

UMA TEORIA EMPIRICAMENTE FUNDAMENTADA SOBRE OS EFEITOS DO MÉTODO LOW-CODE NO DESENVOLVIMENTO DE SOFTWARE

Lucas Mello dos Santos¹

Kleinner Farias²

Resumo: No atual cenário de desenvolvimento de software, a pressão por agilidade e eficiência é mais intensa do que nunca. À medida que as empresas buscam criar e atualizar aplicativos de forma rápida para atender às demandas do mercado, as plataformas de desenvolvimento low-code (PDLC) emergiram como uma solução promissora. Essas PDLC permitem que desenvolvedores e até mesmo profissionais sem o conhecimento técnico avançado, criem aplicativos com maior velocidade e menos esforço, reduzindo significativamente o ciclo de desenvolvimento. Por conseguinte, este trabalho propõe o estabelecimento de uma teoria fundamentada empiricamente, visando oferecer orientações valiosas aos desenvolvedores no âmbito do desenvolvimento low-code. Para a formulação dessas proposições teóricas, foi realizada uma revisão aprofundada da literatura, aliada à experiência profissional do autor, desenvolvendo softwares com a PDLC Oracle APEX. As proposições geradas foram então validadas por meio de uma pesquisa quantitativa junto à comunidade de desenvolvedores imersos no low-code. A metodologia adotada consistiu na combinação dessa revisão criteriosa da literatura pertinente com a experiência prática acumulada, gerando hipóteses sobre o tema. Essas hipóteses foram submetidas à validação através de técnicas de pesquisa quantitativa. Através dessa abordagem interdisciplinar, busca-se não apenas fornecer insights para os desenvolvedores, mas também contribuir para o avanço do conhecimento na área do desenvolvimento de software low-code. O resultado deste estudo aspira fornecer uma estrutura teórico-prática e empiricamente respaldada, para beneficiar a comunidade de desenvolvedores nesse campo em constante evolução.

Palavras Chave: Desenvolvimento de software, low-code, produtividade.

1 INTRODUÇÃO

A transformação digital é uma realidade atual e envolve a adoção e aplicação de tecnologias digitais em todos os setores de uma empresa, abrangendo modelos de negócios, experiências do cliente, processos operacionais, e requer também uma mudança de mentalidade constante. Ela define-se como um fenômeno social ou evolução cultural, e para as empresas como uma evolução ou criação de modelo de negócios (HENRIETTE; FEKI; BOUGHZALA, 2016). Tendo o cenário da transformação digital como plano de fundo, as áreas de negócio e a TI devem buscar o desenvolvimento Low-code se quiserem aproveitar a vantagem de tempo que ele oferece (PHALAKE; JOSHI, 2021). O low-code é uma opção que aproxima as equipes de negócio e processos da área de Tecnologia da Informação (TI) e as inclui na participação da criação dos aplicativos (KHORRAM; MOTTU; SUNYÉ, 2020). Possibilitando uma maior eficiência

¹Graduando em Sistemas de Informação pela Unisinos. Email: lucas.mello.santos@outlook.com

²Possui doutorado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2012), mestrado em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (2008), graduação em Ciência da Computação pela Universidade Federal de Alagoas (2006) e em Tecnologia da Informação pelo Instituto Federal de Alagoas. Email: kleinnerfarias@unisinos.br

e redução de problemas nos processos, já que as partes com maior conhecimento dos procedimentos têm participação ativa desde a etapa de concepção até a fase de validação das soluções digitais. Sendo assim como dizem **Phalake e Joshi (2021)**, já que o método de desenvolvimento tradicional falha em acompanhar as demandas da área de negócio, o Low-code é fundamental para lidar com os desafios das organizações de forma ágil.

No panorama atual há uma carência na literatura em relação a estudos que investiguem os impactos e benefícios da adoção desse método de programação, bem como uma limitação da bibliografia disponível sobre o tema. A falta de uma metodologia consolidada que aborde de forma abrangente as vantagens do low-code como abordagem de desenvolvimento de software destaca a necessidade urgente de explorar esse tema para melhor compreendê-lo e propor sua utilização.

Diante desse crescente aumento da utilização das PDLC, alguns trabalhos foram realizados ao longo dos últimos anos. Trigo, Varajão e Almeida (2022) realizaram um estudo experimental comparando a produtividade no desenvolvimento low-code em contraste ao desenvolvimento baseado em código. Marques (2023) apresentou os resultados de um estudo comparativo entre PDLCs distintas. Al Alamin et al. (2021) realizaram um estudo empírico sobre as impressões da comunidade de desenvolvedores low-code sobre as PDLCs do mercado, baseados em tópicos de discussão no site Stackoverflow³. Alamin et al. (2023) realizaram um novo estudo empírico baseado em tópicos de discussão do site Stackoverflow, que trouxe novos dados e reflexões versus o trabalho apresentado em 2021(Al Alamin et al., 2021). Bock e Frank (2021) realizaram um estudo exploratório que englobou sete PDLCs, com o objetivo de entender suas características e analisá-las criticamente com base em pesquisas acadêmicas sobre desenvolvimento de sistemas de informação. Santos (2023) realizou uma revisão bibliográfica documental e qualitativa, que buscou avaliar como o low-code entrega vantagem competitiva para as áreas de engenharia. Dahlberg (2020) apresentou um estudo exploratório que objetivou compreender a experiência do desenvolvedor em ambientes que utilizam o low-code.

Este trabalho, portanto, apresenta a proposição de uma teoria sobre o método de desenvolvimento low-code, fundamentada empiricamente e utilizando como referência o framework de construção de teorias na engenharia de software proposto por Sjøberg et al. (2008). Para o embasamento da teoria foi realizada uma revisão aprofundada da literatura e para a avaliação da teoria proposta, foi realizada uma pesquisa quantitativa com trabalhadores da área de TI que atuam com low-code no seu dia-a-dia. Dentro desse público tivemos a adesão de 36 participantes, que responderam 20 questões divididas em 8 questões de caracterização do perfil e 12 questões para avaliar as proposições da teoria. Como resultados mais expressivos dessa pesquisa, temos que a maior parte dos entrevistados concordam que o treinamento do time de desenvolvimento afeta positivamente a utilização do low-code, e também concordam que o low-code reduz os custos do time de desenvolvimento devido ao aumento da produtividade.

Este trabalho é organizado da seguinte forma: A Seção 2 apresenta a fundamentação teórica

³<https://stackoverflow.com/>

sobre os principais conceitos necessários para o entendimento deste estudo. Na Seção 3 é realizada uma análise comparativa de trabalhos relacionados. Na Seção 4 a proposição da teoria é apresentada. A Seção 5 descreve a análise dos resultados obtidos. A Seção 6 apresentará as discussões adicionais sobre os resultados obtidos. E por fim, a Seção 7 apresenta as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais conceitos essenciais para o entendimento do trabalho desenvolvido.

2.1 Desenvolvimento Low-code

Conforme as empresas se tornam mais competitivas entre si, a capacidade de mudar e responder rapidamente é cada vez mais importante para o sucesso das corporações. Sendo assim, ferramentas low-code tornam-se estratégicas, considerando que proporcionam uma forma mais rápida e menos custosa de implementação de softwares na indústria (WASZKOWSKI, 2019). Além disso, o modelo low-code oferece vantagens, tornando maior a acessibilidade para que usuários não habituados em programação possam em um curto período de tempo tornarem-se novos desenvolvedores (KHORRAM; MOTTU; SUNYÉ, 2020; SAHAY et al., 2020).

O termo low-code foi usado, primeiramente, no contexto da transformação digital, em 2014, pela Forrester Research (RICHARDSON et al., 2014), que declara que as empresas preferem escolher alternativas de baixo código para entrega rápida e contínua, bem como para a realização de teste e obtenção de aprendizado. As abordagens que deram origem à programação low-code foram as de desenvolvimento de software baseado em modelo (template), desenvolvimento de aplicação rápida, geração automática de código e programação visual (WASZKOWSKI, 2019).

Segundo Luz (2021) as plataformas low-code consistem em plataformas que permitem o desenvolvimento de aplicações web de forma rápida e ágil. Essas plataformas contêm um conjunto de funcionalidades genéricas que podem ser personalizadas pelo utilizador. Uma das grandes vantagens de utilizar tais ferramentas deve-se ao fato de que elas permitem que tanto indivíduos que não sabem programar como programadores experientes consigam criar aplicações, fazendo com que o processo para os programadores experientes se torne muito mais rápido.

Ademais, nas plataformas low-code a criação de aplicativos não cabe mais somente aos desenvolvedores de software; os usuários corporativos comuns também podem criar aplicativos por conta própria. Nas interfaces mais simples de usuário com pouco código, de maneira intuitiva arrastar e soltar substitui a codificação linha por linha. A automatização assume a posição do trabalho manual. E, principalmente, a velocidade agiliza o processo de desenvolvimento.

O low-code é uma opção que aproxima as equipes de negócio e processos da área de Tecnologia da Informação (TI) e as inclui na participação da criação dos aplicativos (KHORRAM;

Tabela 1: Arquétipo da teoria de exemplo utilizada por Sjøberg et al. (2008)

Classe Arquétipo	Subclasses
Ator	Indivíduo, time, projeto, organização ou indústria
Tecnologia	Modelo de processo, método, técnica ferramenta ou linguagem
Atividade	Planejar, criar, modificar ou analisar (um sistema de software)
Sistema de Software	Sistemas de software podem ser classificados em muitas dimensões, como tamanho, complexidade, domínio de aplicação, projeto empresarial/científico/estudante ou administrativo/incorporado/tempo real, etc.

MOTTU; SUNYÉ, 2020). Essa abordagem também promove uma maior aceitação e adoção das soluções digitais pelas partes interessadas, pois elas se sentem parte do processo de tomada de decisão e têm a oportunidade de influenciar o resultado. Isso reduz a resistência à mudança e aumenta a probabilidade de sucesso da implementação. Esse paradigma também permite que o desenvolvedor gaste menos tempo com codificação e foque em questões de alto nível, como funcionalidades, estética e experiência do usuário (WASZKOWSKI, 2019). Assim, há um aumento na geração de valor e maiores ganhos estratégicos para o negócio.

Conforme as pesquisas da Forrester (HEFFNER et al., 2020), plataformas low-code podem acelerar o desenvolvimento de software em até 10 vezes comparado com os métodos tradicionais de programação. Inevitavelmente, esse mercado de plataformas low-code vem crescendo cerca de 40% ao ano, com um gasto que excedeu US \$21 bilhões até 2022. Ou seja, é inevitável utilizar dessas plataformas inovadoras para competir de forma mais eficaz.

2.2 Construção de teorias na Engenharia de Software

As teorias são ferramentas que nos ajudam a entender e organizar nossas observações. Para desenvolvê-las, é necessário questionar o “porquê” de um fenômeno específico ocorrer. Por exemplo, se pretende-se formular um argumento que justifique a existência de uma lei e, conseqüentemente, por que essa lei pode ser considerada verdadeira, precisaremos de uma teoria (CHAPETTA, 2018; ENDRES; ROMBACH, 2003).

Uma teoria de Engenharia de Software(ES) é conjecturada para elucidar ou prever um fenômeno ocorrendo na ES. O cenário característico de ES é que um **ator** aplica **tecnologias** para realizar determinadas **atividades** em um **software** (existente ou planejado) (SJØBERG et al., 2008). Na teoria de exemplo utilizada por Sjøberg et al. (2008) temos os arquétipos apresentados na Tabela 1.

Segundo Sjøberg et al. (2008) os termos destacados acima são as classes arquétipos do Framework para a construção de teorias na ES, que constituem os construtos da teoria. Tendo os construtos definidos, é necessário apresentar claramente as proposições e suas explicações, e o escopo da teoria de ES.

Ainda segundo Sjøberg et al. (2008), a construção da teoria se dá em 5 passos que são os descritos a seguir, juntamente com os elementos da teoria utilizada como exemplo por ele:

Tabela 2: Construtos da teoria de exemplo utilizada por Sjøberg et al. (2008)

Construtos	
C1	Método de desenvolvimento baseado em UML
C2	Custos
C3	Comunicação
C4	Design
C5	Documentação
C6	Testabilidade
C7	Treinamento
C8	Coordenação de requisitos e times
C9	Código Legado

Tabela 3: Proposições da teoria de exemplo utilizada por Sjøberg et al. (2008)

Proposições	
P1	O uso do método de desenvolvimento baseado em UML aumenta os custos
P2	O uso do método de desenvolvimento baseado em UML afeta positivamente a comunicação
P3	O uso do método de desenvolvimento baseado em UML afeta positivamente o design
P4	O uso do método de desenvolvimento baseado em UML afeta positivamente a documentação
P5	O uso do método de desenvolvimento baseado em UML afeta positivamente a testabilidade
P6	Os efeitos positivos do método de desenvolvimento baseado em UML são reduzidos se o treinamento não for suficiente e adaptado
P7	Os efeitos positivos do método de desenvolvimento baseado em UML são reduzidos se houver coordenação insuficiente das atividades de modelagem entre equipes distribuídas trabalhando no mesmo projeto
P8	Os efeitos positivos do método de desenvolvimento baseado em UML são reduzidos se a atividade incluir modificação de código legado

- **Definição dos construtos da teoria:** O primeiro passo envolve a identificação e escolha dos construtos para a teoria, podendo ocorrer através da criação de novos construtos em uma nova teoria, da introdução de novos construtos em uma teoria existente, da exclusão de construtos de uma teoria existente, da adição e exclusão de construtos de uma teoria existente, ou da definição mais precisa dos construtos de uma teoria existente. Sjøberg et al. (2008) utiliza para exemplo os construtos da Tabela 5.
- **Definição das proposições da teoria:** O segundo passo constitui-se da especificação das proposições que a teoria irá apresentar pelo meio de novas proposições dentre existentes ou novos construtos em uma nova teoria, da exclusão de proposições de uma teoria existente, da adição e deleção de proposições de uma teoria existente ou de uma melhor definição das proposições de uma teoria existente. Sjøberg et al. (2008) utiliza para exemplo as proposições da Tabela 3.

- **Explicações para justificar a teoria:** O terceiro passo fornece o porquê da teoria. O ponto principal deste passo é o de prover premissas claras e argumentos lógicos para os construtos e proposições da teoria. Sjøberg et al. (2008) utiliza para exemplo a explicação abaixo referente a proposição *P4*.

E4. A documentação é: mais completa; mais consistente devido à rastreabilidade no meio de modelos e entre modelos e código; mais legível e facilita a localização de informações específicas, devido a um formato comum; mais entendível para pessoas não técnicas; pode ser vista de diferentes perspectivas devido a diferentes tipos de diagrama.

- **Determinar o escopo da teoria:** O quarto passo do processo de construção da teoria pode ser efetuado de duas formas. Sendo elas: a especificação rigorosa dos valores do construto com os quais a teoria será fundamentada, ou em contraponto, a especificação rigorosa dos valores com os quais a teoria não será fundamentada. Sjøberg et al. (2008) utiliza para exemplo o escopo abaixo:

A teoria deve ser aplicável a projetos distribuídos que criam e modificam subsistemas grandes, incorporados e críticos para a segurança, com base em código legado ou novo código.

- **Validar a teoria por meio de uma pesquisa empírica:** No último passo, para efetuar a validação da teoria, diferentes tipos de estudos empíricos podem ser empregados, como estudos de caso, estudos experimentais, pesquisa quantitativa, pesquisa exploratória, etc.

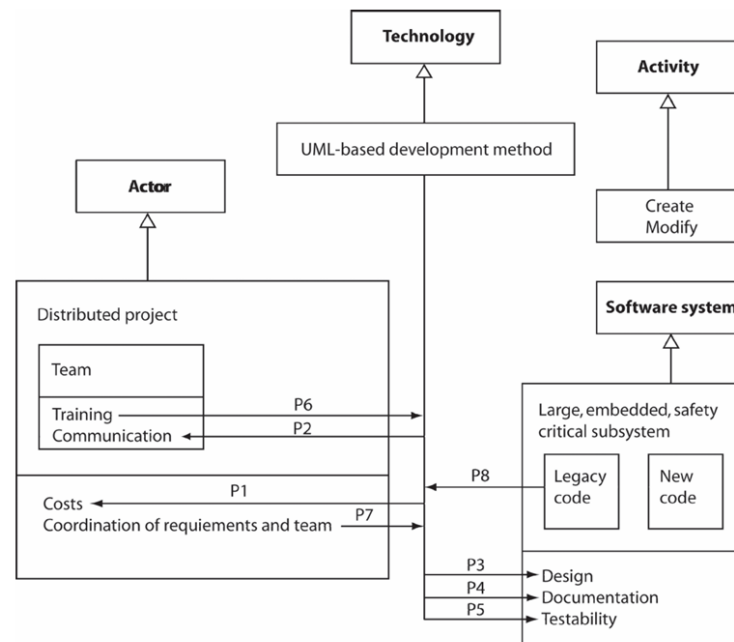
Ao final dos 5 passos descritos por Sjøberg et al. (2008), deve-se ter como resultado um diagrama como o que está exemplificado na Figura 1. Nesta figura cada construto é representado por uma classe arquétipo (Ator: Projeto Distribuído) ou um atributo de classe (Custo: atributo da classe Projeto Distribuído). As relações são modeladas como setas, isso significa que em uma seta de *A* para *B*, *A* afeta *B*, sendo *A* uma classe ou um atributo, e *B* um atributo (SJØBERG et al., 2008). Um exemplo de relações, a seta *PI*, indica que a classe arquétipo *Método de desenvolvimento baseado em UML* afeta de alguma forma o atributo *Custos*, pois a direção da seta parte do *Método de desenvolvimento baseado em UML* para o atributo *Custos*.

2.3 Desenvolvedor Cidadão

Como consequência do movimento da transformação digital no mundo corporativo, da ascensão das PDLs e da escassez de talentos na área do desenvolvimento, estão culminando em um envolvimento maior das áreas de negócio com a TI e no surgimento de um novo perfil dentro dessas áreas, o de Desenvolvedor Cidadão (Citizen Developer) (LEBENS et al., 2022).

O Desenvolvedor Cidadão é um profissional não-técnico que cria novos aplicativos para atender às necessidades das áreas de negócios de sua empresa, sem precisar escrever uma linha

Figura 1: Exemplo de diagrama que representa uma teoria no Framework de Sjøberg et al. (2008)



Fonte: (SJØBERG et al., 2008)

de código. Eles geralmente são especialistas em sua área de atuação, podendo ser gerentes de departamento ou colaboradores individuais.

Por meio do desenvolvimento realizado por esse perfil, há uma aceleração da transformação digital dentro das corporações, aumentando a eficiência da área de negócio com automatização de processos que simplificam o trabalho dos demais colegas não técnicos e melhoram a colaboração da equipe (LEBENS et al., 2022). O conceito de desenvolvedor cidadão defende que qualquer pessoa das equipes pode criar soluções para atender a demandas pontuais nos negócios.

3 TRABALHOS RELACIONADOS

Esta seção apresenta uma análise dos trabalhos relacionados disponíveis na literatura, para isso foi feita uma seleção de trabalhos que abordam o uso de plataformas de desenvolvimento de software low-code. Os trabalhos selecionados são apresentados na Seção 3.1. A análise comparativa e oportunidades de pesquisa são discutidos na Seção 3.2.

Os trabalhos foram selecionados através de pesquisas na plataforma Google Scholar⁴ e pelos termos: "Low Code Development Productivity", "Low-Code", "Low Code Empirical Study" e "Low Code Platform".

⁴<https://scholar.google.com/>

3.1 Análise dos Trabalhos Relacionados

Trigo, Varajão e Almeida (2022): apresentaram os desfechos de um experimento conduzido com o intuito de comparar a eficiência de duas tecnologias no desenvolvimento de software, especificamente no contexto low-code e no desenvolvimento baseado em código. Os resultados obtidos demonstram uma clara vantagem na agilidade do desenvolvimento e na manutenção de software ao utilizar a tecnologia low-code, evidenciando seu notável potencial. Através desse estudo, se torna possível apoiar com base científica, os potenciais ganhos de produtividade da tecnologia low-code, porém não podemos esquecer algumas limitações que se apresentam, tais como: escolha limitada de tecnologias testadas (apenas duas), foco em um tipo específico de aplicação (software de sistemas de informação de gestão) e baixa complexidade da aplicação testada. Além disso, algo que pode ameaçar a validade do estudo, são os perfis culturais de cada empresa, que independente da tecnologia escolhida, pode impactar na produtividade do desenvolvedor.

Marques (2023): realizou um estudo analítico sobre plataformas de desenvolvimento low-Code, focando em três plataformas, sendo elas Microsoft Power Apps, Mendix e Outsystems. Foram realizadas análises de diversas características dessas plataformas, como também comparações entre elas. Além disso, realizou-se um estudo empírico envolvendo múltiplos participantes oriundos de diversas universidades. Portanto, essa investigação pode ser aplicada como uma ferramenta para avaliar a usabilidade das plataformas. Ao final do estudo, a autora conclui que plataformas de desenvolvimento low-code simplificam o desenvolvimento ao permitirem que um mesmo desenvolvedor atue no backend e frontend sem grandes divisões. Também destaca que apesar do que é vendido pelos donos dessas plataformas, alguns pontos em sua utilização não são tão intuitivos e um pouco complexas se comparadas ao uso de código tradicional. De toda forma, apesar dos pontos mencionados, ela também conclui que é mais acessível aprender a utilizar plataformas low-code do que dominar múltiplas linguagens de programação para desenvolvimento completo de aplicações.

Al Alamin et al. (2021): realizaram um estudo empírico analisando tópicos de discussão no portal Stackoverflow⁵, relacionados ao desenvolvimento de software com plataformas de low-code. Os tópicos foram organizados em quatro categorias: Customização, Adoção de Plataforma, Gerenciamento de Banco de Dados e Integração de Terceiros. Neste trabalho, os autores concluem que os desenvolvedores consideram a categoria de tópico de integração de API externa a mais desafiadora e a categoria de banco de dados a menos difícil. O tratamento dinâmico de eventos é o tópico mais popular e também o mais desafiador. Também foi encontrada uma grave falta de boa documentação dos fornecedores das plataformas de desenvolvimento low-code, que impede uma adaptação suave as plataformas.

Alamin et al. (2023): realizaram um estudo empírico sobre tópicos de discussão no Stackoverflow, que estão relacionados ao desenvolvimento de software utilizando tecnologias low-

⁵<https://stackoverflow.com/>

code. Há nesse trabalho uma versão atualizada do artigo de **Al Alamin et al. (2021)**. Os tópicos foram agrupados em cinco categorias: Personalização de Aplicativos, Gerenciamento de Banco de Dados e Arquivos, Adoção de Plataforma, Manutenção de Plataforma e Integração de API de Terceiros. Através desse estudo, os autores concluem que categoria de tópico Adoção de plataforma ganhou popularidade recentemente. Por outro lado também identificam que os profissionais consideram os tópicos relacionados à adoção e manutenção de plataformas mais desafiadores. As perguntas do tipo "como" são as mais comuns, mas a pesquisa revela que os profissionais consideram as perguntas do tipo "o que" e "por que" são mais difíceis.

Bock e Frank (2021): conduziram uma investigação exploratória abrangendo sete plataformas de desenvolvimento low-code, visando compreender sua natureza e avaliá-las de forma crítica à luz dos estudos de pesquisa em desenvolvimento de sistemas de informação. O estudo revela que quase nenhuma característica do desenvolvimento low-code é inovadora por si só, sendo o que distingue elas é a forma com que integram, em um só ambiente, múltiplos componentes de design de sistema bem conhecidos e tradicionais, de modo a reduzir os esforços de tarefas rotineiras na implementação de aplicações de negócios dentro dos limites de certas estruturas mais ou menos restritivas.

Santos (2023): apresentou uma revisão bibliográfica documental e qualitativa, através de uma pesquisa observacional e exploratória, objetivando-se avaliar como a tecnologia low-code / no-code pode gerar uma vantagem competitiva nas áreas de engenharia. Ao fim do estudo o autor conclui que as plataformas de desenvolvimento no-code e low-code conseguem entregar vantagem competitiva a área de engenharia devido a eficiência dessas plataforma no que tange a escalabilidade e melhor experiência dos usuários.

Dahlberg (2020): procedeu com um estudo que consiste em uma pesquisa qualitativa visando obter uma compreensão da experiência do desenvolvedor em ambientes que utilizam o low-code. O estudo foi realizado com uma empresa de TI recentemente especializada em soluções de low-code, onde os participantes foram escolhidos com base na experiência anterior com desenvolvimento de low-code. As principais experiências positivas encontradas foram sentir-se mais produtivo, melhorar o relacionamento com o cliente, focar no objetivo, entender o desenvolvedor compartilhado e aprender rapidamente. As principais experiências negativas encontradas foram: ter trabalho limitado, liberdade e criatividade limitadas, documentação e visão geral inadequadas e ter capacidades de trabalho em equipa fracas e inseguras.

3.2 Análise Comparativa e Oportunidades de Pesquisa

A análise comparativa foi conduzida utilizando critérios específicos. Estudos anteriores realizados por (VIEIRA; FARIAS, 2020; RUBERT; FARIAS, 2021), os quais adotaram essa abordagem, evidenciaram sua eficácia como meio de estabelecer comparações entre trabalhos e identificar oportunidades de pesquisa.

Ao realizar a análise comparativa, foram identificados pontos de semelhança e distinção

entre o trabalho atual e os estudos relacionados previamente selecionados. Essa comparação foi essencial para a identificação objetiva de oportunidades de pesquisa. Para este fim, foram definidos 5 Critérios de Comparação, detalhados a seguir:

- **Estudo Empírico (CC01):** trabalhos que realizaram o estudo de forma empírica.
- **Aplicação programada em Low-code (CC02):** estudos que realizaram o desenvolvimento de um software com o Low-code.
- **Métricas (CC03):** estudos que utilizaram métricas para analisar a efetividade do low-code .
- **Low-code (CC04):** estudos onde o Low-code fazia parte do tema central de pesquisa.
- **Oracle APEX (CC05):** estudos onde houve análise de algum aspecto do Low-code, relacionado a plataforma Oracle Application Express.

Tabela 4: Análise comparativa dos trabalhos relacionados.

Trabalho Relacionado	Critério de Comparação				
	CC1	CC2	CC3	CC4	CC5
Trabalho proposto	●	●	●	●	●
Trigo, Varajão e Almeida (2022)	●	●	●	●	○
Marques (2023)	●	●	●	●	◐
Al Alamin et al. (2021)	●	○	○	●	○
Alamin et al. (2023)	●	○	○	●	●
Bock e Frank (2021)	○	◐	●	●	○
Santos (2023)	●	○	○	●	○
Dahlberg (2020)	●	○	●	●	○

● Similar ◐ Parcialmente similar ○ Não similar

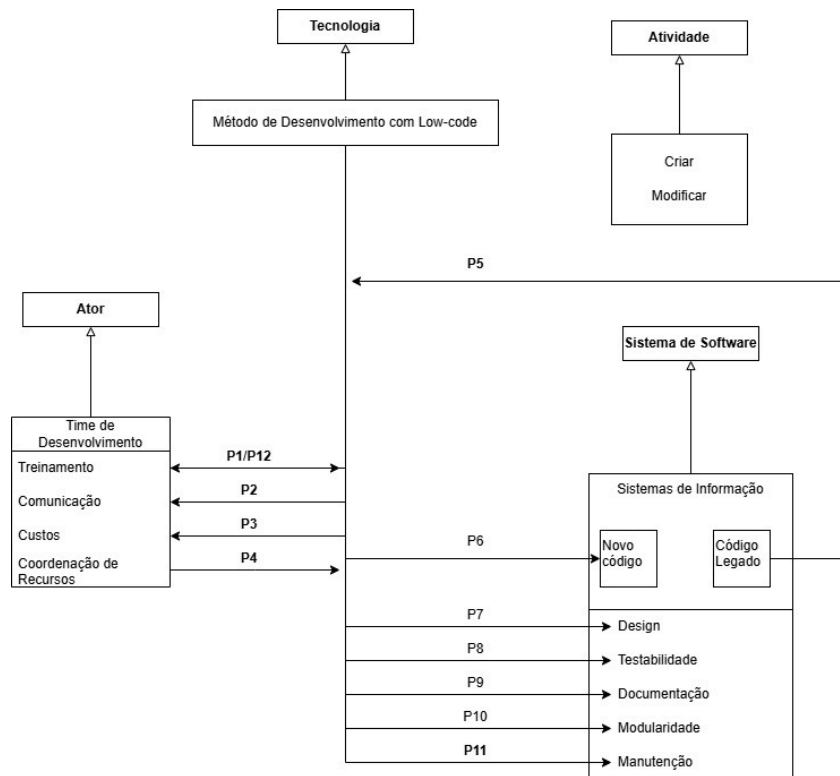
Fonte: elaborado pelo autor.

O resultado da comparação dos trabalho relacionados e do trabalho proposto com base nos Critérios de Comparação é demonstrado na Tabela 4. Analisando o resultado da comparação, foram identificados as seguintes pontos de oportunidades:

- a maior parte dos trabalhos realizaram o estudo de forma empírica sobre o Low-code;
- poucos trabalhos mencionaram ou realizaram algum tipo de análise referente ao Oracle APEX, mesmo essa sendo um dos líderes de mercado melhores avaliados pelos usuários, conforme pesquisa da Gartner⁶ em 2022;
- menos da metade dos trabalhos avaliados realizou o desenvolvimento de um software utilizando o low-code, para o levantamento de hipóteses e/ou criação de teorias.

⁶<https://blogs.oracle.com/apex/post/oracle-named-a-2022-gartner-peer-insights-voice-of-the-customer-enterprise-low-code-application-platforms>

Figura 2: Diagrama que representa a teoria, utilizando o Framework de Sjøberg et al. (2008)



Fonte: elaborado pelo autor

Oportunidade de Pesquisa. Com base nos pontos de oportunidades sinalizados, foi identificada a seguinte oportunidade de pesquisa: execução de estudos experimentais de forma empírica sobre o low-code para a elaboração de uma teoria que busca auxiliar a comunidade de desenvolvedores dessa tecnologia e o perfil de "Desenvolvedores Cidadãos". Esta oportunidade de pesquisa é explorada nas próximas seções.

4 PROPOSIÇÃO DA TEORIA

Esta seção apresentará o desenvolvimento da teoria utilizando como guia o Framework para a construção de teorias em ES, elaborado por Sjøberg et al. (2008). Ao final da sua aplicação o objetivo é termos um diagrama como o exemplo representado na Figura 1, já apresentado na Seção 2.2. Conforme os cinco passos do Framework forem sendo apresentados, também já serão definidos os Construtos e as Proposições da Teoria, tendo como base o próprio trabalho de Sjøberg et al. (2008), o trabalho de Sousa et al. (2018), o trabalho de Luz, Pinto e Bonifácio (2019) e a experiência profissional do autor, desenvolvendo softwares com a PDLC Oracle APEX. A Figura 2 apresenta o resultado da teoria que será desenvolvida durante esse capítulo.

Tabela 5: Construtos da teoria

Classe Arquétipo	Construto
Tecnologia	Método de Desenvolvimento com low-code
Ator	Time de Desenvolvimento
	Treinamento
	Comunicação
	Custos
	Coordenação de Recursos
Sistema de Software	Novo Código
	Código Legado
	Design
	Testabilidade
	Documentação
	Modularidade
	Manutenção

Fonte: elaborado pelo autor.

4.1 Passo 1: Definição dos Construtos da Teoria

Após revisão da literatura na Seção 2 e da análise dos trabalhos relacionados na Seção 3, foi possível elaborar os construtos, que são as caixas no diagrama de exemplo, identificando-os por suas classes de arquétipos, conforme lista abaixo e a Tabela 5:

- **Ator:** Time de desenvolvimento;
- **Tecnologia:** Método de Desenvolvimento com Low-code;
- **Atividade:** Criar ou modificar(um sistema de software);
- **Sistema de Software:** Sistemas de informação que incluem novo código e código legado.

4.2 Passo 2: Definição das Proposições da Teoria

Possuindo a definição dos construtos, realizou-se a definição das proposições, que são as setas no diagrama de exemplo e representam as relações entre um construto e outro, e como eles se afetam. As proposições foram elaboradas seguindo como referência trabalhos consolidados como o de Sjøberg et al. (2008) e de Sousa et al. (2018). Abaixo segue a lista das proposições:

- **P1:** O treinamento do time de desenvolvimento afeta positivamente o uso do método de desenvolvimento low-code ;
- **P2:** O método de desenvolvimento low-code afeta positivamente a comunicação entre o time de desenvolvimento;
- **P3:** O método de desenvolvimento low-code afeta positivamente os custos do time de desenvolvimento, visto o aumento da produtividade proporcionado;

- **P4:** A coordenação de recursos, quando bem executada, aumenta os efeitos positivos do método de desenvolvimento low-code;
- **P5:** O código legado deve ser analisado de forma detalhada para que os efeitos positivos do método de desenvolvimento low-code sejam percebidos;
- **P6:** O método de desenvolvimento low-code afeta positivamente a criação ou modificação de novo código;
- **P7:** O método de desenvolvimento low-code afeta positivamente o Design, simplificando-o;
- **P8:** O método de desenvolvimento low-code afeta positivamente a Testabilidade do sistema, facilitando a criação de casos de teste e sua abrangência;
- **P9:** O método de desenvolvimento low-code afeta positivamente a documentação do sistema, tornando-a mais simples de criar e estudá-la;
- **P10:** O método de desenvolvimento low-code afeta positivamente a modularidade do sistema
- **P11:** O método de desenvolvimento low-code afeta positivamente a manutenção de sistemas;
- **P12:** O método de desenvolvimento low-code facilita o treinamento do time de desenvolvimento, tornando-o mais simples e prático.

4.3 Passo 3: Fornecer explicações para justificar a teoria

Neste passo foi realizada a explicação dos porquês que justificam as proposições da teoria. Abaixo foram listadas as explicações para as proposições onde cada explicação faz referência a uma proposição, tendo *E1* referente a proposição *P1* e assim sucessivamente.

- **E1:** Segundo Trigo, Varajão e Almeida (2022) devido ao método low-code possibilitar que o desenvolvimento seja realizado em grande parte de forma gráfica com pouca ou nenhuma linha de código, permitindo que pessoas sem treinamentos possam criar aplicações. Sendo assim entende-se que com o treinamento adequado melhora a competência da equipe em usar plataformas low-code, maximizando os benefícios dessas ferramentas.
- **E2:** A comunicação é facilitada em grande parte pois como descreve Bock e Frank (2021) as PDLC concentram em um único ambiente, vários sistemas comuns e componentes tradicionais de design de modo que simplifica e padroniza componentes, tornando a comunicação mais clara e efetiva entre membros da equipe com diferentes níveis de habilidade técnica.

- **E3:** Conforme argumentam Bock e Frank (2021) e Phalake e Joshi (2021), o aumento de produtividade e redução do tempo de desenvolvimento são diretamente proporcionados pelo uso de plataformas low-code, resultando em uma diminuição dos custos operacionais ao reduzir a necessidade de codificação manual e retrabalho.
- **E4:** Segundo Prinz, Rentrop e Huber (2021), uma gestão eficiente de recursos garante que as ferramentas low-code sejam utilizadas de maneira otimizada, potencializando seus benefícios ao alinhar a disponibilidade de recursos com as necessidades do projeto.
- **E5:** Segundo Takahashi, Javed e Kohda (2024) o código legado analisado de forma criteriosa com planejamento cuidadoso, permite identificar possíveis conflitos e integrações necessárias, garantindo que a transição para uma plataforma low-code seja suave e que os benefícios esperados sejam alcançados. Ainda sobre a integração entre o low-code e código legado, há a oportunidade para as organizações modernizarem as interfaces de software, aprimorarem a funcionalidade e melhorarem a eficiência (TAKAHASHI; JAVED; KOHDA, 2024).
- **E6:** Trigo, Varajão e Almeida (2022) dizem que as ferramentas low-code são projetadas para simplificar a criação e modificação de código. Já Marques (2023) destaca as funcionalidades prontas para uso, que eliminam a necessidade de desenvolvimento de módulos centrais ao início de cada desenvolvimento de sistemas.
- **E7:** As plataformas low-code frequentemente utilizam designs modulares e componentes pré-desenvolvidos, facilitando a construção de uma arquitetura de sistema clara e bem organizada, que é fácil de entender e manter. Segundo Marques (2023) as PDLCs também simplificam e abstraem detalhes como os relacionados a configuração de infraestrutura e integridade de dados.
- **E8:** Khorram, Mottu e Sunyé (2020) ao realizarem uma análise das funcionalidades de teste das PDLCs, constataam que elas frequentemente incluem funcionalidades integradas para testes, permitindo a criação e execução de casos de teste de forma mais intuitiva e abrangente, o que melhora a qualidade do software.
- **E9:** Alamin et al. (2023) em seu levantamento sobre os tópicos de pesquisa de low-code, identificam que devido a todos os Stakeholders terem uma melhor visão sobre o que está sendo desenvolvido, eles também podem contribuir de forma fácil para a documentação dos sistemas criados com PDLCs. Também destacam que o uso de ferramentas visuais e componentes modulares em PDLCs facilita a geração de documentação compreensível, que é essencial para a manutenção e evolução do sistema.
- **E10:** Conforme explica Marques (2023) a modularidade está na essência das PDLCs, devido a disponibilização de funcionalidades prontas para que o desenvolvimento seja acelerado.

- **E11:** Devido a simplificação que Trigo, Varajão e Almeida (2022) identificam na criação de código das PDLs, temos como resultado também a fácil e prática revisão dos códigos criados e funcionalidades utilizadas, para uma manutenção rápida e assertiva.
- **E12:** Bock e Frank (2021), Phalake e Joshi (2021) e Marques (2023) entendem que devido a abstração de conceitos e a funcionalidades criadas apenas com ações de *arrastar e largar* o treinamento se torna mais rápido e com exemplos mais práticos sobre o resultado de cada ação realizada nas PDLs. Inclusive essa forma mais visual torna mais acessível o treinamento para o perfil de Desenvolvedores Cidadãos.

4.4 Passo 4: Determinar o escopo da teoria

Neste passo será necessário apresentar de forma bem específica o escopo de cada classe arquétipo. Entretanto quanto mais o escopo de validade for especificado, tornará a teoria aplicável a apenas poucos projetos de software. Portanto será buscado um meio termo no momento de definir o escopo, para que seja possível manter o objetivo de auxiliar a comunidade de desenvolvedores low-code e desenvolvedores cidadãos da forma mais abrangente possível. Então seguindo o que foi apresentado pelo Framework proposto por Sjøberg et al. (2008) e o contexto do trabalho desenvolvido até aqui, entende-se que o escopo da teoria é o seguinte:

A teoria deve ser aplicável a times de desenvolvimento que criam e modificam sistemas de informação baseados em código novo ou código legado.

4.5 Passo 5: Testando a teoria através de pesquisa empírica

Após toda a construção realizada nos passos anteriores, foi realizada uma pesquisa quantitativa, com a comunidade de desenvolvedores low-code, para ser avaliado se as proposições e explicações fornecidas na construção da teoria serão consideradas válidas ou não. O detalhamento dessa pesquisa será realizado durante a Seção 5.

4.6 Descrição da teoria

Realizados os 5 passos descritos no Framework de Sjøberg et al. (2008), é obtido como resultado a Figura 2, que é um diagrama formatado na notação proposta por Sjøberg et al. (2008). Essa notação foi escolhida, pois já foi utilizada em trabalhos largamente experimentados, discutidos e validados como o de Sjøberg et al. (2008) e de Sousa et al. (2018).

Nesse diagrama percebe-se que algumas proposições estão sendo representadas em negrito e outras não. Os resultados que embasam essa notação e a validação inicial das proposições serão apresentados na Seção 5.6.

5 AVALIAÇÃO

Esta seção apresenta os procedimentos adotados para a avaliação da teoria que foi proposta durante a Seção 4. Para isso a Seção 5.1 descreve o Objetivo e Questões de Pesquisa acerca da avaliação da teoria, a Seção 5.2 descreve o contexto de seleção dos participantes, a Seção 5.3 apresenta a construção do questionário utilizado, a Seção 5.4 apresenta o processo de avaliação, a Seção 5.5 apresenta a análise do perfil dos participantes e por fim a Seção 5.6 apresenta a avaliação das proposições.

5.1 Objetivo e Questões de Pesquisa

Esta avaliação essencialmente objetiva a proposição de uma teoria sobre o desenvolvimento de software utilizando PDLCS, para compreender os efeitos do método low-code no contexto de desenvolvimento de software. A validação inicial da teoria ocorreu com uma pesquisa quantitativa que teve como público alvo a comunidade de desenvolvedores de low-code, demais trabalhadores de áreas de TI e de negócio que atuem com low-code. O objetivo dessa avaliação é apresentado e baseado no modelo GQM, tal modelo mostrou-se eficaz na elaboração de objetivos, que pode ser visto nos seguintes trabalhos: (FARIAS; GARCIA; LUCENA, 2013) e (FARIAS et al., 2014). O resultado produzido pelo modelo é demonstrado abaixo:

avaliar a metodologia low-code
com a finalidade de propor uma teoria fundamentada empiricamente
no que diz respeito aos efeitos deste método
do ponto de vista do programador
no contexto de desenvolvimento de software

Particularmente, este estudo se concentra em elaborar uma teoria que possa servir como guia para os desenvolvedores e desenvolvedores cidadãos trabalharem com a tecnologia low-code. Com isso, o estudo se concentra na seguinte questão de pesquisa:

- **QP:** Como pode ser proposta uma teoria empiricamente fundamentada que auxilie os desenvolvedores e demais profissionais que utilizam o método low-code a entender seus efeitos no desenvolvimento de software?

5.2 Contexto de Seleção dos Participantes

Os participantes deste estudo foram selecionados com base em critérios rigorosos para garantir uma amostra rigorosa e representativa. Para cada participante foi solicitado a confirmar sua experiência com desenvolvimento de software e sua experiência com o método de desenvolvimento low-code para ser elegível para participar do estudo. Utilizamos amostragem por

conveniência. Sobre a amostragem e público-alvo, realizou-se a divulgação do questionário via internet, por redes sociais e aplicativos de mensagens, sempre buscando focar em grupos e páginas que possuam alta visibilidade para a comunidade de desenvolvedores e profissionais que atuam com o low-code. Dessa forma pode-se ter uma avaliação mais assertiva das proposições e explicações fornecidas pela teoria elaborada. A pesquisa foi realizada com 36 participantes que atuam em empresas brasileiras.

5.3 Questionário

A validação das proposições da teoria ocorreu com base na abordagem de pesquisa quantitativa por questionário, apresentada por Wainer et al. (2007). Dessa abordagem, para a avaliação das afirmações apresentadas ao respondentes, foi utilizada a escala Likert. O questionário foi construído utilizando a plataforma Google Forms e ele é apresentado, em sua íntegra, no Apêndice A.

O formulário de pesquisa foi dividido em duas partes, sendo a primeira de caracterização do perfil dos participantes e a segunda parte com a avaliação das proposições em si. No início do formulário, antes de iniciar propriamente a caracterização do perfil dos participantes, utilizou-se a pergunta "*Você possui experiência com o método de desenvolvimento Low-code?*", com as opções *Sim* e *Não*. Caso o respondente assinalasse que não possuía experiência a pesquisa era encerrada, evitando assim que houvesse respondentes fora do público desejado.

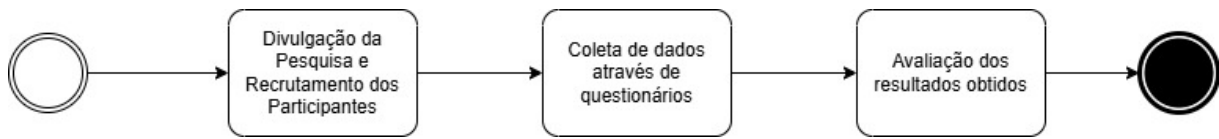
Para a caracterização do perfil dos participantes foram utilizados os critérios de idade, gênero, grau de escolaridade, formação acadêmica, tempo de estudo em universidade, cargo atual, tempo de experiência com desenvolvimento de software, e por fim, o tempo de experiência com o método de desenvolvimento low-code. A partir desses critérios foi possível entender de forma precisa quem eram os respondentes.

5.4 Processo de Avaliação

O processo de avaliação da teoria proposta sobre os efeitos do método de desenvolvimento low-code no desenvolvimento de software foi conduzido em três etapas principais: divulgação da pesquisa e recrutamento dos participantes, coleta de dados através de questionários, e avaliação dos resultados obtidos. Este capítulo detalha cada uma dessas etapas ilustradas na Figura 3, destacando os métodos utilizados e os resultados alcançados.

Etapas 1: Divulgação da pesquisa e recrutamento dos participantes. A primeira etapa do processo de avaliação envolveu a divulgação da pesquisa e o recrutamento dos participantes. A divulgação foi realizada através de diferentes canais de comunicação, incluindo redes sociais, fóruns de desenvolvedores e listas de e-mails específicas da comunidade de software. A mensagem de divulgação continha informações sobre o objetivo da pesquisa, a importância do estudo para a comunidade de desenvolvedores, e um convite para participar respondendo a um

Figura 3: Etapas da avaliação



Fonte: elaborado pelo autor

questionário online.

Para garantir a participação de um público diversificado e representativo, foram adotadas estratégias como a utilização de grupos de discussão em plataformas como LinkedIn, GitHub e Stack Overflow, além do contato direto com empresas e desenvolvedores conhecidos pela utilização de plataformas low-code. O critério de inclusão para os participantes era ter experiência prévia com o uso de plataformas low-code no desenvolvimento de software, o que assegurou que as respostas coletadas fossem relevantes e bem-informadas.

Etapa 2: Coleta de dados através de questionários. A segunda etapa consistiu na coleta das respostas através de um questionário estruturado. Conforme demonstrado na Seção 5.3, o questionário foi elaborado com base nas proposições teóricas formuladas no estudo e incluía perguntas fechadas para capturar dados quantitativos. As perguntas foram projetadas para realizar a caracterização do perfil dos participantes e também para avaliar as proposições elaboradas durante a construção da teoria.

Os questionários foram disponibilizados online utilizando a plataforma Google Forms, permitindo fácil acesso e resposta pelos participantes. A coleta de respostas ocorreu durante um período de 4 dias, garantindo tempo suficiente para a participação de um número significativo de desenvolvedores. Durante este período, lembretes periódicos foram enviados aos potenciais participantes para aumentar a taxa de resposta.

Etapa 3: Avaliação dos resultados obtidos.

A terceira e última etapa envolveu a análise dos resultados obtidos a partir das respostas dos questionários. A avaliação foi realizada através de análise estatística dos dados quantitativos. Esta análise é detalhada nas próximas subseções.

Para ilustração dos dados obtidos foram utilizadas funcionalidades de criação de tabela do próprio editor de texto Overleaf⁷ e da plataforma Draw IO⁸ para a construção de gráficos.

5.5 Análise do Perfil dos Participantes

Esta seção descreve os aspectos dos dados coletados respectivos ao perfil dos participantes do questionário de validação das proposições desta teoria.

Triagem. A amostragem de participantes utilizada foi a amostragem por conveniência. Todos os participantes foram convidados a participar da avaliação das proposições a partir da

⁷<https://pt.overleaf.com/>

⁸<https://app.diagrams.net/>

rede de contatos profissional do autor que inclui grupos de aplicativo de mensageria voltados para desenvolvedores de low-code, grupos de comunicação interna do ambiente profissional e através de divulgação em redes sociais do autor.

A partir da divulgação mencionada, o questionário (descrito na Seção 5.2) foi respondido por 48 participantes, porém somente 36 possuíam experiência com o método de desenvolvimento com low-code. Sendo assim, apenas esses respondentes tiveram o seu perfil caracterizado e puderam realizar a avaliação das proposições.

Perfil dos participantes. A Tabela 6, descreve o perfil dos participantes, a partir dos critérios de idade, gênero, grau de escolaridade, formação acadêmica, tempo de estudo em universidade, cargo atual, tempo de experiência com desenvolvimento de software, e por fim, o tempo de experiência com o método de desenvolvimento low-code.

Idade. Os participantes apresentam uma faixa etária variada, com uma concentração significativa nas faixas de 30 a 34 anos (22,2%) e 40 a 44 anos (22,2%). A faixa etária de 50 anos ou mais representa 16,7% dos participantes, mostrando que há uma participação relevante de profissionais mais experientes. As faixas de 25 a 29 anos (11,1%), 35 a 39 anos (13,9%), 20 a 24 anos (5,6%), e 45 a 49 anos (8,3%) completam o perfil etário, evidenciando uma diversidade de idades entre os respondentes.

Gênero. O gênero masculino predomina entre os respondentes, com 80,6%, enquanto o gênero feminino representa 19,4%. Essa disparidade pode refletir a predominância masculina historicamente observada nas áreas de tecnologia da informação.

Grau de Escolaridade. A grande maioria dos participantes possui Ensino Superior - Graduação (86%), seguido por Ensino Superior - Mestrado (5,6%), com uma pequena parcela tendo Ensino Técnico (2,8%), Ensino Superior - Pós Graduação (2,8%), e Ensino Superior - Doutorado (2,8%). Isso demonstra um alto nível de escolaridade entre os respondentes, não necessariamente essencial para a compreensão e uso de ferramentas de low-code, já que elas se propõem a facilitar o acesso ao desenvolvimento de perfis como o de Desenvolvedor Cidadão.

Formação Acadêmica. A formação acadêmica dos participantes é predominantemente em Análise de Sistemas (38,9%), seguida por Gestão de TI, Ciência da Computação e Sistemas de Informação, cada uma com 13,9%. Engenharia da Computação e Estatística representam 5,6% cada, enquanto Banco de Dados, Pedagogia e Relações Internacionais possuem 2,8% cada. Esta diversidade de formações acadêmicas pode indicar um amplo interesse e aplicação do desenvolvimento low-code em diferentes áreas.

Tempo de Aprendizado Acadêmico. O tempo de aprendizado acadêmico varia principalmente entre 5 a 6 anos (38,9%) e 2 a 4 anos (27,8%). Menos de 2 anos, 7 a 8 anos e mais de 8 anos têm representações menores, com 2,8%, 16,7% e 13,9%, respectivamente. Isso sugere que a maioria dos participantes tem uma base acadêmica sólida e um período considerável de estudo formal.

Cargo Atual. Os cargos atuais dos participantes incluem Analista (36,1%), Arquiteto (22,2%), Programador (19,4%), Gerente (13,9%), CTO (2,8%), Coordenador (2,8%) e Projetista (2,8%).

A predominância de cargos técnicos como Analista, Arquiteto e Programador mostra que a pesquisa ainda não conseguiu atingir usuários do low-code do lado das áreas de negócio das corporações.

Experiência com Desenvolvimento de Software. A experiência com desenvolvimento de software é predominantemente superior a 8 anos (61,1%), indicando que a maioria dos participantes possui vasta experiência na área. Outros 11,1% têm entre 2 a 4 anos e 5 a 6 anos de experiência, enquanto 8,3% têm menos de 2 anos e 7 a 8 anos de experiência. Esta experiência extensa é crucial para avaliar adequadamente as proposições sobre o desenvolvimento low-code.

Experiência com o Método de Desenvolvimento Low-Code. A experiência com o método de desenvolvimento low-code varia, com 38,9% dos participantes tendo menos de 2 anos de experiência, seguido por 27,8% com 2 a 4 anos e mais de 8 anos. Apenas 2,8% dos participantes possuem entre 5 a 6 anos e 7 a 8 anos de experiência com low-code. Isso pode indicar uma adoção relativamente recente desta tecnologia, alinhada com a popularização do termo low-code desde que foi proposto pela Forrester em 2014.

Os dados mostram que os participantes têm um perfil diversificado e altamente qualificado, com ampla experiência em desenvolvimento de software e uma base educacional robusta. A experiência variada com o método de desenvolvimento low-code reflete a evolução e a adoção crescente desta tecnologia desde a sua introdução formal no mercado.

5.6 Avaliação das Proposições da Teoria

Em um primeiro momento, o resultado das avaliações das proposições seria dividido favorável e não favorável. Porém como houveram resultados favoráveis em todas as proposições, elas foram reunidas na Figura 4, com os resultados obtidos. Avaliou-se todas como favoráveis, devido a todas as proposições obtiveram 50% ou mais de avaliação favorável na escala Likert, somando as avaliações *Concordo parcialmente* e *Concordo totalmente*. As proposições que atingiram 70% ou mais de avaliação favorável, foram classificadas como fortemente favorável.

P1. Fortemente favorável. A grande maioria dos participantes acredita que o treinamento afeta positivamente o uso do método de desenvolvimento low-code, com 94% de avaliações positivas.

P2. Fortemente favorável. 72% dos respondentes concordam que o método low-code melhora a comunicação no time de desenvolvimento.

P3. Fortemente favorável. 83% dos participantes consideram que o método low-code reduz custos devido ao aumento de produtividade.

P4. Fortemente favorável. A coordenação de recursos é vista positivamente, com 91% de aprovação.

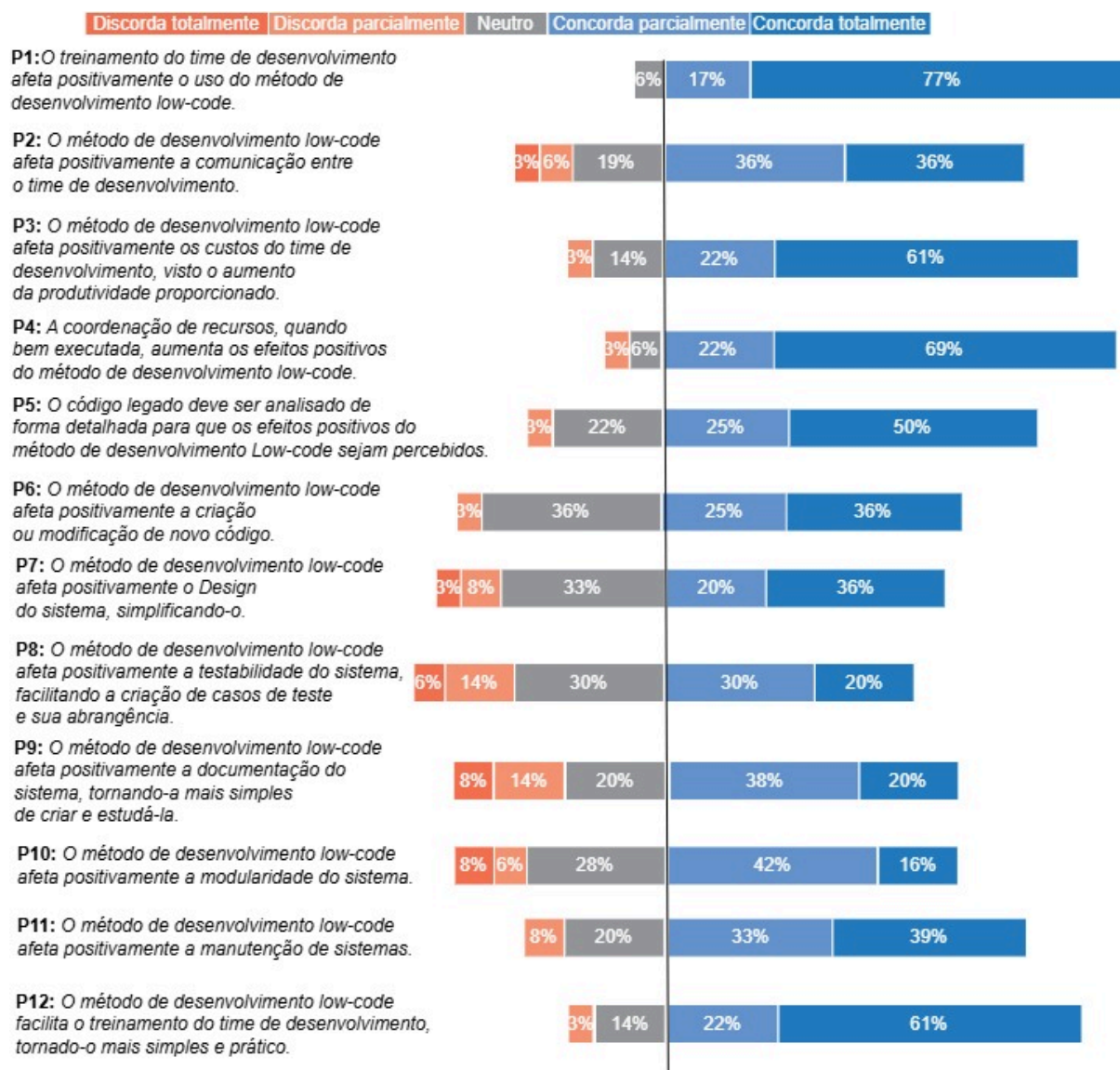
P5. Fortemente favorável. A análise detalhada do código legado é considerada essencial, com 75% de concordância.

Tabela 6: Dados do perfil dos participantes

Característica (n=36)	Resposta	Quantidade	Porcentagem
Idade	De 20 a 24 anos	2	5,6%
	De 25 a 29 anos	4	11,1%
	De 30 a 34 anos	8	22,2%
	De 35 a 39 anos	5	13,9%
	De 40 a 44 anos	8	22,2%
	De 45 a 49 anos	3	8,3%
	De 50 anos ou acima	6	16,7%
Gênero	Feminino	7	19,4%
	Masculino	29	80,6%
Grau de Escolaridade	Ensino técnico	1	2,8%
	Ensino Superior - Graduação	31	86%
	Ensino Superior - Pós Graduação	1	2,8%
	Ensino Superior - Mestrado	2	5,6%
	Ensino Superior - Doutorado	1	2,8%
Formação Acadêmica	Análise de Sistemas	14	38,9%
	Gestão de TI	5	13,9%
	Ciência da Computação	5	13,9%
	Sistemas de Informação	5	13,9%
	Engenharia da Computação	2	5,6%
	Estatística	2	5,6%
	Banco de Dados	1	2,8%
	Pedagogia	1	2,8%
Tempo de aprendizado acadêmico	Relações Internacionais	1	2,8%
	Menos de 2 anos	1	2,8%
	De 2 a 4 anos	10	27,8%
	De 5 a 6 anos	14	38,9%
	De 7 a 8 anos	6	16,7%
	Mais de 8 anos	5	13,9%
Cargo Atual	Analista	13	36,1%
	Arquiteto	8	22,2%
	Programador	7	19,4%
	Gerente	5	13,9%
	CTO	1	2,8%
	Coordenador	1	2,8%
	Projetista	1	2,8%
Experiência com desenvolvimento de software	Menos de 2 anos	3	8,3%
	De 2 a 4 anos	4	11,1%
	De 5 a 6 anos	4	11,1%
	De 7 a 8 anos	3	8,3%
	Mais de 8 anos	22	61,1%
Experiência com o método de desenvolvimento low-code	Menos de 2 anos	14	38,9%
	De 2 a 4 anos	10	27,8%
	De 5 a 6 anos	1	2,8%
	De 7 a 8 anos	1	2,8%
	Mais de 8 anos	10	27,8%

Fonte: elaborado pelo autor.

Figura 4: Dados coletados da avaliação das proposições da teoria



Fonte: elaborado pelo autor

P6. Favorável. 61% dos participantes acreditam que o método low-code favorece a criação e modificação de código.

P7. Favorável. O design do sistema é simplificado pelo low-code, com 56% de concordância.

P8. Favorável. 50% dos respondentes veem um impacto positivo na testabilidade do sistema.

P9. Favorável. A criação de casos de uso é positivamente afetada, com 58% de aprovação.

P10. Favorável. 58% dos participantes acreditam que o low-code facilita a documentação do sistema.

P11. Fortemente favorável. 72% dos participantes consideram que o low-code favorece a modularidade do sistema.

P12. Fortemente favorável. 83% dos respondentes consideram que o low-code facilita o treinamento do time de desenvolvimento.

A análise das proposições sobre o desenvolvimento low-code revela uma forte aceitação e reconhecimento dos benefícios desse método por parte dos respondentes. A maioria das proposições (7 de 12) são classificadas como fortemente favoráveis, indicando uma alta concordância quanto aos benefícios do desenvolvimento low-code em diversas áreas, como treinamento, comunicação, redução de custos, coordenação de recursos, análise de código legado, modularidade e treinamento do time.

As proposições restantes são classificadas como favoráveis, sugerindo que, embora haja uma percepção positiva, ainda há espaço para melhorias ou que a percepção não é tão fortemente unânime como nas outras áreas. As áreas de criação e modificação de código, design do sistema, testabilidade, criação de casos de uso e documentação do sistema são vistas de forma positiva, mas não alcançam o mesmo nível de forte aceitação.

6 DISCUSSÃO

Baseado nos resultados obtidos na pesquisa sobre as proposições, no Framework proposto por Sjøberg et al. (2008), replicado no trabalho de Sousa et al. (2018) e no trabalho de Luz, Pinto e Bonifácio (2019), há como resultado o diagrama da Figura 2, onde as proposições representadas com sua fonte em negrito foram as que analisamos como fortemente favoráveis e as demais apenas como favoráveis.

Além de haver como produto dos resultados da pesquisa a formulação do diagrama, pode-se evidenciar também a validação de características descritas por Luz (2021), Khorram, Mottu e Sunyé (2020) e Waszkowski (2019), que descrevem o low-code como rápido e ágil para o desenvolvimento web, de fácil acesso para os desenvolvedores mesmo com pouco treinamento, e de fácil geração de novo código.

Devido a falta de teorias relacionadas ao low-code, identificada na revisão da literatura, entende-se que a construção realizada nesse trabalho é um importante passo inicial para cen-

tralizarmos o conhecimento acerca dos benefícios entregues pelo low-code, bem como os seus pontos fracos.

6.1 Desafios e Implicações

Esta seção apresenta os desafios e implicações que foram derivados a partir da análise dos resultados dos dados coletados.

Desafio 1: Integração com Sistemas Legados. A integração de métodos low-code com sistemas legados pode apresentar desafios significativos. Conforme avaliado por Takahashi, Javed e Kohda (2024), os sistemas legados muitas vezes possuem arquiteturas complexas e documentação insuficiente, o que dificulta a implementação de soluções low-code. É essencial desenvolver estratégias específicas para avaliar a compatibilidade e integração eficiente, evitando riscos de segurança e inconsistências de dados.

Desafio 2: Segurança e Conformidade. Segundo o que foi apontado por Marques (2023) as plataformas low-code, por sua natureza, podem introduzir riscos de segurança se não forem gerenciadas adequadamente. Desenvolvedores podem não estar cientes das melhores práticas de segurança, o que pode resultar em vulnerabilidades, principalmente se for recordado que temos indivíduos não técnicos realizando desenvolvimento (Desenvolvedor Cidadão). Além disso, garantir que as soluções desenvolvidas estejam em conformidade com regulamentações e padrões de segurança é um desafio contínuo. Ferramentas e práticas de segurança robustas devem ser incorporadas para proteger dados sensíveis e garantir a conformidade legal.

Implicação 1: Aumento da Produtividade. Trigo, Varajão e Almeida (2022) evidenciam que o uso de plataformas low-code pode aumentar significativamente a produtividade dos desenvolvedores, em comparação a métodos baseados em código, permitindo que tarefas que antes demandavam semanas sejam concluídas em dias. Isso implica uma maior capacidade de resposta às necessidades de negócio e uma aceleração no lançamento de novas funcionalidades e produtos.

Implicação 2: Necessidade de Treinamento Contínuo. Apesar da simplicidade e facilidade de uso, o sucesso na implementação de plataformas low-code depende de um treinamento contínuo e atualizado. Desenvolvedores precisam estar constantemente atualizados sobre as novas funcionalidades e melhores práticas para maximizar os benefícios da plataforma. Em adição a isso Bock e Frank (2021) argumenta que cada PDLC possui uma dinâmica e funcionalidades próprias, então por isso é necessário também incentivar treinamentos relativos não somente ao low-code mas também a engenharia de software.

6.2 Limitações

Todo estudo científico possui suas limitações, e este trabalho sobre a teoria do desenvolvimento de software low-code não é exceção. Reconhecer essas limitações é crucial para uma

interpretação adequada dos resultados e para direcionar futuras pesquisas na área. Abaixo, destacam-se as principais limitações deste estudo:

Limitação 1: Amostra Restrita. A amostra de participantes foi composta por desenvolvedores que responderam voluntariamente ao questionário online. Embora a diversidade dos canais de recrutamento tenha buscado alcançar uma variedade ampla de desenvolvedores, não é possível garantir que a amostra seja completamente representativa de toda a comunidade de desenvolvedores low-code. A auto-seleção dos participantes pode introduzir um viés, já que aqueles que têm experiências mais positivas ou negativas com o low-code podem ter sido mais propensos a participar.

Limitação 2: Escopo Geográfico. A pesquisa foi majoritariamente conduzida em uma base a níveis de Brasil, mas sem um controle rigoroso sobre a distribuição geográfica dos participantes. Diferentes regiões podem ter variações significativas em termos de adoção e práticas de desenvolvimento low-code, influenciadas por fatores culturais, econômicos e tecnológicos. A falta de um controle geográfico mais detalhado pode limitar a generalização dos resultados para contextos específicos.

Limitação 3: Validação das Proposições. Embora as proposições teóricas tenham sido validadas através de uma pesquisa quantitativa, a validação foi limitada a percepções subjetivas dos participantes. A ausência de um estudo experimental controlado, onde as variáveis poderiam ser rigorosamente manipuladas e observadas, limita a capacidade de estabelecer relações causais definitivas entre o método low-code e os resultados observados.

7 CONCLUSÃO

Este trabalho apresentou um estudo empírico para a formulação de uma teoria sobre os efeitos do desenvolvimento de software baseado em low-code. Durante a revisão da literatura realizada, foi possível verificar a falta de trabalhos e autores que teorizem sobre o campo crescente do low-code. De toda forma percebe-se um cenário promissor, pois essa tecnologia cada vez mais está sendo utilizada como facilitadora de movimentos de transformação digital nas organizações (PHALAKE; JOSHI, 2021).

Os resultados deste estudo confirmam a eficácia das plataformas de desenvolvimento low-code em acelerar o ciclo de desenvolvimento de software e aumentar a produtividade dos desenvolvedores. Através da análise empírica realizada com a comunidade de desenvolvedores, foi possível observar a opinião favorável sobre a redução custo o desenvolvimento de aplicações, além de uma melhoria na comunicação e colaboração entre as equipes de TI e as áreas de negócio. Em resumo, a implementação do low-code se revela como uma estratégia promissora para empresas que buscam agilidade e inovação contínua em seus processos de transformação digital.

Para complementar este trabalho, alguns estudos futuros podem ser realizados: (1) Comparação com outros métodos de desenvolvimento de software; (2) Investigações adicionais sobre

o impacto do low-code na qualidade do software; (3) Estudo de caso mais profundo sobre o impacto econômico que demonstre economias de custo, aumento da produtividade e retorno sobre investimento (ROI). Este trabalho é um estudo inicial dos efeitos do desenvolvimento de software baseado em low-code, servindo de suporte para novos estudos mais aprofundados relacionados ao tema.

REFERÊNCIAS

- AL ALAMIN, M. A. et al. An empirical study of developer discussions on low-code software development challenges. In: IEEE/ACM 18TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p. 46–57.
- ALAMIN, M. A. A. et al. Developer discussion topics on the adoption and barriers of low code software development platforms. **Empirical software engineering**, [S.l.], v. 28, n. 1, p. 4, 2023.
- BOCK, A. C.; FRANK, U. In search of the essence of low-code: an exploratory study of seven development platforms. In: ACM/IEEE INTERNATIONAL CONFERENCE ON MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS COMPANION (MODELS-C), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p. 57–66.
- CHAPETTA, W. A. A influência de fatores na produtividade do desenvolvimento de software de acordo com um modelo de estruturas teóricas. , [S.l.], 2018.
- DAHLBERG, D. **Developer experience of a low-code platform: an exploratory study**. 2020.
- ENDRES, A.; ROMBACH, H. D. **A handbook of software and systems engineering: empirical observations, laws, and theories**. [S.l.]: Pearson Education, 2003.
- FARIAS, K. et al. Evaluating the effort of composing design models: a controlled experiment. **Softw Syst Model (2015) 14:1349–1365**, [S.l.], p. 14:1349—1365, May 2014.
- FARIAS, K.; GARCIA, A.; LUCENA, C. Effects of stability on model composition effort: an exploratory study. **Softw Syst Model (2014) 13:1473–1494**, [S.l.], p. 13:1473—1494, Jan. 2013.
- HEFFNER, R. et al. Predictions 2021: software development. **Forrester: Cambridge, MA, USA**, [S.l.], 2020.
- HENRIETTE, E.; FEKI, M.; BOUGHZALA, I. Digital transformation challenges. , [S.l.], 2016.
- KHORRAM, F.; MOTTU, J.-M.; SUNYÉ, G. Challenges & opportunities in low-code testing. In: ACM/IEEE INTERNATIONAL CONFERENCE ON MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS: COMPANION PROCEEDINGS, 23., 2020. **Proceedings...** [S.l.: s.n.], 2020. p. 1–10.

LEBENS, M. et al. Rise of the citizen developer. **Muma Business Review**, [S.l.], v. 5, p. 101–111, 2022.

LUZ, M. P. **Assinaturas digitais em aplicações low-code com recurso a fluxos de trabalho**. 2021. Tese (Doutorado em Ciência da Computação) — Universidade de Lisboa (Portugal), 2021.

LUZ, W. P.; PINTO, G.; BONIFÁCIO, R. Adopting devops in the real world: a theory, a model, and a case study. **Journal of Systems and Software**, [S.l.], v. 157, p. 110384, 2019.

MARQUES, M. G. **Comparação de plataformas low-code**. 2023. Tese (Doutorado em Ciência da Computação) — , 2023.

PHALAKE, V. S.; JOSHI, S. D. Low code development platform for digital transformation. In: INFORMATION AND COMMUNICATION TECHNOLOGY FOR COMPETITIVE STRATEGIES (ICTCS 2020) INTELLIGENT STRATEGIES FOR ICT, 2021. **Anais...** [S.l.: s.n.], 2021. p. 689–697.

PRINZ, N.; RENTROP, C.; HUBER, M. Low-code development platforms-a literature review. In: AMCIS, 2021. **Anais...** [S.l.: s.n.], 2021.

RICHARDSON, C. et al. New development platforms emerge for customer-facing applications. **Forrester: Cambridge, MA, USA**, [S.l.], v. 15, 2014.

RUBERT, M.; FARIAS, K. On the effects of continuous delivery on code quality: a case study in industry. **Computer Standards and Interfaces**, [S.l.], p. 103588, 2021.

SAHAY, A. et al. Supporting the understanding and comparison of low-code development platforms. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS (SEAA), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p. 171–178.

SANTOS, P. B. d. Engenharia sem código: as vantagens da linguagem low-code e no-code. , [S.l.], 2023.

SJØBERG, D. I. et al. Building theories in software engineering. **Guide to advanced empirical software engineering**, [S.l.], p. 312–336, 2008.

SOUSA, L. et al. Identifying design problems in the source code: a grounded theory. In: OF THE 40TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2018. **Proceedings...** [S.l.: s.n.], 2018. p. 921–931.

TAKAHASHI, N.; JAVED, A.; KOHDA, Y. How low-code tools contribute to diversity, equity, and inclusion (dei) in the workplace: a case study of a large japanese corporation. **Sustainability**, [S.l.], v. 16, n. 13, p. 5327, 2024.

TRIGO, A.; VARAJÃO, J.; ALMEIDA, M. Low-code versus code-based software development: which wins the productivity game? **IT Professional**, [S.l.], v. 24, n. 5, p. 61–68, 2022.

VIEIRA, R. D.; FARIAS, K. Usage of psychophysiological data as an improvement in the context of software engineering: a systematic mapping study. **SBSI'20: XVI Brazilian Symposium on Information Systems**, [S.l.], p. 1–8, Nov. 2020.

WAINER, J. et al. Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. **Atualização em informática**, [S.l.], v. 1, n. 221-262, p. 32–33, 2007.

WASZKOWSKI, R. Low-code platform for automating business processes in manufacturing. **IFAC-PapersOnLine**, [S.l.], v. 52, n. 10, p. 376–381, 2019.

APÊNDICE A – QUESTIONÁRIO DE ENTREVISTA

Figura 5: Questionário - Página 1

Low-code: efeitos no desenvolvimento de software

B *I* U  

Este questionário tem como objetivo avaliar proposições acerca do método de desenvolvimento low-code e seus efeitos.

E-mail *

E-mail válido

Este formulário está coletando e-mails. [Alterar configurações](#)

Nome completo *

Texto de resposta curta

Você possui experiência com o método de desenvolvimento Low-code? *

☐ Sim

☐ Não

Fonte: elaborado pelo autor

Figura 6: Questionário - Página 2

Apenas mais algumas perguntas sobre você e já vamos para a avaliação das proposições! ✕ ⋮

Descrição (opcional)

Idade *

☐ Até 19 anos

☐ De 20 a 24 anos

☐ De 25 a 29 anos

☐ De 30 a 34 anos

☐ De 35 a 39 anos

☐ De 40 a 44 anos

☐ De 45 a 49 anos

☐ De 50 anos ou acima

Gênero *

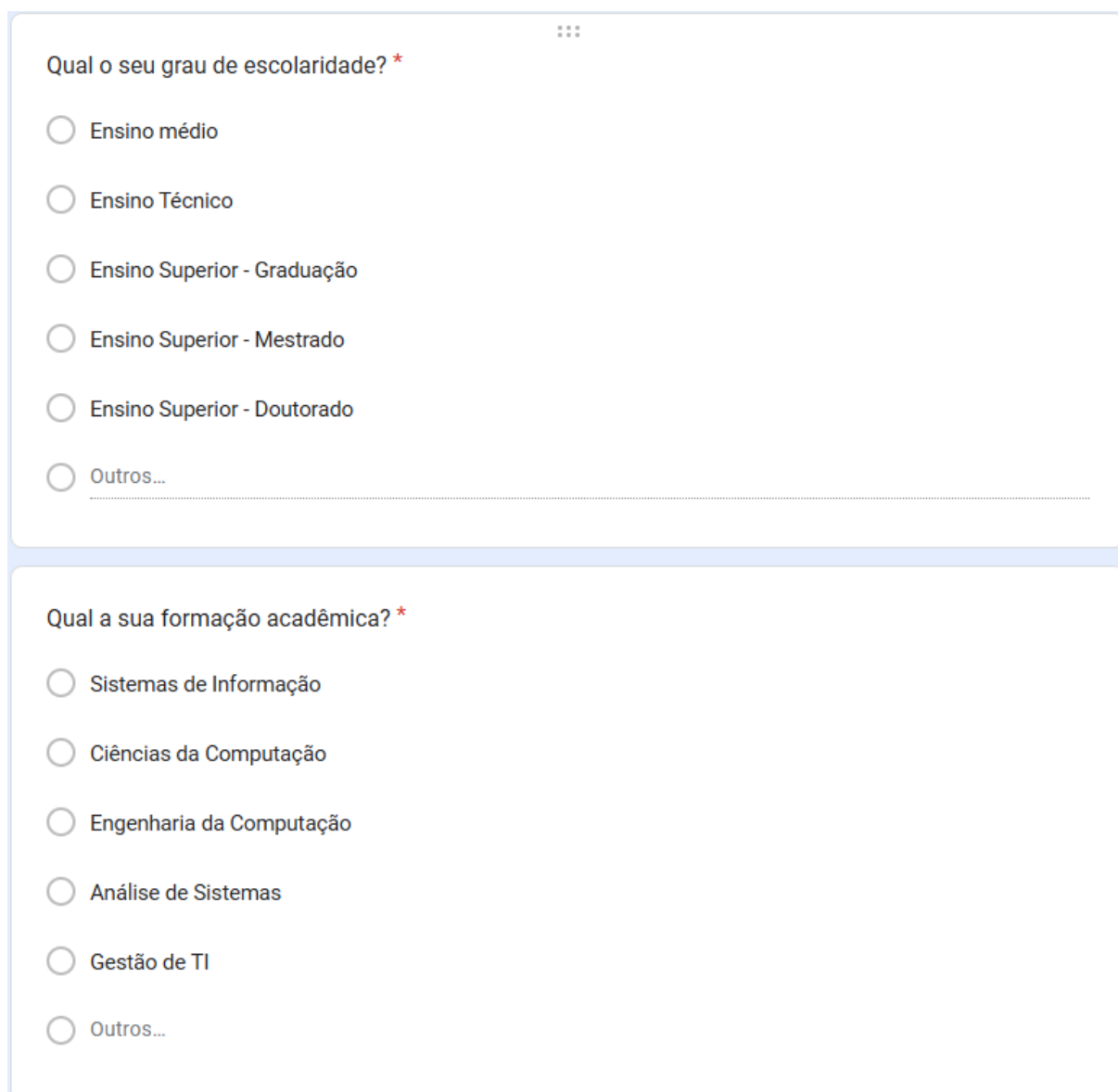
☐ Feminino

☐ Masculino

☐ Prefiro não dizer / Outro

Fonte: elaborado pelo autor

Figura 7: Questionário - Página 3



Qual o seu grau de escolaridade? *

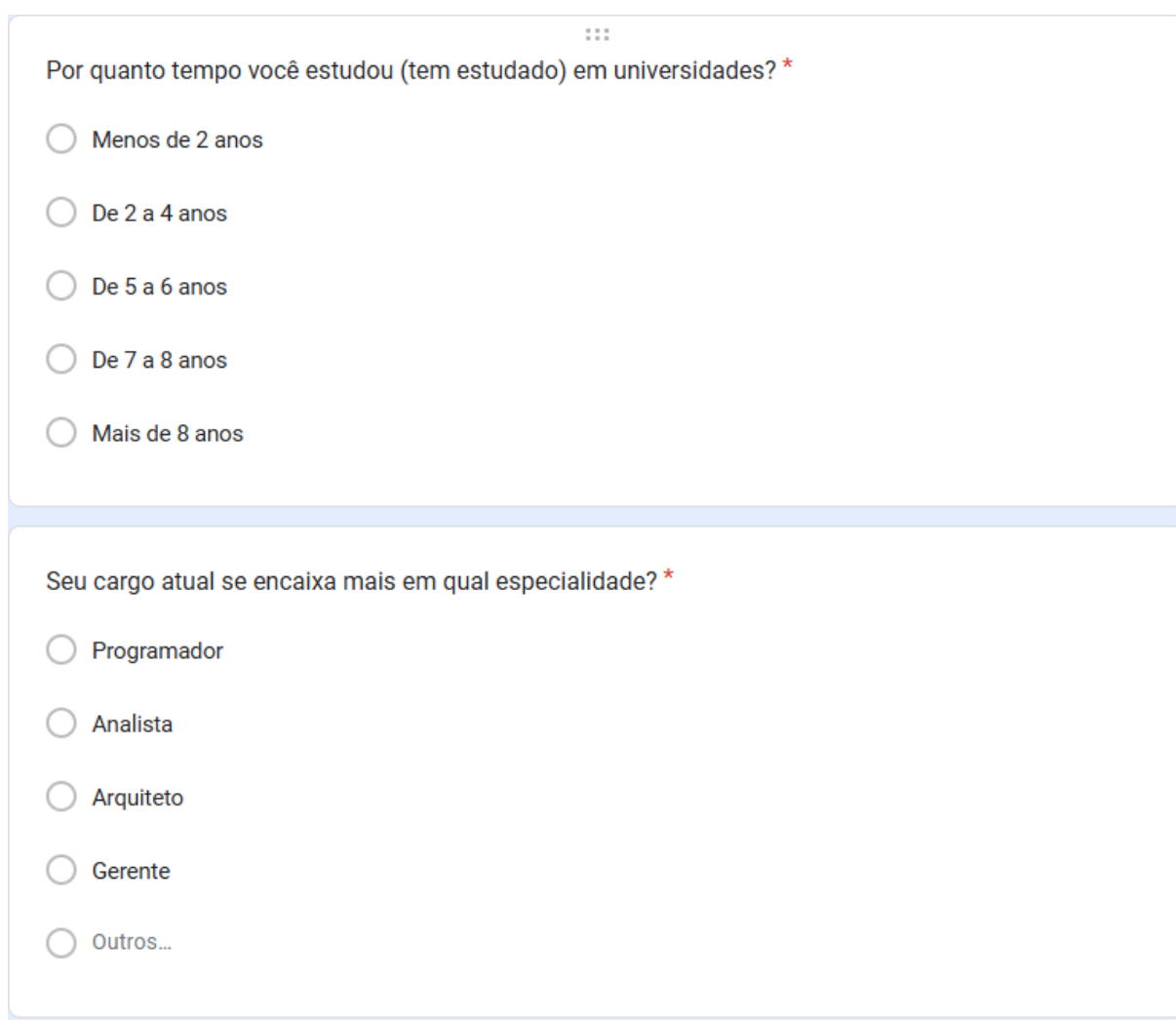
- ☐ Ensino médio
- ☐ Ensino Técnico
- ☐ Ensino Superior - Graduação
- ☐ Ensino Superior - Mestrado
- ☐ Ensino Superior - Doutorado
- ☐ Outros...

Qual a sua formação acadêmica? *

- ☐ Sistemas de Informação
- ☐ Ciências da Computação
- ☐ Engenharia da Computação
- ☐ Análise de Sistemas
- ☐ Gestão de TI
- ☐ Outros...

Fonte: elaborado pelo autor

Figura 8: Questionário - Página 4



...

Por quanto tempo você estudou (tem estudado) em universidades? *

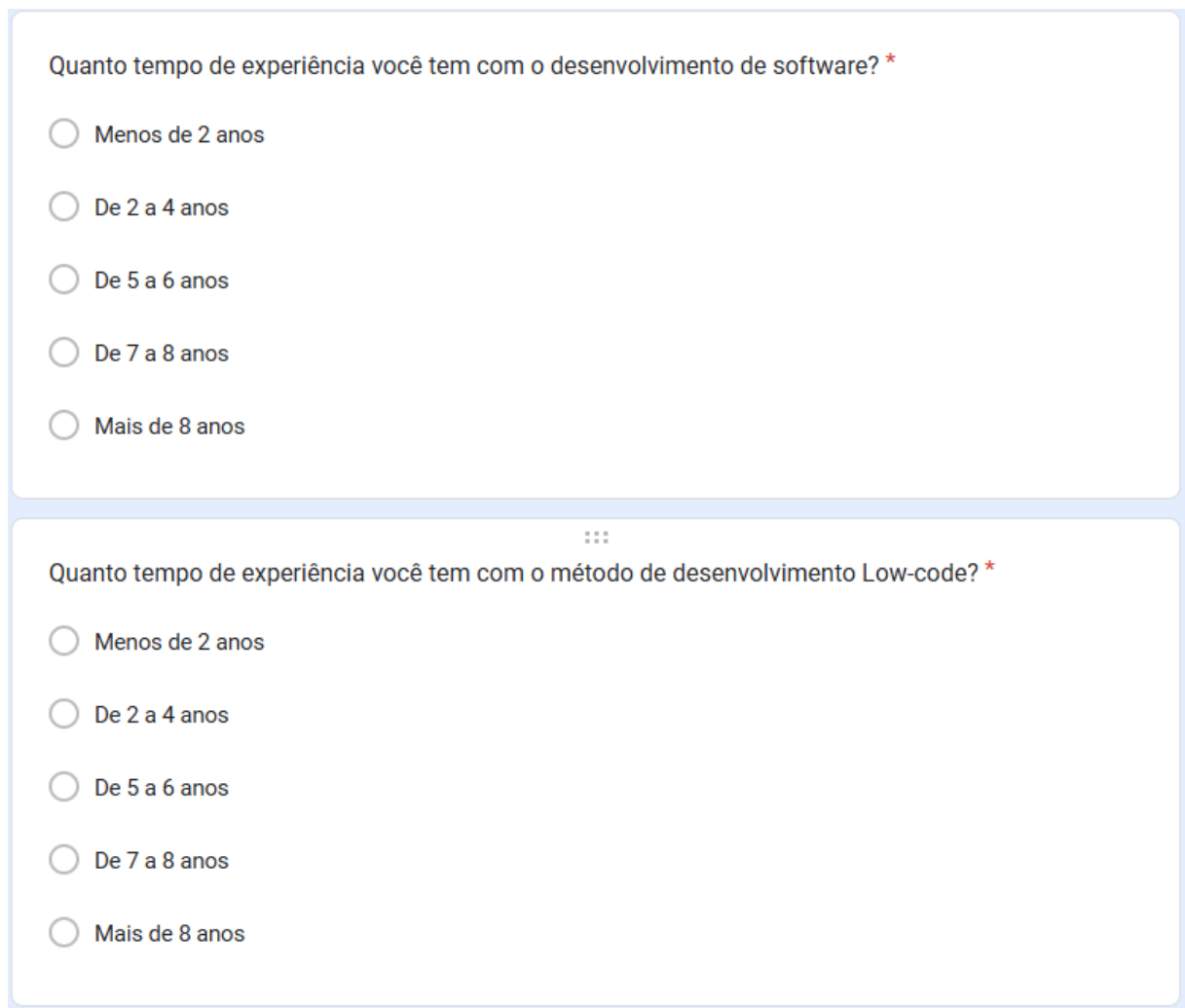
- ☐ Menos de 2 anos
- ☐ De 2 a 4 anos
- ☐ De 5 a 6 anos
- ☐ De 7 a 8 anos
- ☐ Mais de 8 anos

Seu cargo atual se encaixa mais em qual especialidade? *

- ☐ Programador
- ☐ Analista
- ☐ Arquiteto
- ☐ Gerente
- ☐ Outros...

Fonte: elaborado pelo autor

Figura 9: Questionário - Página 5



Quanto tempo de experiência você tem com o desenvolvimento de software? *

- ☐ Menos de 2 anos
- ☐ De 2 a 4 anos
- ☐ De 5 a 6 anos
- ☐ De 7 a 8 anos
- ☐ Mais de 8 anos

...

Quanto tempo de experiência você tem com o método de desenvolvimento Low-code? *

- ☐ Menos de 2 anos
- ☐ De 2 a 4 anos
- ☐ De 5 a 6 anos
- ☐ De 7 a 8 anos
- ☐ Mais de 8 anos

Fonte: elaborado pelo autor

Figura 10: Questionário - Página 6

Avaliação de proposições acerca do método de desenvolvimento Low-code

Peço que avalie as proposições a seguir de acordo com a sua experiência, classificando-as de 1 a 5, sendo o 1 "discordo totalmente" e o 5 "concordo totalmente"

P1: O treinamento do time de desenvolvimento afeta positivamente o uso do método de desenvolvimento low-code

1 2 3 4 5

Discordo totalmente ☐ ☐ ☐ ☐ ☐ Concordo totalmente

P2: O método de desenvolvimento low-code afeta positivamente a comunicação entre o time de desenvolvimento

1 2 3 4 5

Discordo totalmente ☐ ☐ ☐ ☐ ☐ Concordo totalmente

P3: O método de desenvolvimento low-code afeta positivamente os custos do time de desenvolvimento, visto o aumento da produtividade proporcionado

1 2 3 4 5

Discordo totalmente ☐ ☐ ☐ ☐ ☐ Concordo totalmente

Fonte: elaborado pelo autor

Figura 11: Questionário - Página 7

<p>P4: A coordenação de recursos, quando bem executada, aumenta os efeitos positivos do método de desenvolvimento low-code *</p>						
	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente
...						
<p>P5: O código legado deve ser analisado de forma detalhada para que os efeitos positivos do método de desenvolvimento Low-code sejam percebidos. *</p>						
	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente
...						
<p>P6: O método de desenvolvimento low-code afeta positivamente a criação ou modificação de novo código. *</p>						
	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente
...						
<p>P7: O método de desenvolvimento low-code afeta positivamente o Design do sistema, simplificando-o. *</p>						
	1	2	3	4	5	

Fonte: elaborado pelo autor

Figura 12: Questionário - Página 8

P8: O método de desenvolvimento low-code afeta positivamente a testabilidade do sistema, facilitando a criação de casos de teste e sua abrangência. *						
	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

P9: O método de desenvolvimento low-code afeta positivamente a documentação do sistema, tornando-a mais simples de criar e estudá-la.						
	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

P10: O método de desenvolvimento low-code afeta positivamente a modularidade do sistema. *						
	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

Fonte: elaborado pelo autor

Figura 13: Questionário - Página 9

⋮

P11: O método de desenvolvimento low-code afeta positivamente a manutenção de sistemas. *

1 2 3 4 5

Discordo totalmente ☐ ☐ ☐ ☐ ☐ Concordo totalmente

P12: O método de desenvolvimento low-code facilita o treinamento do time de desenvolvimento, tornando-o mais simples e prático. *

1 2 3 4 5

Discordo totalmente ☐ ☐ ☐ ☐ ☐ Concordo totalmente

Fonte: elaborado pelo autor