aplicação a partir da camada de dados. Ao mesmo tempo, a presença de opções como *MySQL*, *PostgreSQL*, *Oracle* e outras fontes externas também evidencia que a ferramenta pressupõe algum grau de familiaridade técnica, característica compatível com sua proposta *low-code*.

A modelagem de dados foi avaliada como satisfatória. A Figura 5.7 mostra a estrutura de dados criada para a PoC, contemplando entidades como pacientes, médicos, agendamentos e mensagens. A organização apresentada na figura evidencia uma separação clara entre as tabelas e seus respectivos campos, além da possibilidade de gerenciar registros diretamente pela interface da plataforma. Essa estrutura contribuiu para tornar o desenvolvimento mais previsível, pois as telas e formulários puderam ser gerados a partir das entidades previamente definidas. Apesar disso, a criação dessa estrutura ainda exigiu entendimento sobre entidades, atributos e relacionamentos, além de ajustes manuais após a geração inicial, o que reforça a natureza *low-code* da ferramenta.

A velocidade de desenvolvimento também foi avaliada como satisfatória. A principal razão foi a geração automática de telas do tipo *CRUD* a partir das tabelas criadas. A

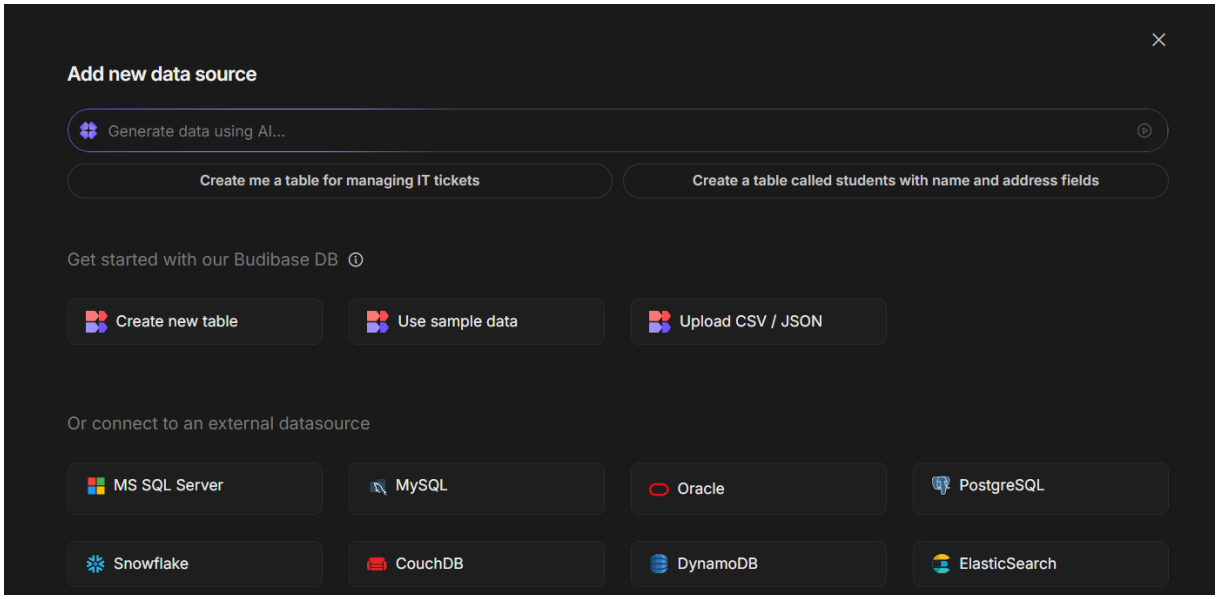

Figura 5.6: Tela inicial para criação de base de dados na *Budibase*, com opções de criação manual, importação de dados e conexão com fontes externas.

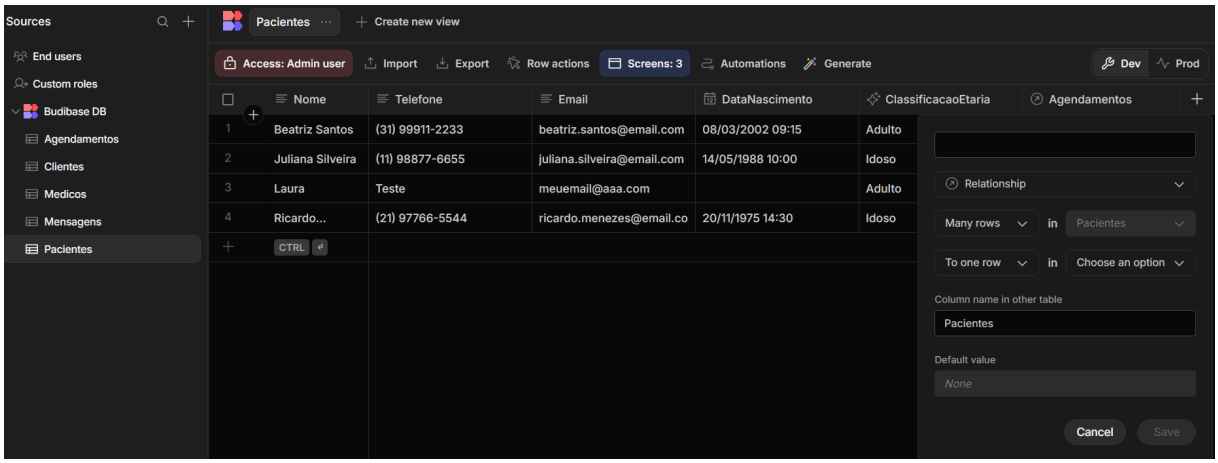

Figura 5.7: Estrutura de dados criada na *Budibase* para a PoC, evidenciando a organização das entidades e campos utilizados na aplicação.

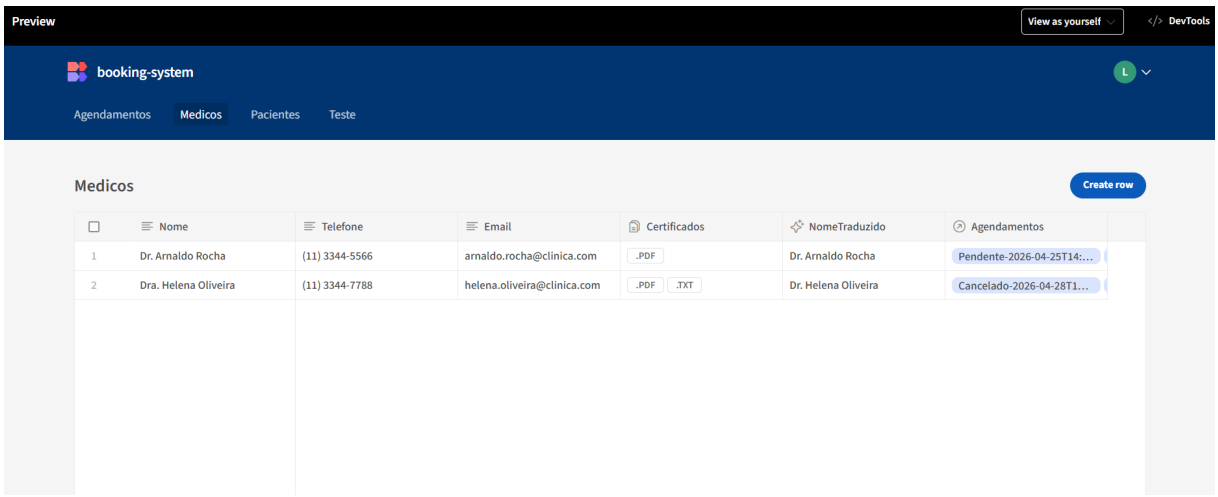

Figura 5.8: Tela do tipo *CRUD* gerada automaticamente pela *Budibase* a partir da estrutura de dados definida.

Figura 5.8 evidencia esse comportamento ao apresentar uma tela gerada automaticamente para listagem de registros, com colunas, dados e opções de interação já estruturadas. Esse recurso reduziu o trabalho operacional necessário para criar telas básicas de consulta, cadastro e edição, tornando o desenvolvimento mais rápido e previsível em comparação à experiência observada na *Bubble*. Embora a interface gerada fosse simples, ela se mostrou funcional e alinhada ao objetivo da PoC.

A qualidade da aplicação gerada automaticamente foi avaliada como satisfatória porque os elementos criados pela plataforma eram básicos, mas funcionais. A Figura 5.9 apresenta um formulário de cadastro de paciente gerado automaticamente a partir da estrutura de dados. A figura mostra que os campos definidos na tabela foram convertidos em componentes de entrada na interface, permitindo a criação de registros sem necessidade de construir manualmente cada elemento visual. Esse resultado contribuiu também para a avaliação satisfatória da facilidade de edição pós-geração, pois a organização entre tabela, tela e formulário facilitou a manutenção e evolução da aplicação. Ainda assim, a interface gerada permaneceu visualmente simples, o que limita sua avaliação no critério de criação de interface.

No grupo de desenvolvimento e flexibilidade, a criação de interface foi avaliada como neutra. Como observado nas Figuras 5.8 e 5.9, a *Budibase* conseguiu gerar telas funcionais a partir da estrutura de dados, incluindo listagens e formulários. No entanto, essas telas apresentaram um padrão visual básico, com pouca sofisticação estética e de-

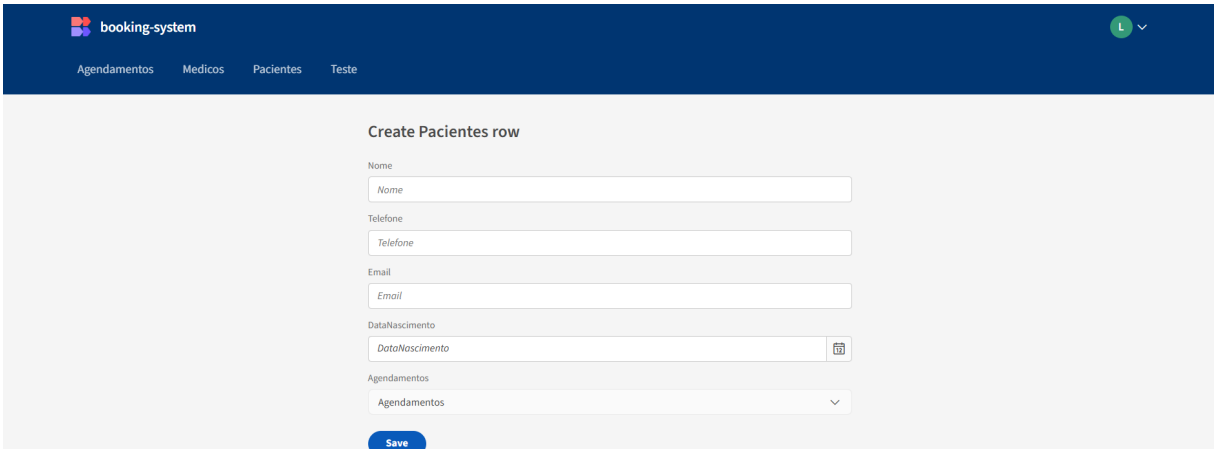

Figura 5.9: Formulário de cadastro de paciente gerado automaticamente pela *Budibase* a partir dos campos definidos na tabela.

pendência de customizações, caso o objetivo fosse obter uma interface mais refinada ou próxima de um produto final. Assim, a plataforma se mostrou eficiente para gerar rapidamente uma interface administrativa funcional, mas não necessariamente para produzir uma experiência visual mais elaborada.

A flexibilidade para customizações foi avaliada como neutra. A plataforma oferece possibilidades técnicas relevantes, como ajustes de componentes, configuração de telas, uso de *widgets*, conexões externas e automações. Entretanto, essas possibilidades estão condicionadas aos recursos disponibilizados pela própria ferramenta. Desse modo, embora a *Budibase* tenha apresentado uma estrutura mais organizada e previsível para evolução do sistema, principalmente quando comparada à experiência na *Bubble*, ainda há limitações associadas ao nível de personalização permitido pelos componentes disponíveis.

Nas integrações, a *Budibase* recebeu avaliação satisfatória para conexão com APIs externas. A plataforma possui recursos adequados para APIs, plugins e conectores, embora seu uso exija conhecimento técnico. Essa exigência, contudo, foi considerada compatível com a proposta *low-code*. Diferentemente da *Bubble*, em que a configuração da integração exigiu maior esforço para organizar chamadas externas e fluxos, a *Budibase* apresentou uma estrutura mais previsível para trabalhar com dados e conexões. Ainda assim, por se tratar de uma atividade que envolve requisições, parâmetros, autenticação e manipulação de dados, esse critério não elimina a necessidade de conhecimento técnico.

A configuração de automações foi avaliada como neutra. A lógica baseada em eventos, como inserção ou alteração em tabelas, tornou o processo mais estruturado, mas

ainda demandou entendimento técnico de conceitos como requisições, eventos, gatilhos e manipulação de dados. Nesse sentido, as Figuras 5.6, 5.8 e 5.9 ajudam a compreender a base sobre a qual essas automações são construídas: primeiro define-se a estrutura de dados, depois são geradas telas e formulários associados a essa estrutura, e então os eventos podem ser configurados a partir das ações realizadas sobre esses dados. Assim, a automação não foi percebida como um recurso puramente visual e trivial, mas como uma etapa que depende da compreensão da organização interna da aplicação.

Em relação às limitações e riscos, a dependência da plataforma foi avaliada como muito insatisfatória. Assim como ocorre em outras ferramentas *low-code/no-code*, há dependência relevante do ambiente da plataforma e limitações para transportar o sistema para fora dela. As próprias Figuras 5.7, 5.8 e 5.9 evidenciam que a aplicação é construída dentro de uma estrutura própria da *Budibase*, com tabelas, telas e formulários gerenciados pelo ambiente da ferramenta. Essa organização favorece a produtividade durante o desenvolvimento, mas também reforça a dependência em relação ao ecossistema da plataforma.

A necessidade de conhecimento técnico, por outro lado, foi avaliada como satisfatória. Isso ocorre porque a *Budibase* exige conhecimento sobre dados, relacionamentos, eventos e integrações, mas essa exigência é coerente com sua proposta *low-code* e não contradiz o posicionamento da ferramenta. A facilidade de debug foi marcada como não aplicável, pois não houve cenário relevante de debug durante a PoC que permitisse avaliar esse critério com segurança.

A clareza da plataforma foi avaliada como satisfatória, uma vez que a *Budibase* se mostrou organizada, consistente e mais previsível durante o desenvolvimento. Essa percepção é reforçada pela sequência apresentada nas Figuras 5.6, 5.7, 5.8 e 5.9, que mostram um fluxo relativamente claro: criação da base de dados, estruturação das entidades, geração de telas *CRUD* e geração de formulários. A documentação e suporte, por outro lado, foram avaliados como insatisfatórios, pois parte da documentação consultada aparentou estar desatualizada em relação à versão atual da plataforma. De modo geral, a *Budibase* apresentou uma experiência mais produtiva e estruturada do que a *Bubble*, especialmente por reduzir o trabalho manual e oferecer uma lógica mais próxima de práticas

tradicionais de desenvolvimento.

5.3 Comparação com aplicação gerada por IA generativa

Considerando o cenário atual de desenvolvimento apoiado por ferramentas de IA generativa, esta pesquisa buscou também realizar uma comparação complementar entre o esforço de desenvolvimento observado nas plataformas *low-code* e *no-code* avaliadas e o esforço necessário para gerar uma versão alternativa da PoC com o uso da ferramenta *Lovable*. Essa comparação não pretende esgotar a discussão sobre diferenças entre plataformas *low-code/no-code* e ferramentas de IA generativa, nem atribuir notas à *Lovable* na mesma matriz aplicada à *Bubble* e à *Budibase*. O protocolo avaliativo definido neste trabalho, apresentado na Subseção 4.2.2 e consolidado na Tabela 5.3, foi construído especificamente para analisar plataformas *low-code* e *no-code* a partir de critérios como modelagem de dados, criação de interface, integração com APIs externas, configuração de automações, dependência da plataforma, facilidade de edição pós-geração e documentação.

A *Lovable*, por sua vez, não foi submetida ao mesmo protocolo porque possui uma natureza diferente das plataformas avaliadas. Enquanto *Bubble* e *Budibase* foram analisadas como ambientes de desenvolvimento visual, nos quais a aplicação é construída, configurada e ajustada dentro da própria plataforma, a *Lovable* foi utilizada como uma ferramenta de geração de aplicação por linguagem natural, com foco na criação rápida de uma versão inicial da PoC a partir de *prompt*. Dessa forma, alguns critérios definidos para plataformas *low-code/no-code*, como configuração manual de automações, modelagem de dados dentro da plataforma, clareza dos menus da ferramenta e facilidade de edição por componentes visuais, não seriam diretamente comparáveis ou exigiriam adaptações metodológicas específicas. Por esse motivo, a comparação realizada nesta seção tem caráter exploratório e contextual, servindo apenas para observar indícios sobre velocidade inicial de geração, qualidade visual percebida e esforço inicial de prototipação, sem substituir a avaliação formal aplicada às plataformas *Bubble* e *Budibase*.

Ainda assim, considera-se relevante destacar diferenças observadas em relação ao

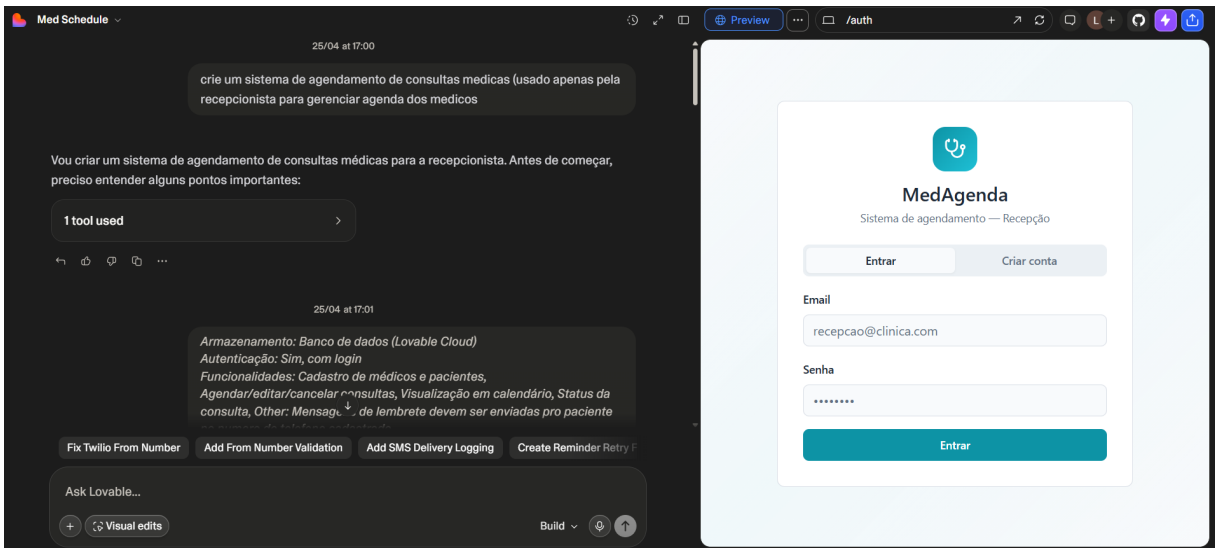

Figura 5.10: Prompt inicial na ferramenta *Lovable* e resultado compilado

esforço de desenvolvimento, à curva de aprendizado, à qualidade visual percebida e ao potencial de evolução da aplicação nas duas abordagens.

Para isso, foi gerada uma versão alternativa da aplicação utilizando a *Lovable*. Essa versão foi produzida em 21 minutos e apresentou resultado visualmente mais refinado e funcional em uma primeira observação, especialmente em termos de qualidade visual percebida e rapidez de geração. No entanto, esse tempo foi registrado apenas para a geração da versão alternativa pela ferramenta de IA generativa, não havendo medição formal equivalente do tempo total dedicado às implementações na *Bubble* e na *Budibase*. Por esse motivo, não se pretende estabelecer uma comparação quantitativa direta entre as abordagens. Ainda assim, durante a implementação nas plataformas *low-code/no-code*, observou-se um esforço consideravelmente maior, envolvendo ajustes manuais, configuração de integrações, estudo da documentação, revisão da estrutura da aplicação e resolução de limitações específicas de cada plataforma.

Esse resultado sugere que a promessa de agilidade tradicionalmente associada às plataformas *low-code* e *no-code* passou a ser tensionada pelo avanço recente das ferramentas de IA generativa aplicadas ao desenvolvimento de software. Sauvola et al. (SAUVOLA et al., 2024) discutem que a IA generativa pode apoiar tarefas criativas e automatizar atividades repetitivas no desenvolvimento. De forma complementar, Monteiro et al. (MONTEIRO et al., 2025) mostram que interfaces *no-code* baseadas em modelos de linguagem podem apoiar a construção de pequenas aplicações web a partir de requisitos

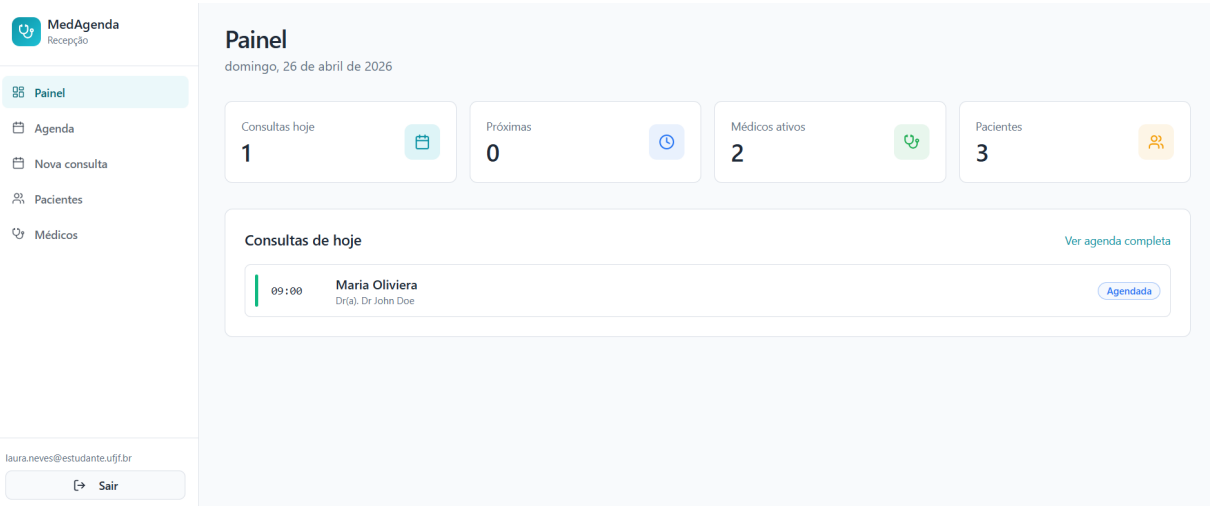

Figura 5.11: Tela inicial após login, desenvolvido na ferramenta *Lovable*

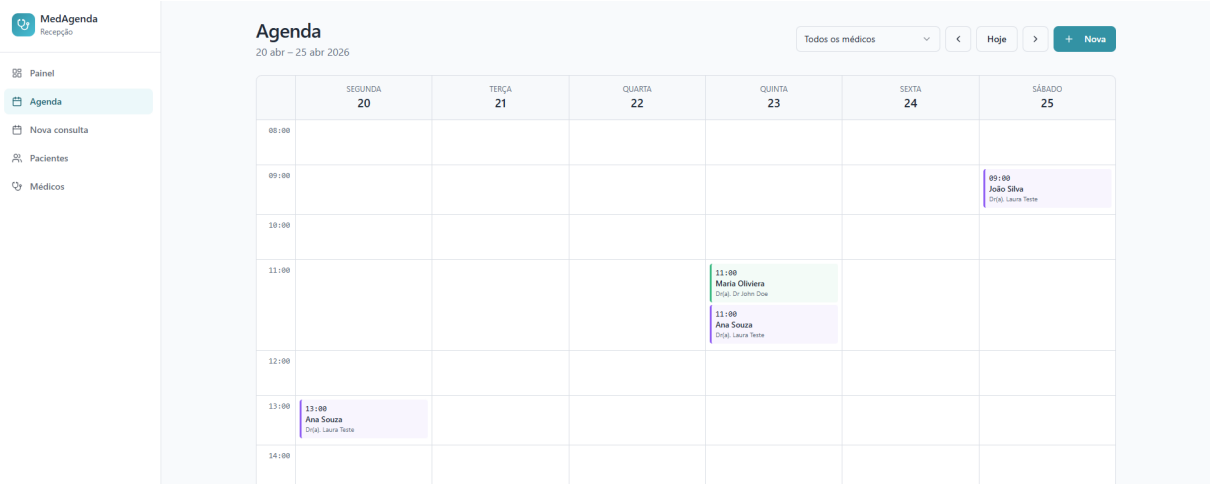

Figura 5.12: Tela de agenda concentrando todos agendamentos da semana e filtros, na ferramenta *Lovable*

descritos em linguagem natural. Já Bruhin et al. (BRUHIN et al., 2024) indicam que a IA generativa tende a modificar a forma como plataformas *low-code* são utilizadas e avaliadas.

5.4 Visão geral da avaliação

Após a análise individual das plataformas e da comparação complementar com uma aplicação gerada por IA generativa, esta seção apresenta uma visão consolidada das notas atribuídas à *Bubble* e à *Budibase* nos critérios avaliativos definidos neste trabalho. Essa consolidação tem como objetivo organizar, de forma sintética, as diferenças observadas durante a implementação da PoC, sem substituir as justificativas qualitativas apresentadas nas seções anteriores.

É importante destacar que, nas plataformas avaliadas, os recursos de inteligência artificial foram tratados como parte da experiência nativa de uso de cada ferramenta, e não como uma variável isolada de comparação. Assim, quando a plataforma ofereceu um fluxo inicial baseado em IA, esse recurso foi utilizado como parte do percurso sugerido ao usuário. No caso da *Bubble*, a IA foi empregada como ponto de partida para geração inicial da aplicação, conforme o fluxo de *onboarding* da própria plataforma. Já na *Budibase*, o uso de IA esteve mais associado à criação e estruturação de dados, enquanto a construção das telas pôde ser realizada automaticamente a partir das tabelas criadas no banco de dados por meio dos recursos nativos da plataforma, sendo posteriormente ajustada e complementada conforme as necessidades da aplicação. Dessa forma, a avaliação não comparou versões alternativas da mesma PoC com e sem IA em cada plataforma, mas observou como os recursos de IA disponíveis influenciaram a experiência prática de desenvolvimento em cada ambiente.

A Tabela 5.3 e a Figura 5.13 apresentam a comparação das notas atribuídas às duas plataformas em cada critério avaliado. Considerando a média simples das notas aplicáveis, a *Bubble* obteve média de 2,14 nos 14 critérios avaliados, enquanto a *Budibase* obteve média de 3,38 nos 13 critérios aplicáveis, uma vez que o critério de facilidade de debug foi classificado como N/A para essa plataforma.

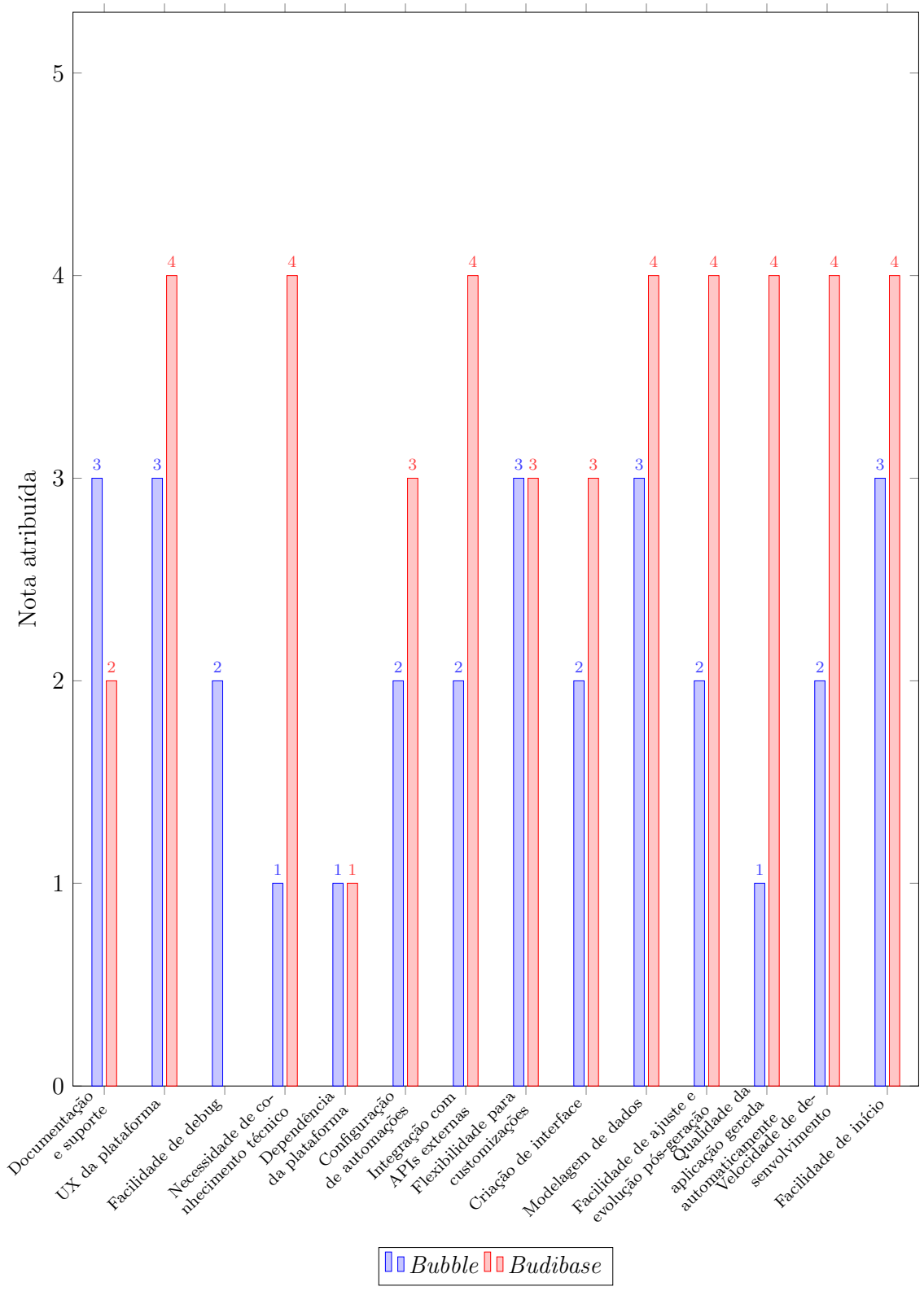

Figura 5.13: Comparação das notas atribuídas às plataformas *Bubble* e *Budibase* nos critérios avaliativos do estudo. O critério "Facilidade de debug" foi classificado como N/A para a *Budibase* e, por isso, não possui barra correspondente.

Tabela 5.3: Notas atribuídas às plataformas avaliadas.

Critério	***Bubble***	***Budibase***
Velocidade de desenvolvimento	2	4
Qualidade da aplicação gerada automaticamente	1	4
Facilidade de ajuste e evolução pós-geração	2	4
Modelagem de dados	3	4
Criação de interface (UI)	2	3
Flexibilidade para customizações	3	3
Integração com APIs externas	2	4
Configuração de automações	2	3
Dependência da plataforma	1	1
Necessidade de conhecimento técnico	1	4
Facilidade de debug	2	N/A
Facilidade de início	3	4
UX da plataforma	3	4
Documentação e suporte	3	2
Média das notas aplicáveis	**2,14**	**3,38**

5.5 Síntese dos resultados

A análise consolidada indica que a classificação de uma ferramenta como *no-code* não garante, por si só, maior velocidade no desenvolvimento de software. A avaliação deve considerar também fatores como esforço de configuração, flexibilidade, funcionamento das integrações, dependência de conhecimento técnico e facilidade de evolução da paplicação. O uso do sistema de métricas definido neste trabalho permitiu observar, de forma estruturada, como essas características se manifestaram durante o desenvolvimento da prova de conceito nas plataformas *Bubble* e *Budibase*.

No caso da *Bubble*, observou-se que a existência de geração da aplicação inicial por IA acelerou o inicio do desenvolvimento. Mas a evolução do sistema exigiu uma quantidade significativa de correções, incluindo ajustes visuais, reorganização de elementos, configuração manual de workflows e integração com serviços externos. Esses aspectos impactaram negativamente, especialmente nos critérios de velocidade de desenvolvimento, qualidade da aplicação gerada automaticamente, facilidade de edição pós-geração, criação de interface, integração com APIs externas, automações, dependência da plataforma e necessidade de conhecimento técnico. Dessa forma, a experiência prática demonstrou que a classificação de uma ferramenta como *no-code* não garante, por si só, simplicidade plena ou baixa exigência técnica durante todo o processo de desenvolvimento.

A *Budibase*, por outro lado, não gerou a aplicação completa automaticamente,

mas apresentou um fluxo de desenvolvimento mais estruturado. A criação de tabelas, relacionamentos e telas CRUD reduziu o esforço manual e tornou a evolução do sistema mais previsível. A exigência de conhecimento técnico foi mais perceptível, especialmente em aspectos relacionados a dados, integrações e automações, mas essa característica se mostrou compatível com a proposta *low-code* da ferramenta. Assim, no contexto avaliado, a *Budibase* apresentou melhor equilíbrio entre produtividade, organização interna e controle sobre a aplicação.

A partir desses resultados, observa-se que o uso de recursos de IA dentro das plataformas avaliadas apresentou vantagens principalmente na redução do esforço inicial de configuração. Na *Bubble*, esse ganho apareceu na geração rápida de uma base visual para a aplicação; na *Budibase*, apareceu no apoio à estruturação inicial de tabelas e relacionamentos. Entretanto, os resultados também indicaram limitações. Na *Bubble*, a geração inicial por IA produziu uma aplicação visualmente útil como ponto de partida, mas com problemas de funcionamento, organização interna e necessidade significativa de ajustes manuais. Na *Budibase*, o apoio por IA foi mais restrito à camada de dados, enquanto as telas e formulários foram gerados a partir das tabelas criadas e posteriormente ajustados por meio dos recursos próprios da plataforma. Assim, a IA interna dessas ferramentas contribuiu para acelerar etapas iniciais, mas não eliminou a necessidade de conhecimento técnico, revisão manual e compreensão da lógica de funcionamento de cada ambiente.

Como discussão complementar, a comparação pontual com uma aplicação gerada por ferramenta de inteligência artificial generativa, no caso a *Lovable*, trouxe um elemento relevante para contextualizar o cenário atual de desenvolvimento de software. Embora essa ferramenta não tenha sido submetida ao mesmo protocolo avaliativo aplicado à *Bubble* e à *Budibase*, a geração rápida de uma versão alternativa da PoC indicou que soluções baseadas em IA generativa podem tensionar a proposta de valor tradicionalmente associada às plataformas *low-code* e *no-code*. Em especial, observou-se que a IA apresentou vantagem em relação à velocidade inicial de geração, à qualidade visual percebida e ao potencial de evolução da aplicação.

Dessa forma, os resultados sugerem que plataformas *low-code* e *no-code* ainda

podem possuir valor em contextos específicos, especialmente quando oferecem ambientes integrados, componentes prontos e recursos de automação. No entanto, sua adoção deve ser analisada de forma criteriosa, considerando não apenas a promessa de rapidez, mas também os custos de manutenção, as limitações de customização, a dependência da plataforma e a necessidade de conhecimento técnico para evoluir a aplicação.

6 Conclusão

Este trabalho teve como objetivo avaliar o uso de plataformas *low-code* e *no-code* no desenvolvimento de software, com base em um sistema de métricas avaliativas aplicado à implementação de uma Prova de Conceito (PoC). Para isso, foi conduzida uma pesquisa de caráter aplicado, exploratório e predominantemente qualitativo, apoiada por revisão bibliográfica e por um estudo prático envolvendo duas plataformas: *Bubble*, classificada neste estudo como uma ferramenta *no-code*, e *Budibase*, classificada como uma ferramenta *low-code*.

A PoC desenvolvida consistiu em um sistema de agendamento de consultas médicas, contemplando funcionalidades como cadastro de pacientes, profissionais e agendamentos, visualização de informações em telas do tipo *CRUD*, automações e integração com serviço externo para envio de mensagens. A avaliação foi conduzida com base em critérios organizados em grupos relacionados à agilidade e evolução, desenvolvimento e flexibilidade, integrações, limitações e riscos, usabilidade, e documentação e suporte. Além disso, foi realizada uma comparação complementar com uma aplicação gerada por IA generativa, utilizando a ferramenta *Lovable*, com o objetivo de contextualizar os resultados diante do cenário recente de desenvolvimento apoiado por inteligência artificial.

De modo geral, a pesquisa atingiu seus objetivos ao propor e aplicar um conjunto de critérios avaliativos para analisar plataformas *low-code* e *no-code* em um cenário prático de desenvolvimento. A aplicação da PoC permitiu observar que a classificação de uma ferramenta como *low-code* ou *no-code* não garante, por si só, maior produtividade, menor complexidade ou menor necessidade de conhecimento técnico. Os resultados demonstraram que os ganhos e limitações dependem da estrutura da plataforma, do tipo de aplicação desenvolvida, do nível de customização exigido, das integrações necessárias e da capacidade de evolução da aplicação após sua criação inicial.

Em relação à **RQ1**, que questiona como realizar uma avaliação prática de plataformas *low-code* e *no-code* no desenvolvimento de software com base em um sistema de métricas avaliativas, o trabalho demonstrou que essa avaliação pode ser conduzida a

partir da definição prévia de uma PoC comum, da seleção de critérios avaliativos e da aplicação de uma escala de notas acompanhada de justificativas qualitativas. A matriz avaliativa proposta permitiu organizar a análise de forma mais estruturada, evitando que a comparação ficasse restrita apenas à velocidade de desenvolvimento ou à percepção subjetiva de facilidade de uso. Dessa forma, a avaliação considerou aspectos como modelagem de dados, criação de interface, integrações, automações, dependência da plataforma, documentação, necessidade de conhecimento técnico e facilidade de edição pós-geração.

Em relação à **RQ2**, que busca identificar possíveis ganhos, limitações e desafios no uso de plataformas *low-code* e *no-code*, os resultados indicaram que essas ferramentas podem oferecer ganhos relevantes em etapas iniciais do desenvolvimento, especialmente na criação de estruturas básicas, telas administrativas e operações *CRUD*. No caso da *Budibase*, a geração automática de telas a partir da estrutura de dados contribuiu para uma experiência mais organizada e previsível. Por outro lado, também foram observadas limitações importantes, como dependência da plataforma, restrições de customização, necessidade de conhecimento técnico para lidar com integrações e automações, além de dificuldades relacionadas à documentação e à manutenção da aplicação. No caso da *Bubble*, embora a geração inicial por IA tenha acelerado o início do desenvolvimento, a evolução da aplicação exigiu correções visuais, reconfiguração de fluxos, ajustes manuais e maior esforço técnico do que o esperado para uma ferramenta *no-code*.

Em relação à **RQ3**, que discute em que medida ferramentas de desenvolvimento baseadas em IA generativa podem influenciar a utilidade percebida e a proposta de valor das plataformas *low-code* e *no-code*, a comparação complementar com a *Lovable* indicou que ferramentas desse tipo podem tensionar a proposta de valor tradicionalmente associada às plataformas avaliadas. A versão alternativa da PoC gerada por IA foi produzida em tempo reduzido e apresentou boa qualidade visual percebida, sugerindo que ferramentas generativas podem competir especialmente em aspectos como velocidade inicial, prototipação e geração de interfaces. Entretanto, essa comparação foi exploratória e não substitui uma avaliação formal da *Lovable* com os mesmos critérios aplicados à *Bubble* e à *Budibase*. Assim, não se conclui que plataformas *low-code* e *no-code* tenham se tornado obsoletas, mas sim que sua proposta de valor precisa ser reavaliada diante do avanço

recente das ferramentas de IA generativa aplicadas ao desenvolvimento de software.

Uma limitação importante desta pesquisa está relacionada à cronologia do próprio tema investigado. Parte das discussões sobre plataformas *low-code* e *no-code* foi construída em um momento anterior à popularização recente das ferramentas de IA generativa aplicadas ao desenvolvimento de software. Dessa forma, algumas premissas tradicionalmente associadas a essas plataformas, como aceleração do desenvolvimento e redução de esforço manual, passaram a ser comparadas com um cenário tecnológico em rápida transformação. Essa mudança não invalida a pesquisa, mas reforça a necessidade de interpretar seus resultados considerando o momento em que o estudo foi conduzido.

Outra limitação diz respeito ao número de plataformas avaliadas. O estudo prático foi realizado apenas com *Bubble* e *Budibase*, escolhidas como casos representativos e viáveis dentro do tempo disponível. Embora essa escolha tenha permitido uma análise mais detalhada das duas ferramentas, os resultados não podem ser generalizados para todas as plataformas *low-code* e *no-code* existentes. Ferramentas como *OutSystems*, *Mendix*, *Power Apps* e outras soluções poderiam apresentar comportamentos diferentes em relação aos mesmos critérios.

Também se destaca como limitação o fato de a avaliação ter sido baseada em uma única PoC. O sistema de agendamento de consultas médicas permitiu observar funcionalidades relevantes, como *CRUD*, relacionamentos entre entidades, automações e integração externa. No entanto, outros tipos de sistemas poderiam exigir requisitos distintos, como maior escalabilidade, controle de permissões mais refinado, regras de negócio mais complexas, requisitos rigorosos de segurança, integração com sistemas legados ou maior volume de usuários. Assim, os resultados refletem o comportamento das plataformas no cenário específico analisado.

Além disso, a avaliação foi conduzida a partir da experiência prática de implementação da PoC. Embora o uso de critérios e justificativas qualitativas tenha buscado tornar a análise mais transparente, ainda existe um componente interpretativo associado à experiência, ao conhecimento técnico prévio e às decisões tomadas durante o desenvolvimento. A inclusão de múltiplos avaliadores ou de usuários com diferentes níveis de conhecimento técnico poderia ampliar a confiabilidade e a diversidade das percepções

obtidas.

A comparação com a *Lovable* também constitui uma limitação específica. A ferramenta foi utilizada apenas como elemento complementar de discussão, a partir de um teste rápido de geração da aplicação, e não foi submetida ao mesmo protocolo avaliativo aplicado às plataformas *Bubble* e *Budibase*. Portanto, os resultados relacionados à IA generativa devem ser interpretados como indícios iniciais, e não como uma comparação conclusiva entre plataformas *low-code/no-code* e ferramentas de desenvolvimento baseadas em IA.

Como trabalhos futuros, sugere-se ampliar a avaliação para um número maior de plataformas, incluindo ferramentas *low-code* mais robustas e amplamente utilizadas em ambientes corporativos. Também seria relevante aplicar a matriz avaliativa proposta em diferentes tipos de sistemas, com níveis variados de complexidade, para verificar se os resultados observados nesta pesquisa se mantêm em outros contextos. Outra possibilidade é envolver múltiplos avaliadores, incluindo desenvolvedores experientes, usuários não técnicos e profissionais de negócio, a fim de comparar percepções sobre facilidade de uso, produtividade, curva de aprendizado e manutenção.

Uma oportunidade relevante de continuidade consiste em aprofundar a comparação entre plataformas *low-code/no-code* e ferramentas de IA generativa. Estudos futuros podem aplicar os mesmos critérios avaliativos a ferramentas como *Lovable* e outras soluções baseadas em geração de aplicações por linguagem natural, permitindo uma comparação mais equilibrada entre as abordagens. Essa linha de investigação pode contribuir para compreender se essas ferramentas atuam como substitutas, complementares ou extensões das plataformas *low-code* e *no-code*.

Por fim, este trabalho contribui ao propor uma forma estruturada de avaliar plataformas *low-code* e *no-code* em um cenário prático de desenvolvimento. A matriz de critérios, a PoC construída e a análise comparativa realizada podem servir como base para avaliações futuras, tanto em contextos acadêmicos quanto em decisões práticas sobre adoção dessas ferramentas. Os resultados reforçam que a escolha por plataformas *low-code* e *no-code* deve considerar não apenas a promessa de agilidade, mas também aspectos como manutenção, flexibilidade, integração, dependência da plataforma e adequação ao

tipo de sistema que se pretende desenvolver.

Bibliografia

ALENCAR, L. de. *Farmácia cidadã - desenvolvimento de um sistema de gestão de distribuição de medicamentos utilizando ferramentas low-code*. Dissertação (Trabalho de Conclusão de Curso) — Instituto Federal do Espírito Santo (IFES), Serra, ES, Brazil, 2023. Orientador: Prof. Felipe Frechiani.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. ⟨https://agilemanifesto.org/⟩.

BOCK, A.; FRANK, U. Low-code platform. *Business Information Systems Engineering*, v. 63, 12 2021.

BRUHIN, O. et al. The rise of generative ai in low code development platforms - an analysis and future directions. In: . [S.l.: s.n.], 2024.

BUBBLE. *Bubble official website*. 2025. ⟨https://bubble.io⟩. "Bubble is an AI app platform that lets anyone build powerful web and mobile apps visually, without code.".

BUDIBASE. *Budibase official website*. 2025. ⟨https://budibase.com⟩. "Budibase is an open-source workflow platform that saves engineers time and energy building apps that integrate with any data source and speed-up any process.".

FERREIRA, D. S. *Uma análise das principais plataformas low-code do mercado*. Dissertação (Trabalho de Conclusão de Curso) — Instituto Federal do Sertão Pernambucano (IFSertãoPE), Salgueiro, PE, Brazil, 2024. Orientador: Prof. Esp. Francisco Junio da Silva Fernandes.

FRANK, U.; MAIER, P.; BOCK, A. *Low code platforms: promises, concepts and prospects. A comparative study of ten systems*. Essen, Germany, 2021. Disponível em: ⟨https://doi.org/10.17185/duepublico/75244⟩.

JOSHI, A. et al. Likert scale: Explored and explained. *British Journal of Applied Science and Technology*, v. 7, n. 4, p. 396–403, 2015.

LAKATOS, E. M.; MARCONI, M. d. A. *Fundamentos de metodologia científica*. 8. ed. [S.l.]: Atlas, 2017.

LIKERT, R. A technique for the measurement of attitudes. *Archives of Psychology*, v. 22, n. 140, p. 1–55, 1932.

LOVABLE. *Lovable official website*. 2026. "Lovable is a full-stack AI development platform for building, iterating on, and deploying web applications using natural language, with real code, security, and enterprise governance." Disponível em: ⟨https://lovable.dev⟩. Acesso em: 15 jun. 2026.

LUO, Y. et al. Characteristics and challenges of low-code development: The practitioners' perspective. In: *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '21)*. Bari, Italy: ACM, 2021. October 11–15, 2021.

MARTINS, J.; BRANCO, F.; MAMEDE, H. S. Combining low-code development with chatgpt to novel no-code approaches: A focus-group study. *Intelligent Systems with Applications*, v. 20, p. 200289, 2023.

MOHAJAN, H. Two criteria for good measurements in research: Validity and reliability. *Annals of Spiru Haret University*, v. 17, n. 3, p. 58–82, 2017.

MONTEIRO, M. et al. Nocodegpt: A no-code interface for building web apps with language models. *Software: Practice and Experience*, v. 55, p. 1408–1424, 2025.

PACHECO, G. N. et al. Natural language processing in software engineering: A systematic literature review. *Journal of Software Engineering Research and Development*, v. 13, n. 16, 2025.

PRESSMAN, R. S.; MAXIM, B. R. *Software Engineering: A Practitioner's Approach*. 9. ed. [S.l.]: McGraw-Hill Education, 2020.

SAUVOLA, J. et al. Future of software development with generative ai. *Automated Software Engineering*, v. 31, n. 26, 2024.

SILVA, J. X. da. *Avaliação de plataformas low code e no code como estratégia para aumento da produtividade no desenvolvimento de software*. Dissertação (Dissertação de Mestrado) — Universidade Federal do Piauí (UFPI), Teresina, PI, Brazil, 2024. Orientador: Prof. Dr. Guilherme Amaral Avelino.

SOMMERVILLE, I. *Software Engineering*. 10. ed. [S.l.]: Pearson, 2016.

VALENTE, M. T. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. [S.l.]: Independente, 2020. ISBN 978-65-00-01950-6.

WHITE, J. et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023.

ZHAO, W. et al. *A Survey of Large Language Models*. 2023.