UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

YASMIN EMILY GOMES MACHADO

PERCEPÇÕES DE PROFISSIONAIS SOBRE O USO DE PLATAFORMAS LOW-CODE NO DESENVOLVIMENTO DE SOFTWARE

SOBRAL

2025

YASMIN EMILY GOMES MACHADO

PERCEPÇÕES DE PROFISSIONAIS SOBRE O USO DE PLATAFORMAS LOW-CODE NO DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Wendley Souza da Silva

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M135p Machado, Yasmin Emily Gomes.
Percepções de profissionais sobre o uso de plataformas low-code no desenvolvimento de software / Yasmin Emily Gomes Machado. – 2025.
48 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Prof. Dr. Wendley Souza da Silva.

1. Low-code. 2. Desenvolvimento de software. 3. Produtividade. I. Título.

CDD 621.39

YASMIN EMILY GOMES MACHADO

PERCEPÇÕES DE PROFISSIONAIS SOBRE O USO DE PLATAFORMAS LOW-CODE NO DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 06 de Agosto de 2025

BANCA EXAMINADORA

Prof. Dr. Wendley Souza da Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Iális Cavalcante de Paula Júnior
Universidade Federal do Ceará (UFC)

Prof. Dr. Thiago Iachiley Araújo de Souza
Universidade Federal do Ceará (UFC)

À Deus, por me conceder forças para chegar até aqui. À minha família, por sempre me apoiar e acreditar em mim, sendo meu alicerce em todos os momentos.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, pela sabedoria, força e por me guiar em todos os momentos desta jornada. Sem Ele, nada disso seria possível.

Este foi, de longe, o percurso mais desafiador que já enfrentei. Foram inúmeras madrugadas em claro, alguns dias sem dormir, abdiquei de grande parte da minha vida e perdi momentos preciosos com a minha família. Mas me dediquei integralmente para concluir essa trajetória nestes cinco anos. E só consegui porque estive cercada de pessoas que me deram apoio, incentivo e amor incondicional.

Aos meus pais, Jane e Luis, minha eterna gratidão por estarem sempre ao meu lado, acreditando em mim até mesmo quando eu mesma duvidava. Pelo amor, pela paciência e por cada gesto de cuidado ao longo desses anos.

Às minhas irmãs, Laysa e Jady, obrigada por me acolherem com carinho e tornarem os dias mais leves com a presença de vocês. Obrigada por cada palavra de incentivo, por me ouvirem, por celebrarem minhas vitórias e por compreenderem quando precisei me ausentar. O apoio, a cumplicidade e o amor de vocês foram essenciais para que eu seguisse em frente com coragem e serenidade. Sou muito grata por ter irmãs tão incríveis ao meu lado.

A toda minha família, que sempre torceu por mim e se orgulhou das minhas conquistas, deixo aqui um agradecimento cheio de amor.

Em especial, à minha avó Raimunda, que infelizmente não pôde viver até este momento, mas que sempre esteve comigo em espírito. Sei que sua força e seu amor me acompanharam em cada passo dessa caminhada. Muito do que sou vem dela, e levo comigo tudo o que me ensinou.

Agradeço à minha amiga Ana Lia, por ser mais que uma amiga — por ter sido minha família ao longo desses anos. Estivemos lado a lado em todos os momentos: nas madrugadas de estudo, nas noites de reflexão, nas conquistas e nas pequenas alegrias. Morar com você foi uma bênção, e sua presença constante foi meu ponto de apoio. Levo você comigo em cada conquista, com muita gratidão, amor e carinho.

Aos meus amigos de infância, Geovanna, José Lucas, Éryka e Marília, obrigada por todo apoio, mesmo nos momentos em que a distância ou o tempo pareciam nos separar. Saber que vocês estavam lá por mim fez toda a diferença.

Ao meu companheiro de vida, Roger, que esteve ao meu lado durante os cinco anos desse curso. Fizemos algumas disciplinas juntos, e nos momentos mais difíceis, de maior tensão

e cansaço, sua leveza e presença foram fundamentais para que eu seguisse em frente.

Às amigas Vitória, Raquel e Joana, que estiveram comigo nesse último semestre tão intenso — uma segurando a mão da outra para que ninguém desistisse —, meu sincero agradecimento pela cumplicidade, força e amizade.

Às queridas Iara Amâncio e Vitória Freitas, obrigada por todo o companheirismo, apoio constante e amizade. Vocês estiveram presentes nos momentos bons e ruins, ofereceram ajuda sem que eu precisasse pedir e compartilharam comigo não apenas o peso das dificuldades, mas também a leveza das conquistas. Ter vocês ao meu lado tornou essa jornada não apenas possível, mas muito mais significativa, humana e acolhedora. Levo comigo a amizade de vocês como um dos maiores presentes que a universidade me deu.

Agradeço também ao meu orientador, Wendley, por toda orientação, paciência e dedicação ao longo do desenvolvimento deste trabalho. Sua experiência e apoio foram essenciais para que este TCC fosse possível.

À professora Jermana, pela disciplina Seminários de Computação II, que contribuiu de forma essencial para a estruturação e amadurecimento desta pesquisa. Suas orientações e provocações acadêmicas foram valiosas.

Sou imensamente grata aos grupos e espaços que fizeram parte da minha vivência universitária: PET, Nuclic, Centro Acadêmico, grupo de estudos para a Maratona de Programação, Atlética Imortal, Cursinho de Exatas da UFC, Labelas e a startup Urflex. Em cada um desses lugares, aprendi, cresci e conheci pessoas incríveis que me marcaram para sempre.

Ao meu cunhado Wilson, que sempre abriu caminhos e oportunidades com generosidade, e a todos os professores que tive a honra de conhecer, meu mais profundo respeito e gratidão.

Agradeço também a Bruna, Karine, Ana Lia, Marília e Jordânia por terem feito do AP204 um verdadeiro lar. Obrigada pelas noites de conversa no “restaurante”, pelas sessões de filmes improvisadas, pelas risadas, desabafos e apoio silencioso. Vocês fizeram parte da minha rotina, das minhas memórias mais queridas e me ajudaram a transformar um espaço físico em um espaço de afeto. Serei eternamente grata por isso.

A todos que, de alguma forma, contribuíram para a realização deste trabalho e para minha caminhada até aqui, o meu mais sincero muito obrigada. Essa conquista é nossa.

"O que nunca morre são as lembranças e os amores que criamos, construimos e firmamos ao longo dos anos."

(Jady Leonilya)

RESUMO

O uso de plataformas low-code tem se expandido no desenvolvimento de software devido à sua promessa de maior agilidade, redução de custos e menor dependência de equipes altamente especializadas. No entanto, à medida que essas soluções são escaladas, surgem desafios relacionados à manutenção, personalização, custos operacionais e dependência tecnológica. Este trabalho investigou as percepções de profissionais com experiência em plataformas como OutSystems, Mendix e Salesforce, buscando avaliar os impactos dessas ferramentas no longo prazo. A pesquisa utilizou uma abordagem quantitativa e exploratória, por meio da aplicação de um questionário estruturado a 47 profissionais atuantes na área. Os resultados indicam uma percepção predominantemente positiva em aspectos como produtividade e escalabilidade, mas também revelam preocupações crescentes com o custo total de propriedade e limitações técnicas em contextos mais complexos. O estudo contribui para uma compreensão mais realista dos benefícios e limitações das plataformas low-code, oferecendo subsídios para organizações que consideram adotar ou expandir o uso dessas soluções.

Palavras-chave: Low-code. Desenvolvimento de software. Escalabilidade. Custo total de propriedade. Produtividade. Manutenção. Percepções profissionais.

ABSTRACT

The use of low-code platforms has been expanding in software development due to their promise of greater agility, cost reduction, and lower dependence on highly specialized teams. However, as these solutions scale, challenges emerge related to maintenance, customization, operational costs, and technological dependency. This study investigated the perceptions of professionals with experience in platforms such as OutSystems, Mendix and Salesforce, aiming to assess the long-term impacts of these tools. The research followed a quantitative and exploratory approach, through the application of a structured questionnaire to 47 professionals in the field. The results indicate a predominantly positive perception regarding productivity and scalability, while also revealing growing concerns about total cost of ownership and technical limitations in more complex scenarios. The study contributes to a more realistic understanding of the benefits and limitations of low-code platforms, offering insights for organizations considering the adoption or expansion of such solutions.

Keywords: Low-code. Software development. Scalability. Total cost of ownership. Productivity. Maintenance. Professional perceptions.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicações
IA	Inteligência Artificial
TCO	Custo Total de Propriedade

SUMÁRIO

1 INTRODUÇÃO

No contexto atual de transformação digital, as empresas enfrentam a constante necessidade de adaptar suas operações e melhorar sua eficiência por meio do uso de tecnologias inovadoras. O desenvolvimento de *software* desempenha um papel fundamental nesse processo, sendo uma das áreas mais desafiadoras para muitas organizações, especialmente para aquelas de médio e pequeno porte. A busca por soluções que tornem esse processo mais rápido e econômico tem levado muitas empresas a adotar plataformas de desenvolvimento *low-code*. Surgidas no início dos anos 2010, essas ferramentas ganharam ainda mais popularidade a partir de 2016, quando a consultoria Gartner cunhou o termo Low-Code Application Platform, consolidando seu reconhecimento como uma categoria estratégica de tecnologia. Desde então, essas plataformas vêm sendo impulsionadas pela demanda por soluções ágeis e customizáveis, especialmente em empresas de menor porte (BOCK; FRANK, 2021).

As plataformas *low-code* são ferramentas de desenvolvimento de *software* que permitem criar aplicações de forma rápida e intuitiva, por meio de interfaces visuais e poucos códigos manuais. Elas são ideais para equipes com pouca experiência em programação, permitindo que até mesmo usuários não técnicos desenvolvam soluções personalizadas. Essas plataformas oferecem modelos pré-configurados, componentes reutilizáveis e integração com outros serviços, tornando o desenvolvimento mais eficiente. Entre as principais plataformas *low-code* estão o Microsoft Power Apps, OutSystems, Salesforce e Mendix, que oferecem diferentes recursos e funcionalidades para facilitar a criação de sistemas empresariais (OLTROGGE *et al.*, 2018).

Embora o uso de plataformas *low-code* ofereça inúmeras vantagens, como agilidade e redução de custos iniciais, elas impõem desafios significativos quanto à escalabilidade, manutenção e qualidade do *software* no longo prazo. As soluções desenvolvidas com essas ferramentas podem ser limitadas em termos de flexibilidade e personalização, o que compromete a capacidade de adaptação a novas necessidades empresariais. Além disso, a dependência das plataformas pode gerar custos elevados com licenças e atualizações, tornando a manutenção de sistemas mais difícil e cara à medida que a empresa cresce. Esses desafios são maiores quando a empresa precisa expandir suas soluções ou lidar com sistemas complexos, pois a plataforma pode não suportar adequadamente a alta demanda (ALAMIN *et al.*, 2021).

Estes desafios são comuns a diversos setores e organizações que adotam plataformas *low-code*, independentemente do ramo de atuação ou da plataforma específica utilizada. A dependência das soluções *low-code* pode levar a custos crescentes e limitações técnicas à medida

que as necessidades de negócios evoluem, impactando diretamente a manutenção, escalabilidade e customização das aplicações.

Aqui mostramos, por meio de um estudo quantitativo com profissionais usuários de diferentes plataformas *low-code* (como OutSystems, Salesforce e Mendix), as percepções sobre os benefícios e desafios dessas tecnologias, especialmente relacionados a custos operacionais, escalabilidade, manutenção e qualidade das soluções.

1.1 Objetivo Geral

Este trabalho objetiva analisar as percepções de profissionais sobre o uso de plataformas *low-code* no desenvolvimento de software, focando em aspectos como custos operacionais, escalabilidade, manutenção, personalização e qualidade das soluções ao longo do tempo. Para isso, é realizada uma pesquisa com profissionais experientes que utilizam diferentes plataformas *low-code*, buscando compreender os impactos e desafios do uso dessas ferramentas em variados contextos organizacionais.

1.2 Objetivos Específicos

- Compreender como a experiência prévia com plataformas *low-code* influencia a percepção dos usuários em relação a custos e atualizações.
- Identificar tendências de diferenciação entre plataformas específicas.
- Explorar relações entre variáveis de percepção, como produtividade, inovação e adequação futura.
- Levantar possíveis limitações do uso de *low-code* percebidas por usuários experientes, mesmo em contextos de alta aceitação.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos centrais sobre plataformas *low-code*, sua evolução e impacto no desenvolvimento de *software*. São exploradas suas vantagens iniciais, como agilidade e facilidade de uso, assim como limitações em manutenção, escalabilidade e personalização.

2.1 Introdução às Plataformas *Low-Code*

As plataformas *low-code* permitem a criação de sistemas com pouca ou nenhuma codificação, por meio de interfaces visuais e componentes reutilizáveis. Além de reduzirem o tempo de desenvolvimento, ampliam o acesso à criação de *software* por equipes com menos experiência técnica. A adoção dessas ferramentas cresceu significativamente após 2016, com grandes empresas como Microsoft, Google e Siemens investindo em suas soluções.

2.1.1 Definição, Conceitos Fundamentais e Contexto no Mercado de TI

As plataformas *low-code* são ferramentas para desenvolvimento de *software* que permitem a criação de aplicativos e sistemas com pouca ou nenhuma codificação. Elas utilizam interfaces gráficas intuitivas e recursos de arrastar e soltar componentes para compor a interface do usuário e a lógica de aplicação. Isso torna possível que profissionais sem formação técnica em programação, como analistas de negócios ou gestores, também participem ativamente da construção de soluções tecnológicas (BOCK; FRANK, 2021).

As principais vantagens dessas plataformas são: permitir a automação de tarefas repetitivas, sem a necessidade de escrever códigos complexos; a facilidade de uso, já que não é necessário muito conhecimento técnico; e a agilidade no desenvolvimento, já que não é necessário fazer muito código manual. O propósito das plataformas *low-code* é simplificar o processo de criação de soluções e permitir que mais pessoas, além dos programadores, possam desenvolver *softwares* (OLTROGGE *et al.*, 2018).

As plataformas *low-code* surgiram como uma forma de reduzir o tempo de desenvolvimento e os custos com programação. Desde seu surgimento, no início dos anos 2010, essas ferramentas evoluíram significativamente, principalmente a partir de 2016, quando o termo *Low-Code Application Platform* foi cunhado pela consultoria Gartner, marcando o reconhecimento formal dessas ferramentas como uma categoria estratégica de tecnologia para o desenvolvimento

de aplicações. A partir desse momento, grandes empresas como Microsoft, Google e Siemens passaram a investir fortemente nessas plataformas, impulsionando sua adoção em larga escala no contexto da transformação digital. Na atualidade, estão sendo adotadas principalmente em empresas que buscam aumentar sua agilidade na criação de soluções sem a complexidade dos métodos tradicionais de desenvolvimento de *software* (SUFI, 2023).

A ascensão das plataformas *low-code* é alimentada pela crescente demanda de soluções rápidas e pela evolução das tecnologias que permitem interfaces mais intuitivas e a automação de processos de desenvolvimento. Essas plataformas estão em crescimento, especialmente entre empresas de médio e pequeno porte que precisam de soluções mais rápidas e flexíveis. Elas fazem parte de um movimento que busca democratizar a criação de *software*, permitindo que equipes de tecnologia da informação e até mesmo usuários, donos de negócios, desenvolvam suas próprias soluções (SAHAY *et al.*, 2020).

2.1.2 Principais Plataformas Low-Code

Dentre as ferramentas mais utilizadas destacam-se Microsoft Power Apps, OutSystems, Mendix e Salesforce. Cada uma possui recursos específicos que favorecem a construção rápida de aplicações integradas a serviços existentes. As interfaces visuais e os conectores prontos possibilitam que mesmo usuários não programadores criem soluções personalizadas.

2.1.2.1 Microsoft Power Apps

O Power Apps é uma plataforma de dados que oferece um ambiente de desenvolvimento rápido de aplicativos personalizados para atender às necessidades de negócios. Com ele, é possível criar aplicativos de negócios personalizados rapidamente, conectando-os aos dados armazenados na plataforma de dados (Microsoft Dataverse) ou a outras fontes de dados *online* e locais. Ele oferece recursos de arrastar e soltar e disponibiliza uma coleção de *templates* que permite o reaproveitamento de funcionalidades já desenvolvidas. Além disso, ele integra com muitos serviços do ecossistema Microsoft, como Excel e banco de dados Azure, e fornece conectores para sistemas legados (Microsoft, 2024).

2.1.2.2 OutSystems

A plataforma OutSystems é uma solução *low-code* com tecnologia de Inteligência Artificial (IA) que permite criar rapidamente aplicativos críticos, ajudando a gerenciar e desenvolver *software* de maneira mais eficiente. Essa plataforma permite a criação de *softwares web* e *mobile* que podem ser executados na nuvem ou em infraestruturas locais. Entre seus principais recursos, destaca-se a publicação de aplicativos com um único clique, por meio de uma *URL*. O OutSystems conta com dois componentes principais: um *Studio* para conexão com bancos de dados via *.NET* ou Java, e um *Service Studio* para definir o comportamento da aplicação em desenvolvimento (OutSystems, 2024).

2.1.2.3 Mendix

A plataforma Mendix permite o desenvolvimento rápido de aplicativos com pouco código, sendo a maior parte da programação realizada de forma visual. Ela foi projetada para acelerar a entrega de aplicativos empresariais durante todo o ciclo de vida de desenvolvimento, desde a concepção até a implantação e operação. Ela suporta práticas ágeis e *DevOps*, oferecendo uma plataforma com ferramentas tanto *low-code*, que exigem pouca codificação e são voltadas a desenvolvedores, quanto *no-code*, que dispensam codificação e permitem que usuários sem conhecimento técnico criem aplicações por meio de interfaces visuais. A plataforma permite a criação de aplicativos com conectores preexistentes. Além disso, a Mendix é compatível com Docker e Kubernetes e conta com *templates* e uma galeria de soluções que auxiliam na rapidez do desenvolvimento (Mendix Brasil, 2024).

2.1.2.4 Salesforce

A plataforma Salesforce App Cloud permite aos desenvolvedores criar e publicar aplicativos baseados em nuvem que são seguros e escaláveis. A plataforma oferece ferramentas e operações prontas, integrando-as a serviços externos. Alguns dos recursos mais notáveis incluem o extenso *marketplace* AppExchange, que oferece aplicativos e componentes prontos, objetos e elementos reutilizáveis, o construtor de processos com interface de arrastar e soltar e quadros *kanban* integrados, facilitando a construção de soluções empresariais (Salesforce, 2024).

2.2 Impactos das Plataformas Low-Code na Construção de Software

O desenvolvimento *low-code* acelera a entrega de aplicações, mas pode comprometer o controle sobre o código e a flexibilidade do sistema. Enquanto o modelo tradicional oferece personalização total, o *low-code* é limitado às possibilidades da plataforma, exigindo avaliação cuidadosa conforme o contexto.

2.2.1 Desenvolvimento Tradicional vs. Desenvolvimento Low-Code

O desenvolvimento tradicional, que envolve codificação manual de cada componente do sistema, permite total controle sobre os sistemas e é altamente personalizável. No entanto, ele exige um conhecimento técnico mais avançado e pode ser mais demorada por necessitar de vários processos como: modelagem de dados para a criação do banco de dados; lógica de negócios, Interface de Programação de Aplicações (API), autenticação e integração com o banco de dados no *backend*; interface do usuário, integração com o *backend* via API, responsividade, no *frontend*; além dos testes e colocar o sistema em produção (*deploy*) (BOCK; FRANK, 2021).

Por outro lado, o desenvolvimento *low-code* oferece uma abordagem mais simplificada, fornecendo modelos pré-configurados e componentes reutilizáveis que facilitam a criação de aplicativos, permitindo criar soluções de *software* rapidamente. Nele o Banco de dados é criado e gerido visualmente; no *backend* tem a automação dos processos utilizando interface gráfica; no *frontend* a interface do usuário é construída com componentes visuais; e a integração e *deploy* são feitos de maneira muito mais rápida. Mas o *low-code* têm algumas limitações em relação à personalização, escalabilidade e ao controle do código-fonte (SAHAY *et al.*, 2020).

A Tabela 1 resume as principais diferenças entre o desenvolvimento tradicional e o desenvolvimento *low-code*.

Tabela 1 – Principais diferenças entre o Desenvolvimento Tradicional e o Desenvolvimento Low-Code.

Aspecto	**Desenvolvimento Tradicional**	**Desenvolvimento *Low-Code***
Controle e Personalização	Total controle sobre código e personalização.	Menos controle e personalização, com limitações no código gerado.
Complexidade e Tempo de Desenvolvimento	Requer codificação manual, é mais demorado e envolve várias etapas.	Acelera o desenvolvimento com modelos e componentes reutilizáveis.
Flexibilidade	Altamente flexível, com total liberdade para ajustar e customizar.	Modularidade limitada pela plataforma, dificultando customizações complexas.
Escalabilidade	Escalabilidade dependendo da implementação e da arquitetura escolhida.	Pode enfrentar desafios de escalabilidade para grandes volumes de dados ou usuários.
Controle sobre Código-Fonte	Total controle sobre o código-fonte.	Falta de controle sobre o código gerado, dificultando ajustes mais específicos.
Manutenção e Adaptação	Manutenção mais flexível, mas pode ser mais complexa e demorada.	Dependência da plataforma, que pode gerar desafios com atualizações e descontinuação.
Dependência da Plataforma	Sem dependência de uma plataforma específica.	Dependência de atualizações, segurança e suporte da plataforma.

2.2.2 Exemplo Comparativo: Desenvolvimento Tradicional vs. Low-Code

Para exemplificar as diferenças entre o desenvolvimento tradicional e o desenvolvimento low-code, foram criadas duas versões da mesma tela de login: uma utilizando Angular no frontend e Laravel no backend, e outra utilizando a plataforma low-code OutSystems.

2.2.2.1 Desenvolvimento Tradicional com Angular e Laravel

Diferente da abordagem low-code, onde os componentes são montados visualmente, nessa versão todo o sistema foi desenvolvido manualmente, exigindo a implementação individual de cada parte da aplicação.

O código-fonte 1 mostra o código HTML responsável por estruturar visualmente a tela. Nela são definidos os campos de entrada para email e senha, o botão de login, o checkbox *Remember me* e o link de recuperação de senha. Esses elementos são ligados ao comportamento

da aplicação através de *data binding* com variáveis do componente.

Código-fonte 1 – HTML da tela de login em Angular

```
<div class="login-container">
  <div class="login-card">
    <div class="logo">AI</div>
    <h2>Application Title</h2>
    <form (ngSubmit)="onSubmit()" #loginForm="ngForm"
       novalidate>
      <input type="email" [(ngModel)]="email" name="email"
         required email />
      <input type="password" [(ngModel)]="password" name="
         password" required />
      <div class="options">
        <label><input type="checkbox" [(ngModel)]="
           rememberMe" name="rememberMe" /> Remember me</
           label>
        <a href="#">Forgot password?</a>
      </div>
      <button type="submit" [disabled]="loginForm.invalid">
         Login</button>
    </form>
  </div>
</div>
```

Já o código-fonte 2 apresenta o código CSS que faz a estilização da tela. Ele define o posicionamento centralizado da interface, as cores, sombras, espaçamento e aparência dos elementos visuais.

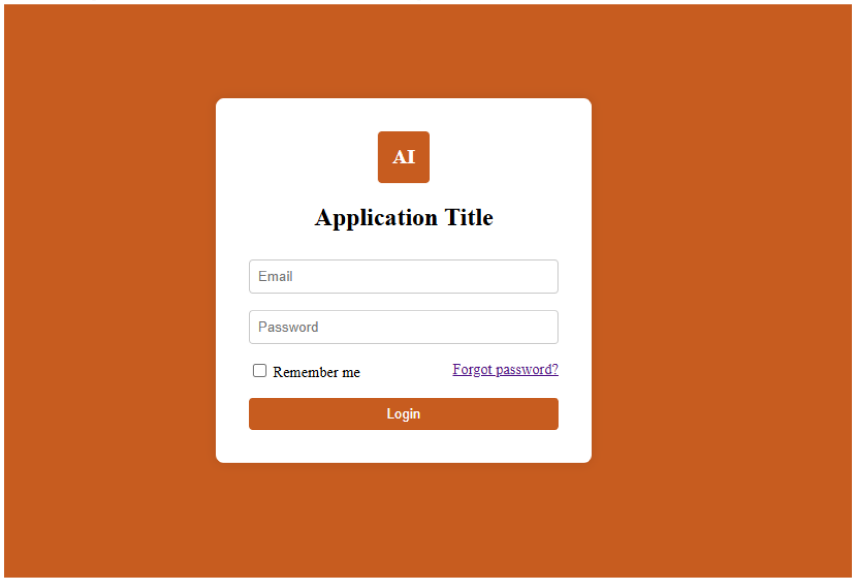
Código-fonte 2 – CSS da tela de login em Angular

```
.login-container {
  background: #c75c1f;
  height: 100vh;
```

```
  display: flex;
  align-items: center;
  justify-content: center;
}
.login-card {
  background: #fff;
  padding: 2rem;
  border-radius: 8px;
  width: 300px;
  text-align: center;
}
.logo {
  background: #c75c1f;
  color: white;
  width: 50px;
  height: 50px;
  margin: 0 auto 1rem;
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight: bold;
  font-size: 20px;
  border-radius: 4px;
}
```

A Figura 1 apresenta o resultado visual da aplicação gerado a partir dos códigos acima.

Figura 1 – Tela de login desenvolvida com Angular.



Fonte: Elaborado pela autora.

A lógica do formulário é implementada em TypeScript, conforme demonstrado no código-fonte 3. Nela, o método `onSubmit()` coleta os dados do formulário e realiza uma requisição HTTP para uma API Laravel.

Código-fonte 3 – TypeScript do componente de login

```
export class LoginComponent {
  email: string = '';
  password: string = '';
  rememberMe: boolean = false;

  constructor(private http: HttpClient) {}

  onSubmit() {
    const payload = {
      email: this.email,
      password: this.password,
      remember: this.rememberMe
    };

    this.http.post('http://localhost:8000/api/login',
```

```
        payload).subscribe({
      next: res => console.log('Login bem-sucedido', res),
      error: err => console.error('Erro ao fazer login',
        err)
    });
  }
}
```

No backend, o código-fonte 4 mostra o código da API de autenticação em Laravel. Ela valida os dados recebidos, busca o usuário no banco de dados e compara a senha informada utilizando hash.

Código-fonte 4 – Trecho da API de login em Laravel

```
public function login(Request $request)
{
    $request->validate([
        'email' => 'required|email',
        'password' => 'required|string',
    ]);

    $user = DB::table('users')->where('email', $request->
       email)->first();

    if ($user && Hash::check($request->password, $user->
       password)) {
        return response()->json([
            'message' => 'Login bem-sucedido',
            'user' => $user
        ]);
    }

    return response()->json(['message' => 'Credenciais
```

```
        inv lidas'], 401);
}
```

Essa abordagem tradicional, embora mais trabalhosa e técnica, permite maior controle e personalização. Em contrapartida, o desenvolvimento low-code reduz significativamente o volume de código, como demonstrado a seguir.

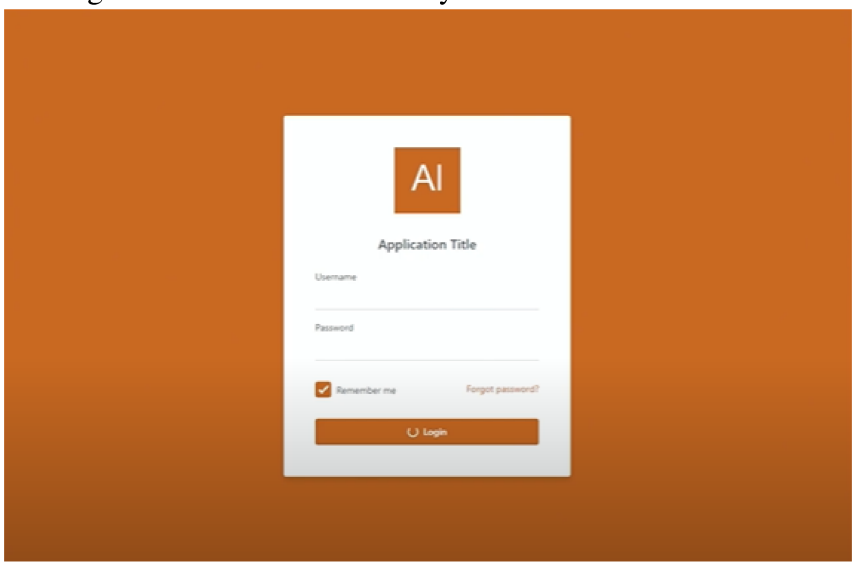
2.2.2.2 OutSystems

Na plataforma OutSystems, a mesma tela de login foi construída por meio de um editor visual, sem a necessidade de codificação manual. A interface é composta por componentes prontos — como campos de entrada e botões — que podem ser arrastados e organizados visualmente na página.

As propriedades de cada componente, como rótulo, obrigatoriedade, aparência e vinculação com variáveis de dados, são definidas diretamente no editor. A identidade visual pode ser ajustada por meio de temas globais ou personalizações CSS limitadas.

A Figura 2 mostra a interface criada com essa abordagem.

Figura 2 – Tela de login desenvolvida com OutSystems.

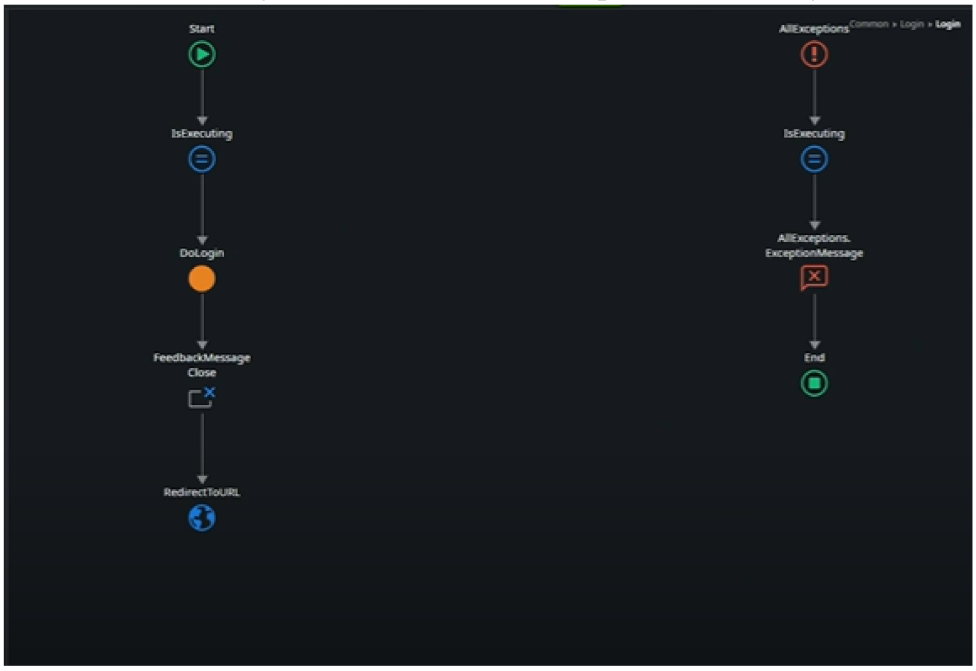


Fonte: Elaborado pela autora.

A lógica da aplicação também é criada visualmente. A Figura 3 apresenta o fluxo de autenticação, implementado por meio de blocos conectados que representam ações, verificações e redirecionamentos. A autenticação é feita com o bloco `DoLogin`, que encapsula a lógica de

validação e criação de sessão.

Figura 3 – Fluxo de autenticação criado visualmente na plataforma OutSystems.



Fonte: Elaborado pela autora.

A integração com banco de dados também é automatizada: o desenvolvedor define entidades e atributos por meio de uma interface visual, e o OutSystems gera o esquema do banco e os métodos de acesso.

Esse modelo acelera o desenvolvimento e permite que interfaces funcionais sejam criadas rapidamente, mesmo por equipes com menos experiência técnica. No entanto, pode haver limitações na personalização ou em funcionalidades mais específicas, que exigiriam extensões em código tradicional.

Essa comparação prática evidencia como o desenvolvimento low-code é vantajoso em termos de agilidade e produtividade, especialmente em projetos com prazos curtos, enquanto o desenvolvimento tradicional é preferível em cenários que exigem flexibilidade total e controle aprofundado.

2.2.3 Modularidade e Flexibilidade no Desenvolvimento

As plataformas *low-code* são projetadas para serem modulares no desenvolvimento de soluções, devido à sua abordagem visual, onde os desenvolvedores podem “arrastar e soltar” componentes. Isso permite a reutilização de módulos prontos, acelerando o processo de desen-

volvimento. No entanto, a flexibilidade é um ponto crítico: a modularidade nas plataformas *low-code* é limitada pela própria plataforma, e há uma falta de controle sobre o código gerado. Quando as necessidades de personalização são mais complexas, a falta de flexibilidade pode se tornar um obstáculo, especialmente se a aplicação precisar de funcionalidades que não são facilmente atendidas pela plataforma (ALSAADI *et al.*, 2021).

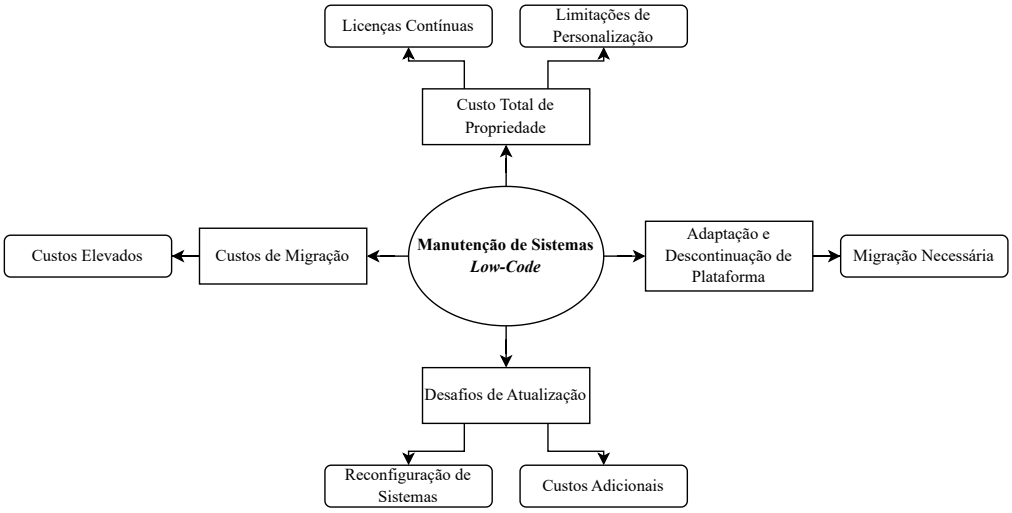
2.2.4 Limitações Técnicas e Dependência da Plataforma

Embora as plataformas *low-code* ofereçam uma solução rápida e eficaz para desenvolvedores com habilidades limitadas, elas impõem algumas limitações técnicas. Por exemplo, a customização do código gerado nem sempre é possível, ou é muito difícil, o que pode limitar a adaptação da aplicação às necessidades específicas de uma empresa. Além disso, as empresas ficam dependentes das atualizações, segurança e suporte da plataforma, o que pode ser um problema se a plataforma for descontinuada ou se os preços aumentarem. A escalabilidade também pode ser um desafio, pois a plataforma pode não suportar adequadamente o crescimento de uma aplicação de forma tão eficiente quanto um código tradicional. Além disso, a dependência de uma plataforma específica pode gerar dificuldades em termos de necessidade de migração para uma solução personalizada quando os requisitos de negócios se tornam mais complexos ou específicos (KÄSS *et al.*, 2023).

2.3 Manutenção de Sistemas Desenvolvidos com Low-Code

Embora o desenvolvimento inicial com plataformas *low-code* apresente menor custo e maior agilidade, a manutenção a longo prazo pode ser onerosa. Além da dependência de licenças e suporte do fornecedor, mudanças na plataforma ou descontinuações podem gerar altos custos de migração e adaptação. A Figura 4 ilustra os principais desafios e custos da manutenção de sistemas *low-code*.

Figura 4 – Desafios e Custos Associados à Manutenção de Sistemas Low-Code.



Fonte: Elaborado pela autora.

2.3.1 Custo Total de Propriedade TCO

O TCO de uma solução *low-code* não se limita ao investimento inicial em desenvolvimento. Apesar da agilidade e facilidade de uso proporcionadas pela plataforma, o custo total tende a aumentar ao longo do tempo devido a diversos fatores operacionais e estratégicos. Entre os principais estão a exigência de pagamento contínuo de licenças, a dependência do fornecedor e as limitações na escalabilidade e personalização dos sistemas criados.

Além disso, muitas plataformas possuem restrições na integração com sistemas legados, exigindo soluções alternativas que elevam os custos de manutenção. Quando a aplicação precisa evoluir para atender a requisitos mais complexos, customizações fora do escopo nativo da plataforma podem demandar extensões em código tradicional — o que reduz a vantagem inicial de simplicidade e aumenta o custo de operação.

Em alguns casos, as organizações optam por migrar para soluções com maior flexibilidade, o que implica custos adicionais com reimplementação, treinamento e adaptação de processos. Por isso, ao adotar plataformas *low-code*, é fundamental que as empresas considerem não apenas a velocidade de entrega, mas também os impactos financeiros cumulativos ao longo de todo o ciclo de vida do sistema (SAHAY *et al.*, 2020).

2.3.2 Desafios de Atualização e Sustentação

Um dos principais desafios no uso de plataformas *low-code* é o custo e a complexidade associados à manutenção dos sistemas. Como muitos sistemas baseados em *low-code* dependem de ferramentas e abstrações fornecidas pela plataforma, quando ocorre uma atualização ou modificação importante na plataforma, o sistema pode enfrentar dificuldades de adaptação ou até ficar obsoleto. As atualizações das plataformas podem exigir reconfiguração dos sistemas existentes, o que implica em custos adicionais de manutenção. Além disso, se uma plataforma *low-code* for descontinuada, como aconteceu com o Google App Maker, cria grandes desafios para os desenvolvedores que precisam migrar seus sistemas para outra plataforma ou até mesmo para soluções totalmente personalizadas (ALAMIN *et al.*, 2021).

2.3.3 Adaptação a Novas Necessidades Empresariais

À medida que as necessidades empresariais mudam, adaptar sistemas desenvolvidos em plataformas *low-code* pode se tornar problemático. Embora essas plataformas ofereçam agilidade no desenvolvimento, elas podem apresentar limitações em termos de funcionalidades e integração com outros sistemas à medida que a complexidade do negócio aumenta. Isso frequentemente leva as empresas a migrar para soluções mais personalizadas ou tradicionais quando a demanda por customizações se torna muito alta, a fim de atender a requisitos empresariais específicos (BOCK; FRANK, 2021).

2.4 Escalabilidade e Sustentabilidade em Soluções Low-Code

Apesar de eficazes em estágios iniciais, as plataformas *low-code* podem enfrentar limitações quando a demanda por funcionalidades cresce. Problemas de desempenho e dificuldades de integração são comuns em cenários complexos. Em muitos casos, a migração para soluções tradicionais torna-se inevitável para manter a qualidade do sistema.

2.4.1 Capacidade de Escalar em Empresas em Crescimento

As plataformas *low-code* podem ser vantajosas para empresas menores ou para desenvolvimento rápido de soluções, especialmente em seus estágios iniciais ou que precisam desenvolver aplicações rapidamente e sem um grande investimento, mas podem enfrentar desa-

fios de escalabilidade à medida que a empresa cresce, especialmente quando se trata de grandes volumes de dados ou de um número muito grande de usuários simultâneos. Isso ocorre porque muitas dessas plataformas são projetadas para criar aplicações de menor porte ou com funcionalidades simples. Quando uma empresa precisa comportar mais usuários, maior complexidade ou integrações com outros sistemas, as limitações da plataforma podem se tornar evidentes. Essas limitações de escalabilidade podem levar as empresas a repensar a utilização de plataformas *low-code*, especialmente quando a demanda por soluções mais robustas cresce (SAHAY *et al.*, 2020).

2.4.2 Impacto do Aumento de Usuários e Complexidade Funcional

Com o aumento do número de usuários e a complexidade funcional de uma aplicação, as soluções criadas com plataformas *low-code* podem começar a enfrentar problemas de desempenho. À medida que a complexidade aumenta, as limitações das plataformas se tornam mais evidentes, especialmente se a plataforma não for bem dimensionada ou configurada corretamente. A necessidade de manter a performance diante de um aumento no número de usuários ou no volume de dados pode exigir ajustes que não são possíveis dentro das limitações da plataforma. À medida que os aplicativos crescem e novos recursos são adicionados, pode se tornar difícil gerenciar a complexidade sem comprometer a qualidade do sistema (ALAMIN *et al.*, 2021).

2.4.3 Necessidade de Migração para Soluções Personalizadas

Quando as empresas crescem e suas necessidades se tornam mais específicas, o custo e as limitações das plataformas *low-code* podem levar a uma migração para soluções desenvolvidas sob medida ou para plataformas mais flexíveis. Essa transição pode ser desafiadora e ter um alto custo, especialmente se a arquitetura da aplicação não for facilmente migrável. Portanto, muitas organizações podem precisar considerar a migração para soluções mais flexíveis e totalmente personalizáveis à medida que seus requisitos se tornam mais sofisticados. Embora as plataformas *low-code* ofereçam uma maneira rápida de criar e escalar aplicativos, elas podem não ser adequadas para todos os tipos de soluções de longo prazo, especialmente em empresas que exigem alta personalização e controle total sobre seu código e arquitetura (BOCK; FRANK, 2021).

3 TRABALHOS RELACIONADOS

Para a seleção dos trabalhos analisados nesta seção, foram considerados critérios como relevância direta para o tema da pesquisa, especialmente no que diz respeito às plataformas low-code, seus impactos operacionais e desafios de escalabilidade. Também foi adotado como critério o ano de publicação, priorizando estudos mais recentes (de 2020 em diante) para garantir atualidade nas discussões. Além disso, levou-se em conta a metodologia empregada nos artigos, com preferência por pesquisas que adotaram abordagens empíricas, estudos de caso ou revisões sistemáticas, a fim de garantir uma base teórica sólida e alinhada aos objetivos deste trabalho.

O estudo Alamin *et al.* (2021) explorou as discussões de desenvolvedores sobre as plataformas de desenvolvimento *low-code* no Stack Overflow. A pesquisa, que analisou aproximadamente 5.000 postagens, utilizou a técnica de *topic modeling* para identificar 13 tópicos principais, agrupados em categorias como personalização, adoção de plataforma, gerenciamento de banco de dados e integração de terceiros. Este estudo ofereceu uma visão detalhada sobre os desafios comuns que os desenvolvedores enfrentam ao utilizar plataformas *low-code*. Contudo, não explorou diretamente os impactos dessas plataformas em termos de custo, manutenção e escalabilidade em ambientes corporativos, aspectos que são foco da presente pesquisa.

O artigo Sufi (2023) fez uma revisão prática do uso de algoritmos em plataformas *low-code/no-code*, abordando 23 estudos recentes e destacando plataformas populares como SetXRM, vf-OS, Aure-BPM, CRISP-DM e Microsoft Power Platform. O estudo foi útil para entender a aplicação dessas plataformas em contextos de pesquisa e como elas facilitaram a implementação de soluções algorítmicas sem a necessidade de codificação extensiva. Este estudo focou predominantemente em plataformas voltadas para pesquisas, o que limita a aplicabilidade de seus achados. Apesar de fornecer uma visão ampla do uso dessas tecnologias, o estudo teve um enfoque maior em aplicações acadêmicas e não abordou questões operacionais e de escalabilidade no contexto empresarial, que são centrais para o presente trabalho.

O estudo Käss *et al.* (2023) examinou as percepções dos profissionais sobre a adoção das Plataformas de Desenvolvimento *low-code* por organizações. A pesquisa focou nos fatores que impulsionam ou inibiram a adoção dessas plataformas, proporcionando *insights* sobre a perspectiva prática de quem utiliza essas ferramentas no dia a dia. Embora tenha fornecido insights valiosos sobre a experiência prática dos usuários, o estudo não aprofundou os desafios específicos relacionados à manutenção, custos de propriedade e escalabilidade em ambientes de desenvolvimento com alta demanda, temas explorados nesta pesquisa.

O artigo Sahay *et al.* (2020) propôs um *framework* comparativo para ajudar na avaliação e compreensão das plataformas de desenvolvimento *low-code*. Ao analisar oito plataformas representativas, o estudo identificou um conjunto de características-chave para comparar as funcionalidades de cada uma. A pesquisa foi útil para aqueles que buscaram avaliar as capacidades de diferentes plataformas de *low-code* antes de tomar decisões estratégicas. Embora essa comparação seja útil para selecionar plataformas, o estudo não discutiu desafios práticos e operacionais que surgem com o uso contínuo dessas ferramentas, como a escalabilidade e o custo total de propriedade, pontos centrais no escopo deste trabalho.

Em conjunto, os autores convergem em pontos centrais como a agilidade no desenvolvimento inicial, custos reduzidos no curto prazo e facilidade de uso, aspectos frequentemente destacados como vantagens das plataformas *low-code*. No entanto, também há consenso quanto às limitações de escalabilidade, dependência de fornecedores, e desafios de manutenção e migração, especialmente em contextos empresariais que demandam alto grau de personalização e integração com sistemas legados. Embora esses estudos ofereçam contribuições relevantes, eles não aprofundam a análise dos impactos operacionais e das percepções dos profissionais em múltiplas plataformas *low-code*, como OutSystems, Mendix e Salesforce, especialmente sob a perspectiva do uso a longo prazo, lacuna que este trabalho busca preencher por meio de uma pesquisa quantitativa com profissionais atuantes nessas plataformas.

Diante dessa análise dos principais estudos relacionados, percebe-se que cada um deles contribui com diferentes perspectivas sobre o uso de plataformas *low-code*, seja no contexto técnico, acadêmico ou prático. No entanto, ainda há uma lacuna quanto à compreensão integrada dos impactos operacionais a longo prazo, especialmente sob a ótica de profissionais que atuam diretamente com essas plataformas no ambiente corporativo. Para facilitar a visualização das características e limitações de cada trabalho analisado, a Tabela 2 apresenta um comparativo resumido entre os estudos discutidos nesta seção.

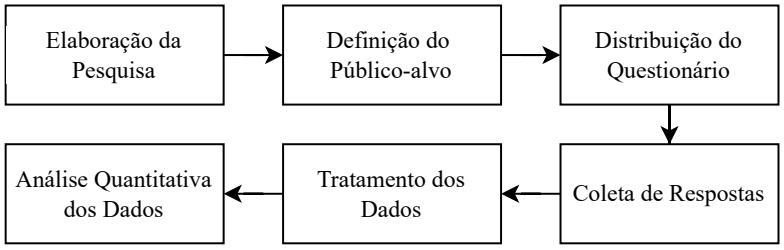
Tabela 2 – Comparação entre os principais trabalhos relacionados.

Autor(es)	**Foco Principal**	**Limitações**	**Contribuições**
Alamin et al. (2021)	Discussões técnicas sobre plataformas *low-code* no Stack Overflow	Não aborda impacto operacional em empresas	Identificação de desafios comuns enfrentados por desenvolvedores
Sufi (2023)	Uso de algoritmos em plataformas *low-code/no-code* em pesquisas acadêmicas	Enfoque acadêmico, sem análise empresarial	Revisão ampla de aplicações em contextos de pesquisa
Käss et al. (2023)	Fatores que influenciam a adoção de plataformas *low-code* por profissionais	Não foca em custos, escalabilidade ou manutenção	Percepções práticas sobre adoção e uso no mercado
Sahay et al. (2020)	Comparação entre funcionalidades de plataformas *low-code*	Não aborda uso contínuo, TCO ou manutenção	Proposta de framework comparativo para seleção de plataformas
Este trabalho	Percepções de profissionais sobre uso, manutenção, custo e escalabilidade	Amostra limitada e uso de abordagem quantitativa apenas	Aborda impacto de longo prazo com foco prático em múltiplas plataformas

4 METODOLOGIA

Este capítulo descreve os procedimentos adotados na realização da pesquisa sobre o impacto a longo prazo do uso de plataformas *low-code* no desenvolvimento, manutenção e escalabilidade de soluções de *software*. A pesquisa foi estruturada com abordagem quantitativa e exploratória, por meio da aplicação de um questionário online, com o objetivo de coletar percepções de profissionais atuantes com plataformas como OutSystems, Mendix e Salesforce. A amostragem foi por conveniência, com os participantes sendo recrutados via rede social LinkedIn. No total, foram obtidas 47 respostas válidas As etapas da pesquisa estão descritas no fluxograma da Figura 5, permitindo uma visão geral do processo metodológico. Os detalhes de cada fase são apresentados nas subseções seguintes.

Figura 5 – Fluxograma das etapas da pesquisa



Fonte: Elaborado pela autora.

4.1 Elaboração da Pesquisa

A primeira etapa consistiu na construção do questionário online, desenvolvido com a ferramenta Google Forms[1]. O questionário foi composto por três perguntas de identificação: cargo atual, plataforma *low-code* utilizada e tempo de experiência com a plataforma. Em seguida, foram elaboradas 18 afirmações distribuídas em temas como custo, manutenção, desempenho, escalabilidade, produtividade, curva de aprendizado e perspectivas futuras. Todas as afirmações foram avaliadas por meio de uma escala Likert de 5 pontos (1 – Discordo totalmente, 5 – Concordo totalmente).

4.2 Definição do Público-alvo

O público-alvo da pesquisa foi composto por profissionais que atuam diretamente com plataformas *low-code*, especificamente: OutSystems, Mendix e Salesforce. A seleção

[1] Disponível em: <https://forms.gle/SAcNjNRayH2mov3r8>. Acesso em: 02 jul. 2025.

priorizou pessoas com experiência prática e envolvimento direto no desenvolvimento de soluções com essas ferramentas, buscando garantir a qualidade das respostas obtidas.

4.3 Distribuição do Questionário

A distribuição do questionário foi realizada de forma online, utilizando a rede social LinkedIn como principal canal de divulgação. Foram feitos envios diretos para profissionais identificados com perfis compatíveis com os critérios da pesquisa. A abordagem utilizada foi personalizada, apresentando os objetivos da pesquisa, destacando sua finalidade acadêmica e garantindo a confidencialidade das respostas. A coleta foi realizada ao longo de duas semanas consecutivas.

4.4 Coleta de Respostas

Durante esse período, as respostas foram coletadas automaticamente por meio do Google Forms. O progresso foi monitorado diariamente, garantindo que os dados estivessem sendo armazenados corretamente. Ao todo, o questionário foi enviado para aproximadamente 100 profissionais da área. No final do prazo, os dados foram exportados no formato CSV para posterior tratamento e análise. No total, foram obtidas 50 respostas, das quais 47 foram consideradas válidas após a exclusão de registros nulos.

4.5 Tratamento dos Dados

Os dados brutos foram organizados em uma planilha e, em seguida, importados para um ambiente de análise utilizando a linguagem Python. Nessa etapa, foram realizadas limpezas básicas, incluindo a exclusão de respostas incompletas, a padronização dos nomes das plataformas (com a remoção de espaços em branco e a conversão para letras minúsculas) e a transformação do tempo de experiência em uma variável numérica ordinal. Essas ações permitiram a condução de análises estatísticas mais consistentes.

4.6 Análise Quantitativa dos Dados

A análise estatística foi conduzida com o uso das bibliotecas `pandas`, `matplotlib`, `seaborn` e `scipy`, no ambiente Google Colab. Foram calculadas frequências absolutas e

relativas, medidas de tendência central (média e mediana), além de representações visuais como histogramas para as 18 afirmações avaliadas.

Adicionalmente, foram geradas comparações entre as médias de respostas por plataforma e por tempo de experiência, possibilitando identificar variações de percepção entre grupos. A relação entre tempo de experiência e percepções foi analisada por meio do coeficiente de correlação de Spearman, apropriado para variáveis ordinais.

Para verificar se havia diferenças estatisticamente significativas nas respostas entre diferentes plataformas, aplicou-se o teste de ANOVA (Análise de Variância). Essas comparações foram complementadas com visualizações por meio de boxplots, facilitando a identificação de dispersões e tendências nas respostas por grupo.

Por fim, foi elaborada uma matriz de correlação (heatmap), utilizando o coeficiente de Spearman entre as 18 variáveis de percepção. Essa visualização permitiu identificar associações e possíveis padrões entre os diferentes aspectos avaliados sobre o uso de plataformas *low-code*.

5 RESULTADOS

Neste capítulo, são apresentados os principais achados obtidos a partir da análise dos dados coletados, organizados conforme as etapas descritas na metodologia.

5.1 Caracterização da Amostra

A amostra final foi composta por 47 participantes, após a exclusão de registros com valores nulos. As plataformas *low-code* mencionadas foram Salesforce, Outsystems e Mendix, sendo Salesforce a mais frequente.

Quanto ao tempo de experiência com as plataformas, a maioria dos participantes, indicou possuir mais de dois anos de uso, sugerindo um público com vivência prática significativa nas ferramentas avaliadas. As figuras a seguir mostram a distribuição das plataformas utilizadas (Figura 6) e a distribuição por tempo de experiência (Figura 7).

Figura 6 – Distribuição das plataformas utilizadas

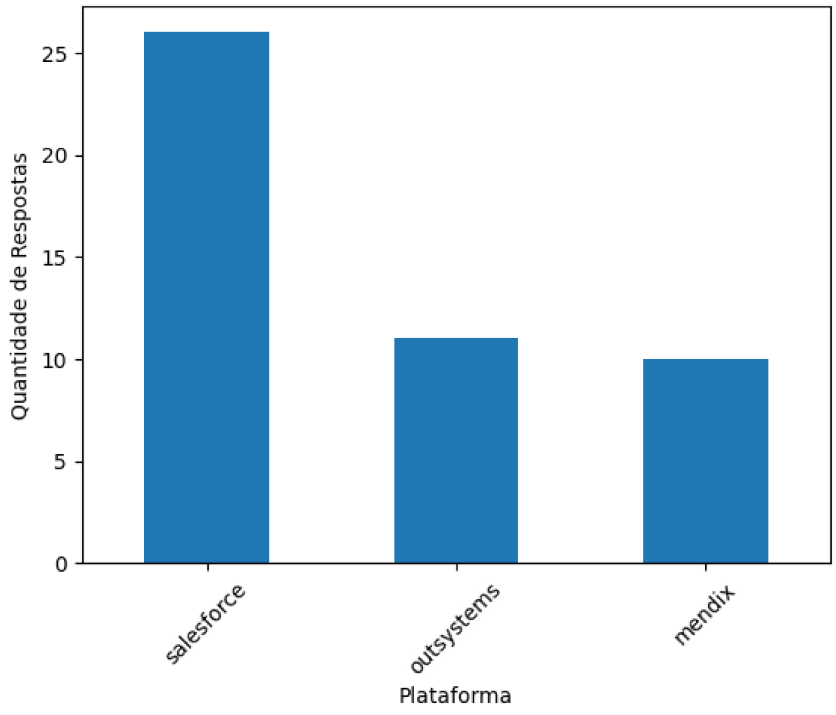
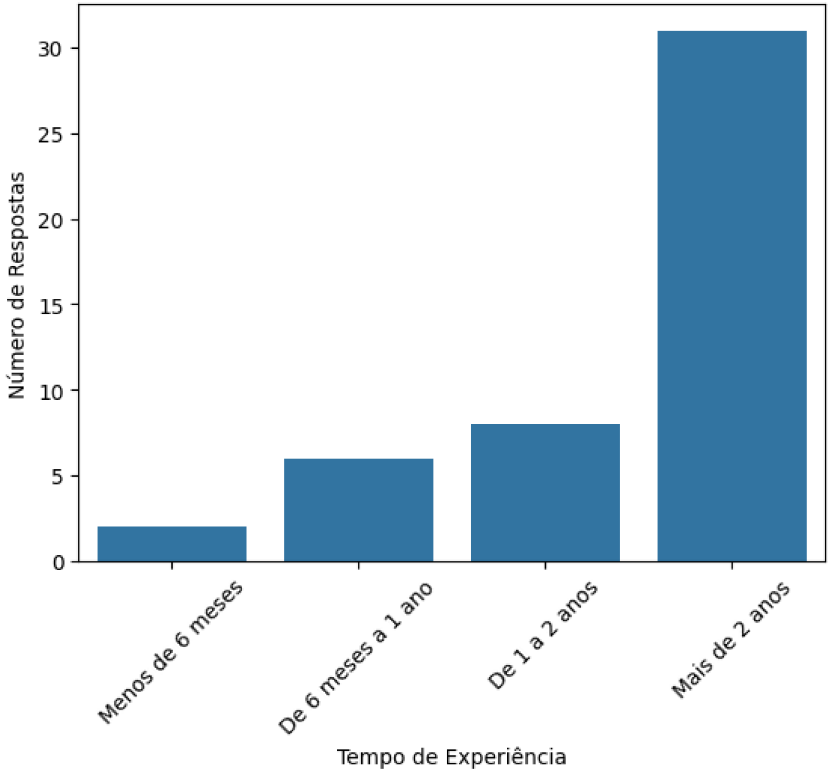


Figura 7 – Distribuição do tempo de experiência dos participantes



5.2 Distribuição Geral das Respostas às Afirmações

Foram analisadas 18 afirmações relacionadas a diferentes dimensões do uso de plataformas *low-code*. As respostas foram registradas em uma escala do tipo *Likert*, variando de 1 (discordância total) a 5 (concordância total). As afirmações com maiores médias de concordância foram:

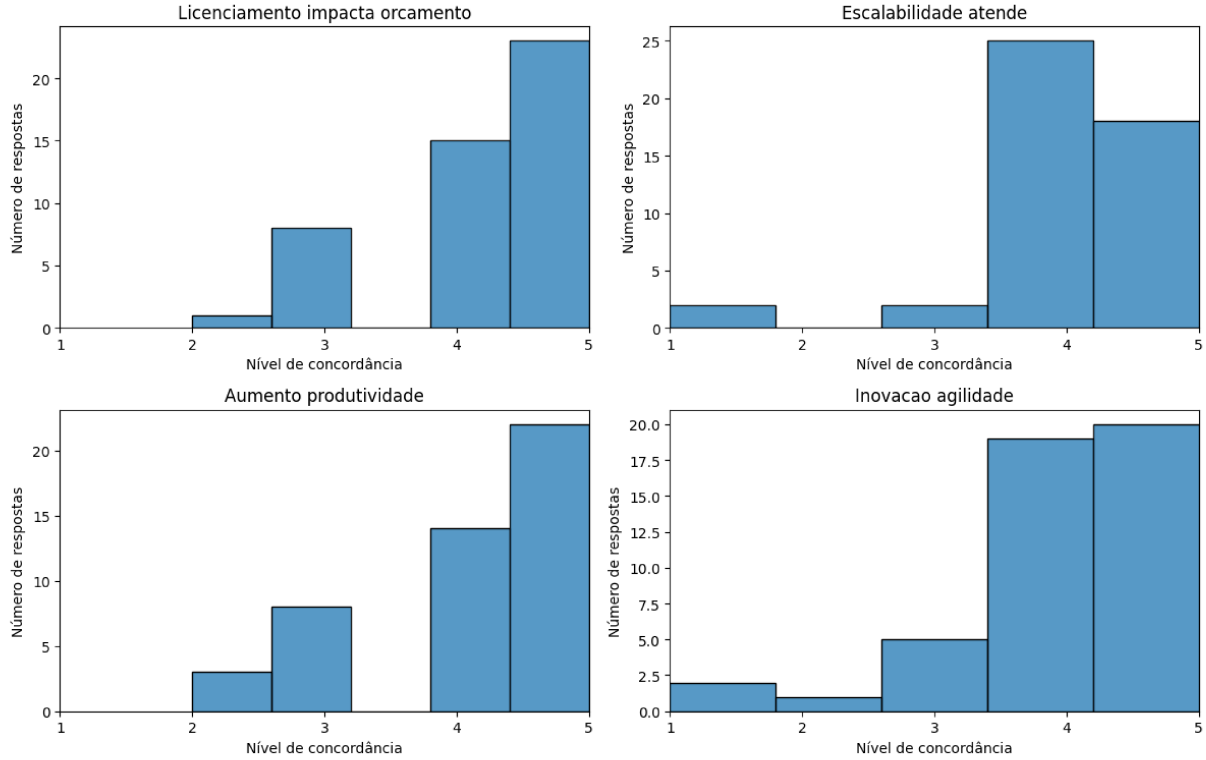
- Os custos de licenciamento da plataforma impactam o orçamento da área de TI (média = 4,29);
- A escalabilidade oferecida pela plataforma atende às necessidades da empresa em crescimento (média = 4,22);
- O uso da plataforma *low-code* aumentou a produtividade da equipe de desenvolvimento (média = 4,16);
- A plataforma contribui para a inovação e agilidade da empresa (média = 4,14).

Tais resultados indicam uma percepção predominantemente positiva quanto à capacidade das plataformas *low-code* de gerar valor para as organizações, especialmente em aspectos como produtividade, escalabilidade e inovação.

As distribuições das respostas para essas afirmações estão ilustradas na Figura 8.

Nota-se uma concentração de respostas nos níveis 4 e 5 da escala, reforçando a percepção positiva dos participantes em relação a essas dimensões avaliadas.

Figura 8 – Distribuição das respostas – afirmações com maiores médias



5.3 Comparação Entre Plataformas

Com o objetivo de identificar possíveis diferenças nas percepções entre as plataformas, foi realizada análise de variância (ANOVA) para cada uma das afirmações. Embora nenhuma variável tenha atingido significância estatística ($p > 0{,}05$), duas apresentaram p-valores próximos ao limiar de significância:

- Adequação da plataforma para o futuro da empresa ($p = 0{,}0594$);
- Capacidade da plataforma de suportar muitos usuários ($p = 0{,}0531$).

Esses resultados sugerem que, embora não se possam confirmar diferenças estatísticas significativas entre as plataformas, há tendências indicativas de variações nas percepções relacionadas à escalabilidade e à perspectiva futura das plataformas. Esses achados poderiam ser melhor explorados com uma amostra maior, a fim de identificar mais claramente se essas diferenças são, de fato, significativas.

As distribuições das respostas para essas afirmações estão ilustradas nas Figuras 9 e 12. Ambas as figuras mostram as diferenças nas respostas das plataformas em relação à

adequação futura e à capacidade de suportar muitos usuários.

Figura 9 – Continuará sendo adequada para os objetivos futuros da empresa.

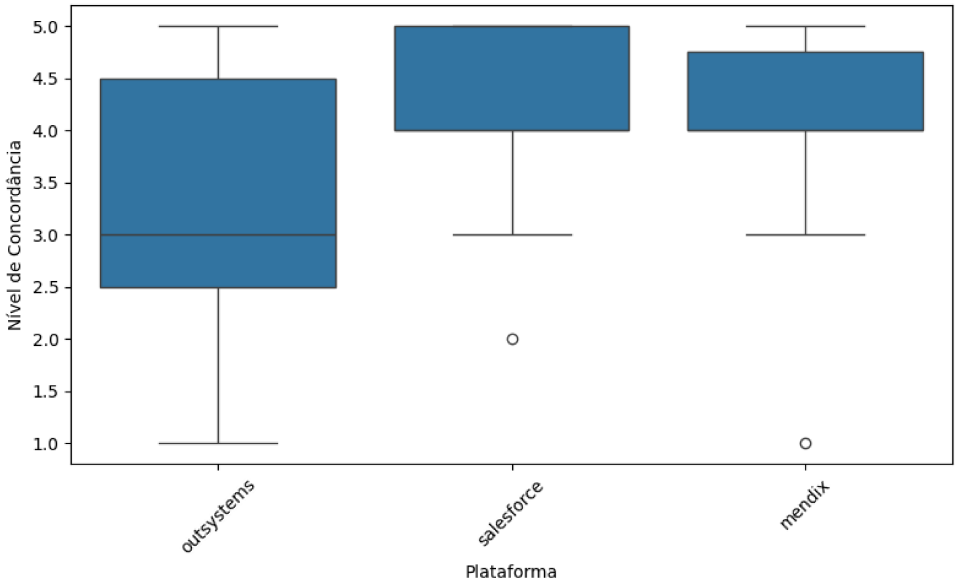
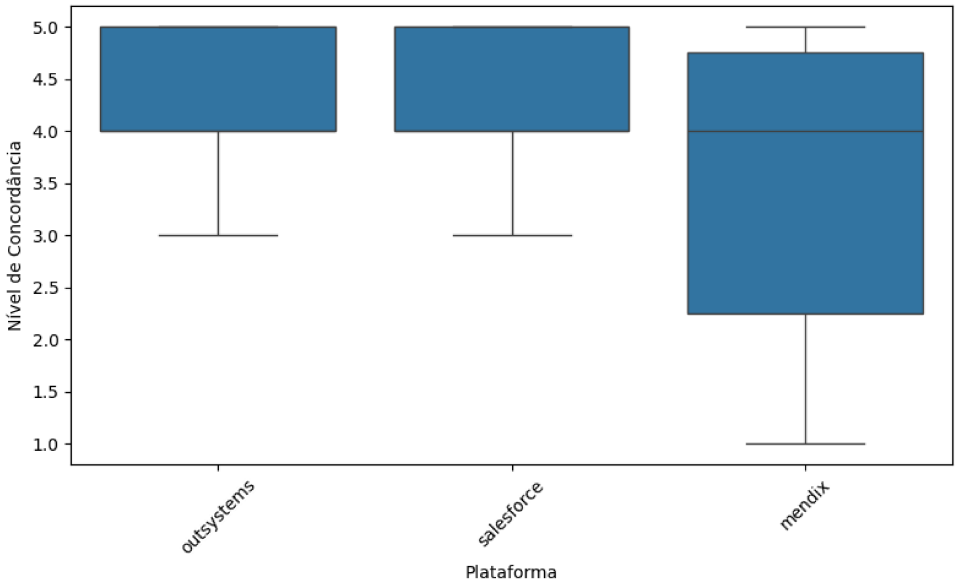


Figura 10 – Atende ao aumento no número de usuários simultâneos.



5.4 Correlação com o Tempo de Experiência

A variável *tempo de experiência* foi transformada em uma escala ordinal (de 0 a 3) para análise estatística. Em seguida, foi calculada a correlação de Spearman entre essa variável e cada uma das 18 afirmações.

De forma geral, as correlações foram fracas. No entanto, duas variáveis apresentaram

correlação estatisticamente significativa com o tempo de experiência:

- O custo total de propriedade aumentou com o tempo: correlação negativa $r = -0{,}32$, $p = 0{,}0271$;
- As atualizações da plataforma causam problemas nos sistemas existentes: correlação negativa $r = -0{,}30$, $p = 0{,}0361$.

Esses achados indicam que participantes com maior tempo de uso das plataformas tendem a discordar mais dessas afirmações, o que pode sugerir uma maior adaptação ou familiaridade com a solução ao longo do tempo.

As distribuições das respostas para essas afirmações, em relação ao tempo de experiência, podem ser visualizadas nos boxplots a seguir:

Figura 11 – O custo total de propriedade aumentou com o tempo.

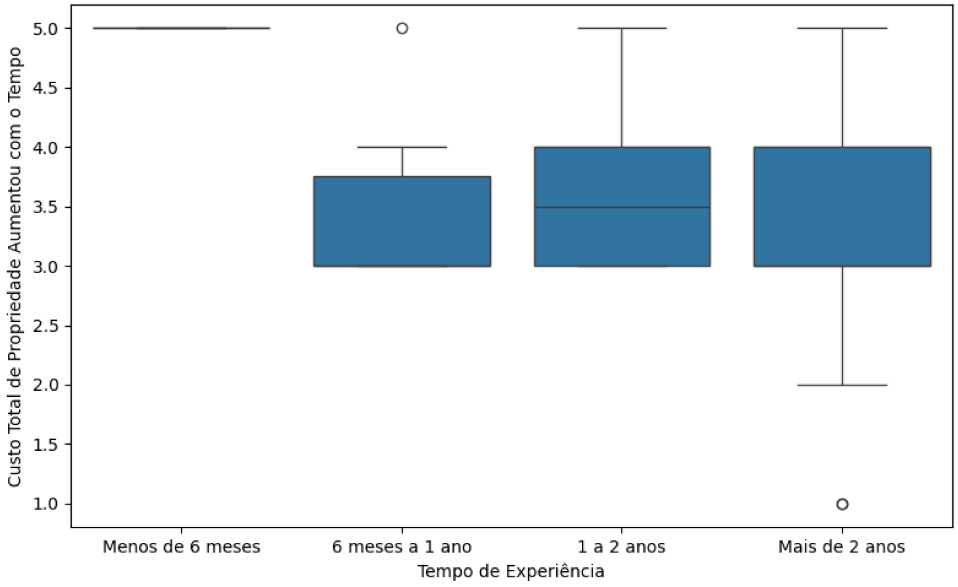
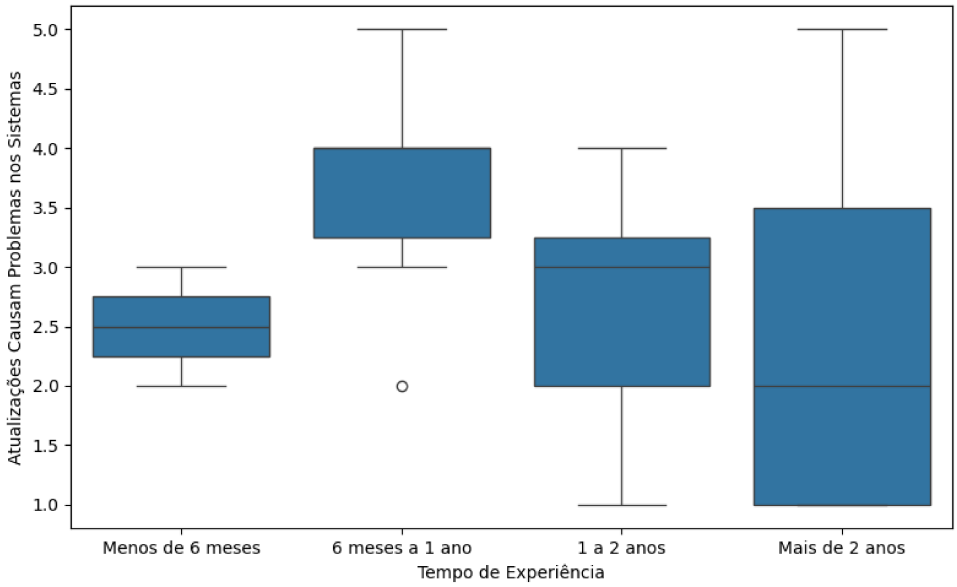


Figura 12 – As atualizações causam problemas nos sistemas existentes.



5.5 Relações Entre as Afirmações

Foi realizada uma análise de correlação entre todas as 18 afirmações utilizando o coeficiente de Spearman. A matriz de correlação revelou algumas associações positivas relevantes, como:

- Forte correlação entre "aumento da produtividade" e "redução do tempo de entrega" ($r = 0{,}73$);
- Correlação entre "a plataforma contribui para inovação e agilidade" e "adequação ao futuro da empresa" ($r = 0{,}62$).
- Correlação negativa entre "adequação futura da plataforma" e "possível migração para solução personalizada" ($r = -0{,}55$).

Tais correlações sugerem que percepções positivas em relação à inovação e produtividade estão associadas a uma menor necessidade de migração futura.

6 DISCUSSÃO

O principal achado desta pesquisa foi a percepção amplamente positiva dos profissionais sobre plataformas *low-code*, especialmente em aspectos como produtividade, escalabilidade e inovação organizacional. A alta média de concordância para afirmações como "a plataforma aumentou a produtividade" (média = 4,16) e "atende à escalabilidade necessária" (média = 4,22) mostra que essas ferramentas têm cumprido o papel de facilitar o desenvolvimento ágil, como já apontado na literatura (BOCK; FRANK, 2021; SAHAY *et al.*, 2020).

Outro ponto relevante foi a preocupação com o TCO e a manutenção no longo prazo. Apesar das médias elevadas para produtividade, afirmações como "o custo total de propriedade aumentou com o tempo" também tiveram concordância significativa, embora com variação de acordo com a experiência dos usuários. Isso indica que os benefícios iniciais não eliminam os desafios de sustentabilidade técnica, corroborando estudos como os de Alamin *et al.* (2021), que discutem o impacto negativo de atualizações e dependência da plataforma.

Um achado interessante foi a correlação negativa entre tempo de experiência e percepção de problemas com atualizações e TCO. Profissionais com mais tempo de uso tendem a relatar menos impactos negativos, o que sugere que a familiaridade com a plataforma pode mitigar parte das dificuldades relatadas por iniciantes. Esse resultado reforça que a experiência do usuário é um fator mediador importante na percepção sobre as plataformas *low-code* — aspecto pouco explorado em estudos anteriores e que representa uma contribuição original desta pesquisa.

Além disso, a ausência de diferença estatisticamente significativa entre plataformas não invalida a tendência observada de variações em aspectos como escalabilidade futura. Estudos com amostras maiores poderiam aprofundar essas possíveis diferenças, principalmente considerando que soluções como OutSystems e Mendix oferecem níveis distintos de customização e suporte técnico.

De forma geral, os resultados obtidos estão em conformidade com a literatura existente, especialmente no que se refere à vantagem de adoção rápida e à complexidade de manutenção posterior (KÄSS *et al.*, 2023). No entanto, este estudo amplia esse debate ao mostrar que a percepção de risco tende a diminuir com a experiência prática, o que pode orientar estratégias de treinamento e integração de novas equipes.

Por fim, vale destacar que as percepções positivas não significam ausência de riscos. Mesmo com altos índices de satisfação, os profissionais reconhecem limitações técnicas, o que

reforça a importância de uma adoção planejada, considerando fatores como crescimento da empresa, dependência tecnológica e necessidade futura de migração ou customização.

7 CONCLUSÃO

A partir da análise dos dados coletados, foi possível avaliar o impacto do uso de plataformas *low-code* no desenvolvimento, manutenção e escalabilidade de soluções de *software*. A pesquisa revelou percepções majoritariamente positivas por parte dos profissionais envolvidos, com destaque para a escalabilidade das plataformas, aumento da produtividade e contribuição para a inovação organizacional.

Os resultados obtidos indicam que as plataformas *low-code* podem oferecer vantagens significativas, especialmente no que tange à melhoria da eficiência do time de desenvolvimento e à capacidade de adaptação às necessidades da empresa. Em particular, a escalabilidade das plataformas foi altamente valorizada pelos participantes, refletindo uma percepção de que essas ferramentas atendem bem às necessidades atuais e futuras das organizações.

A correlação entre o tempo de experiência com as plataformas e as percepções sobre custo e atualização também sugere que usuários mais experientes tendem a ter uma visão mais positiva sobre as soluções, possivelmente devido à maior familiaridade com as nuances dessas ferramentas. Esse achado corrobora a literatura existente, que aponta que a adaptação às plataformas *low-code* se torna mais eficaz com o tempo de uso, minimizando desafios relacionados a custos e manutenção.

No entanto, as análises de variação entre as diferentes plataformas revelaram que, embora não tenha sido encontrada significância estatística, há indicações de que algumas plataformas podem ter características diferenciadas em termos de adequação futura e capacidade de escalabilidade, temas que podem ser melhor explorados em estudos futuros com amostras maiores.

Com base nestes achados, concluímos que o uso de plataformas *low-code* oferece benefícios significativos em termos de produtividade, inovação e escalabilidade, mas que sua implementação deve ser cuidadosamente planejada, considerando as necessidades específicas de cada organização e a experiência dos profissionais envolvidos. A continuidade da pesquisa em diferentes contextos organizacionais e com um número maior de participantes pode trazer insights ainda mais profundos sobre as vantagens e limitações dessas ferramentas no cenário atual de desenvolvimento de *software*.

Portanto, é fundamental que organizações avaliem o uso de plataformas low-code de forma estratégica, ponderando entre ganhos imediatos e desafios a longo prazo, especialmente em contextos que demandam soluções escaláveis e sustentáveis.

7.1 Trabalhos Futuros

Para aprofundar os achados desta pesquisa, sugere-se como linha futura de investigação a realização de um estudo comparativo entre dois perfis distintos de profissionais que utilizam plataformas *low-code*: aqueles com formação prévia em lógica de programação (como desenvolvedores) e aqueles sem esse conhecimento formal (como analistas de negócio, gestores ou usuários corporativos). O objetivo seria analisar como o *background* técnico influencia a curva de aprendizado, a autonomia no uso da plataforma e a forma como os desafios são enfrentados no processo de desenvolvimento.

Essa abordagem permitiria identificar se há diferenças significativas na percepção de usabilidade, produtividade e nas limitações enfrentadas por cada grupo, contribuindo para o desenvolvimento de treinamentos mais direcionados e para a melhoria das estratégias de adoção das ferramentas dentro das organizações.

Outra proposta de continuidade seria a realização de um estudo de caso aprofundado em uma organização de médio ou grande porte que utilize plataformas *low-code* em seus processos internos de desenvolvimento. Por meio da coleta de dados ao longo do tempo, seria possível analisar os impactos reais dessas ferramentas na manutenção, no custo total de propriedade, na escalabilidade e na produtividade da equipe. Essa abordagem permitiria observar, em um ambiente real, os benefícios e limitações discutidos na literatura e oferecer subsídios mais concretos para organizações que consideram adotar tais plataformas em larga escala.

REFERÊNCIAS

ALAMIN, M. A. A.; MALAKAR, S.; UDDIN, G.; AFROZ, S.; HAIDER, T. B.; IQBAL, A. An empirical study of developer discussions on low-code software development challenges. In: **2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)**. [S.l.]: IEEE, 2021. p. 46–57.

ALSAADI, H. A.; RADAIN, D. T.; ALZAHRANI, M. M.; ALSHAMMARI, W. F.; ALAHMADI, D.; FAKIEH, B. Factors that affect the utilization of low-code development platforms: survey study. **Romanian Journal of Information Technology & Automatic Control/Revista Română de Informatică și Automatică**, v. 31, n. 3, 2021.

BOCK, A. C.; FRANK, U. Low-code platform. **Business & Information Systems Engineering**, Springer, v. 63, p. 733–740, 2021.

KÄSS, S.; STRAHRINGER, S.; WESTNER, M. Practitioners' perceptions on the adoption of low code development platforms. **Ieee Access**, IEEE, v. 11, p. 29009–29034, 2023.

Mendix Brasil. **Mendix Brasil**. 2024. Acesso em: 11 dez. 2024. Disponível em: <https://www.mendixbrasil.com>.

Microsoft. **Learn Microsoft**. 2024. Acesso em: 11 dez. 2024. Disponível em: <https://learn.microsoft.com>.

OLTROGGE, M.; DERR, E.; STRANSKY, C.; ACAR, Y.; FAHL, S.; ROSSOW, C.; PELLEGRINO, G.; BUGIEL, S.; BACKES, M. The rise of the citizen developer: Assessing the security impact of online app generators. In: **2018 IEEE Symposium on Security and Privacy (SP)**. [S.l.]: IEEE, 2018. p. 634–647.

OutSystems. **OutSystems**. 2024. Acesso em: 11 dez. 2024. Disponível em: <https://www.outsystems.com>.

SAHAY, A.; INDAMUTSA, A.; RUSCIO, D. D.; PIERANTONIO, A. Supporting the understanding and comparison of low-code development platforms. In: **2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. [S.l.]: IEEE, 2020. p. 171–178.

Salesforce. **Salesforce**. 2024. Acesso em: 11 dez. 2024. Disponível em: <https://www.salesforce.com>.

SUFI, F. Algorithms in low-code-no-code for research applications: a practical review. **Algorithms**, MDPI, v. 16, n. 2, p. 108, 2023.