Universidade do Minho
Escola de Engenharia

Fernando Henrique Duarte Araújo

Desenvolvimento de Sistemas de Informação com Tecnologia Low-Code

Junho de 2022

Desenvolvimento de Sistemas de Informação com Tecnologia Low-Code | Fernando Henrique Duarte Araújo | UMinho

Universidade do Minho
Escola de Engenharia

Fernando Henrique Duarte Araújo
85706

Desenvolvimento de Sistemas de Informação com Tecnologia Low-Code

Dissertação de Mestrado
Mestrado integrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Doutor João Eduardo Quintela Varajão

Junho de 2022

DIREITOS DE AUTOR

AGRADECIMENTOS

Chegou ao fim mais uma etapa importante da minha vida. Sem dúvida que foram muitos os desafios, mas todos foram ultrapassados com sucesso e sempre bem acompanhado, porque a vida é sempre mais fácil pelo menos a dois, ou mais.

Em primeiro lugar gostava de deixar uma palavra de agradecimento ao meu orientador, o professor Doutor João Eduardo Quintela Alves de Sousa Varajão, pelo empenho, disponibilidade e dedicação que demonstrou ao longo de todo o trabalho. Sem dúvida que esta caminhada teria sido mais difícil sem este acompanhamento, para o qual há palavras suficientes para o descrever.

De seguida deixo uma palavra especial à minha família. A família é a base, o suporte, o porto seguro que vive de perto as nossas alegrias e vitórias, bem como as nossas tristezas e derrotas. A eles devo tudo e sem eles nada seria possível.

Por fim, e não menos importante, queria aqui destacar aquela que é a minha segunda família, os meus amigos. Os amigos são um tesouro que se tem, que se cria e que se alimenta. Tal como a família, conhecem de perto as nossas ambições e receios. A eles agradeço os momentos de trabalho, alegria, companheirismo e partilha.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio, nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Assinado por: **FERNANDO HENRIQUE DUARTE ARAÚJO**
Num. de Identificação: 15981907
Data: 2022.07.26 14:22:13+01'00'

RESUMO

Desenvolvimento de Sistemas de Informação com Tecnologia Low-Code

O mundo atravessa uma situação de grande imprevisibilidade, onde a mudança é considerada como algo fundamental e para a qual as organizações devem estar preparadas. As organizações estão cada vez mais pressionadas a responder rapidamente e com qualidade às necessidades e desejos dos seus clientes, os quais estão sempre em mutação. Muitas vezes, essas mesmas empresas, não têm a oportunidade nem a capacidade de contratar recursos qualificados que permitam uma resposta adequada.

A tecnologia *low-code* apresenta-se como uma tecnologia recente que, virtualmente, permite que qualquer pessoa seja capaz de desenvolver soluções de software, através da criação de modelos que são transformados em software funcional, sem ser necessário possuir formação e competências técnicas em programação.

Apesar dos benefícios e da importância reconhecida desta tecnologia, os trabalhos de natureza científica focados em perceber, por exemplo, a produtividade possibilitada pelas tecnologias *low-code*, são ainda escassos, justificando-se, não só a caracterização dos estudos realizados até ao momento sobre *low-code*, como também a explicação aprofundada dos elementos que constituem um projeto realizado com tecnologia *low-code*, incluindo as suas características, limitações, benefícios, etc.

Para explorar estes aspetos foi realizada uma revisão sistemática de literatura sobre a tecnologia *low-code* e realizado um estudo de caso de um projeto desenvolvido com recurso a tecnologia *low-code*.

Desse trabalho, em primeiro lugar, resultou um *framework* de desenvolvimento de sistemas de informação com tecnologia *low-code* com seis dimensões: a dimensão tecnologia, a dimensão projeto, a dimensão *developer*, a dimensão desafios, e a dimensão literatura. Com base no estudo de caso, foi possível suportar os elementos do *framework*.

Além disso, foi ainda possível identificar novos elementos que devem ser adicionados às várias dimensões que compõem o *framework*, tais como: os critérios para a avaliação do desempenho e do resultado final do projeto, a avaliação do sucesso, a gestão de projetos, etc.

Palavras-chave: Low-Code, Software, TI, SI, Desenvolvimento de Sistemas de Informação.

ABSTRACT

Information Systems Development with Low-Code

The world is going through a situation of great unpredictability, where change is considered as something fundamental and for which organizations must be prepared. Organizations are increasingly under pressure to respond quickly and with quality to the needs and desires of their customers, which are constantly changing. Often, these same companies do not have the opportunity or capacity to acquire qualified resources that allow for an adequate response to their needs.

Low-code technology presents itself as a recent technology which, virtually, allows anyone to be able to develop software solutions, through the creation of models that are transformed into functional software, without having the need for training and technical skills in programming.

Despite the benefits and recognized importance of this technology, scientific works focused on understanding, for example, the productivity made possible by low-code technologies, are still scarce, justifying not only the characterization of the studies carried out so far on low-code, as well as an in-depth explanation of the elements that make up a project carried out with low-code technology, including their characteristics, limitations, benefits, etc.

To explore these aspects, a systematic review of the literature on low-code technology was performed and a case study of a project developed using low-code technology was carried out.

This work, resulted in a comprehensive framework for the development of information systems with low-code technology with six dimensions: the technology dimension, the project dimension, the developer dimension, the challenges dimension and the literature dimension. Based on the case study, it was possible to support the elements of the framework.

Furthermore, it was also possible to identify new elements that should be added to the framework, such as: the criteria for evaluating the performance and the final result of the project, the evaluation of success, project management, etc.

Keywords: Low-Code, Software, IT, IS, Information Systems Development.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
DCR	Dynamic Condition Response
DSL	Domain-Specific Languages
IA	Inteligência Artificial
IDE	Integrated Development Environmental
IoT	Internet of Things
IS	Information Systems
IT	Information Technology
LCAP	Low-Code Application Platform
LCD	Low-Code Development
LCDP	Low-Code Development Platform
LCP	Low-Code Platform
MBSE	Model-Based Systems Engineering
MDSD	Model Driven Software Development
ML	Machine Learning
MVP	Minimum Viable Product
NCDP	No-Code Development Platform
NLP	Natural Language Processing
PaaS	Platform as a Service
SI	Sistemas de Informação
SPL	Software Product Line
TI	Tecnologias da Informação
VPL	Visual Programming Languages
WYSIWYG	What You See Is What You Get

1. INTRODUÇÃO

Neste primeiro capítulo é realizada uma contextualização do tema, focando a sua relevância e a importância da investigação. São, também, apresentadas as motivações, assim como os objetivos e a finalidade da dissertação. Por fim, o capítulo termina com a apresentação da estrutura do documento.

1.1. Enquadramento

Vivemos num mundo no qual se verifica uma mudança constante, não raras vezes imprevisível, sendo exemplo disso a pandemia COVID-19. Além do grau de incerteza associado a essa mudança, a mesma provoca transformações cada vez mais rápidas, que conduzem a novas tendências, novas necessidades e nova procura (Sahinaslan, Sahinaslan, & Sabancıoglu, 2021).

Para lidar com esta questão, as organizações são obrigadas a procurar respostas mais rápidas e mais resilientes, que sejam capazes de acompanhar as alterações nos mercados tendo a consciência de que o grau de complexidade das operações está a aumentar, o mesmo acontecendo com o número de relacionamentos estabelecidos entre diferentes entidades (Sanchis, García-Perales, Fraile, & Poler, 2020). Tal faz com que as empresas tenham de desenvolver a sua capacidade de previsão, antecipação e rapidez na entrega de produtos e serviços.

Outro aspeto a salientar está relacionado com as ações das organizações, dado que lhes é pedido para "fazerem mais com menos", o que dificulta a obtenção de novos recursos humanos, financeiros e materiais. Essas dificuldades traduzem-se na realocação interna de recursos, o que pode ser prejudicial se considerarmos que existem recursos críticos que devem estar focados em projetos de maior relevância e importância (da Cruz et al., 2021).

A capacidade competitiva é fundamental neste contexto (Khorram, Mottu, & Sunyé, 2020). As tecnologias da informação (TI), e mais concretamente as aplicações de software, são incontornáveis para potenciar a capacidade competitiva.

Tal requer novas soluções para o desenvolvimento de software, dado que a abordagem tradicional de codificação manual é vista como um processo demorado e bastante complexo (Alsaadi et al., 2021).

O *low-code* tem aqui um importante papel, dado que pode ser visto como um paradigma capaz de dar a resposta a estes desafios, permitindo fornecer rapidamente soluções tecnológicas, ajustadas às necessidades em evolução das organizações, incorporando facilmente o feedback dos utilizadores, reduzindo o tempo de desenvolvimento, e o custo associado.

1.2. Motivação e importância de investigação

O conceito *low-code* surgiu pela primeira vez em 2014, pela mão de analistas da *Forrester Research*. A partir desse mesmo momento, a sua popularidade tendo vindo a crescer de forma exponencial (da Cruz et al., 2021).

Tendo em conta a perspetiva da *Forrester*, em 2018, 23% dos *global players* de desenvolvimento de software usaram plataformas de desenvolvimento *low-code*. Em 2019, 22% dos responsáveis planeavam adotar essas plataformas dentro de um ano (Rymer & Koplowitz, 2019) e que, até ao final de 2021, 75% das organizações que se dedicam ao desenvolvimento de soluções de software usariam tecnologias low-code (Hurlburt, 2021).

Numa análise mais recente, a *Forrester* assegurou que em 2022 o mercado de *low-code* representaria cerca de 21 biliões de dólares (Al Alamin et al., 2021).

A Gartner prevê que, até 2023, mais de 50% das médias e grandes empresas adotem o *low-code* como uma das suas plataformas de desenvolvimento estratégico e que, em 2024, cerca de 65% das empresas irão utilizar o *low-code* como ferramenta de desenvolvimento de soluções tecnológicas (Alexander, 2021a; Di Ruscio, Kolovos, De Lara, Tisi, & Wimmer, 2021).

Segundo a Gartner, existe outro aspeto a salientar, nomeadamente, a falta de profissionais de TI. A procura por novos sistemas de informação irá aumentar cinco vezes mais rápido do que a capacidade que os departamentos de TI têm para os fornecer, devido à escassez de profissionais (Luo, Liang, Wang, Shahin, & Zhan, 2021).

Podemos, assim, concluir que a contratação e recrutamento de profissionais de TI, mais especificamente, engenheiros e profissionais de software, está cada vez mais difícil, devido à

elevada procura e à baixa oferta. As plataformas *no-code* e *low-code* permitem às organizações lidar com esta questão (Fryling, 2019).

A própria Forbes previu que as tecnologias *low-code* e *no-code* seriam a tendência mais disruptiva de 2021 e, mais uma vez, a Gartner definiu este tipo de abordagem de desenvolvimento de soluções de software sob o termo de *Platform as a Service* (PaaS) de alta produtividade (ElBatanony & Succi, 2021a; Pantelimon, Rogojanu, Braileanu, Stanciu, & Dobre, 2019).

De acordo com a perspetiva apresentada por um estudo realizado pela *451 Research*, uma empresa de consultoria e investigação de TI, o desenvolvimento de aplicações e soluções recorrendo a plataformas de *low-code*, pode potenciar uma redução de 50% a 90% no tempo de desenvolvimento, em comparação homóloga ao desenvolvimento com linguagens de codificação clássica (Sahinaslan et al., 2021).

Apesar da importância reconhecida a esta tecnologia, os trabalhos de natureza científica focados em perceber, por exemplo, a produtividade possibilitada pelas tecnologias *low-code*, são ainda escassos, justificando-se, não só a caracterização dos estudos realizados até ao momento sobre *low-code*, como também o estudo aprofundado dos elementos que constituem um projeto com tecnologia *low-code*, incluindo as suas características, limitações, benefícios, etc.

1.3. Finalidade do trabalho

Neste trabalho, é efetuado o estudo do estado da arte da tecnologia *low-code*, assim como é realizado o estudo de um caso de um projeto realizado com recurso a tecnologia *low-code*, com o objetivo de caracterizar com profundidade os projetos realizados com esta tecnologia.

Os resultados esperados são os seguintes: (1) Caracterização do estado da arte de trabalhos de natureza científica no que respeita a *low-code*; (2) Caracterização do estado da arte de trabalhos de natureza não científica no que respeita a *low-code*; (3) Estudo de um caso com utilização de tecnologia *low-code*.

1.4. Estrutura do documento

No primeiro capítulo da dissertação é realizado o enquadramento do tema, focando a motivação e importância de investigação. A finalidade do trabalho é ainda apresentada e, no final do capítulo, é descrita a estrutura do trabalho.

A segunda secção diz respeito à revisão de literatura. Num primeiro momento são definidas as fontes de dados e a estratégia de pesquisa, é efetuada a seleção dos artigos e detalhado o processo de extração e síntese de dados. Posteriormente, é apresentada uma síntese da literatura. A encerrar o capítulo é a realizada uma reflexão crítica sobre o estado da arte.

No terceiro capítulo é apresentado o *framework* construído com base na literatura analisada, bem como cada uma das suas dimensões: a dimensão tecnologia, a dimensão projeto, a dimensão *developer*, a dimensão desafios, e a dimensão literatura.

Na quarta secção, o método de investigação é definido. É feita uma descrição e justificação do método de investigação adotado, uma descrição e seleção do estudo de caso, é descrita a recolha de dados e a análise dos dados.

O capítulo cinco destina-se à apresentação e discussão dos resultados que se encontram divididos em três secções diferentes. No primeiro ponto, a apresentação dos resultados e a sua discussão são feitas à luz do *low-code* e da ferramenta *Genio*. Num segundo ponto, o foco passa a ser o sucesso dos projetos em geral envolvendo *low-code* e, em último lugar, abordado o projeto do estudo de caso.

Por fim, no capítulo seis são elencadas as conclusões e as implicações do trabalho. São também enumeradas as contribuições do trabalho e as suas limitações. Além disso, são apontadas as diferentes direções a seguir em projetos investigação futuros.

2. REVISÃO DE LITERATURA

Para a realização do processo de revisão de literatura foi definida primeiramente uma estratégia de pesquisa, a qual é apresenta pela primeira secção deste capítulo. Posteriormente, nas subsecções seguintes, são descritos detalhadamente os passos seguidos na realização da revisão.

2.1. Fontes de dados e estratégia de pesquisa

Numa fase inicial, antes da pesquisa estruturada de referências, foram efetuadas algumas consultas *ad-hoc*, utilizando diferentes bases de dados. O objetivo principal desta fase foi identificar um conjunto de artigos mais relevantes, incidindo principalmente em *journals*.

Após a pesquisa inicial, tendo em conta o reduzido número de resultados, o espetro da procura de fontes foi alargado, tendo sido considerados todos os tipos de documentação (por exemplo, artigos de conferência). Para realizar a revisão de literatura estruturada, recorreu-se a várias bases de dados, mais especificamente:

- Web of Science (www.webofknowledge.com);
- Scopus (www.scopus.com);
- IEEE Xplore (ieeexplore.ieee.org);
- ACM Digital Library (dl.acm.org).

Outras bases de dados, poderiam ter sido consideradas, no entanto as bases de dados utilizadas apresentam uma grande abrangência de conteúdo de natureza científica.

Após a análise e pesquisa inicial foi estabelecida a seguinte restrição: as expressões consideradas relevantes para a nossa revisão foram *low code*, *no code*, *low-code* e *no-code*.

No que diz respeito ao espaço temporal, não foi definida, nem considerada nenhuma restrição. A tabela 1 apresenta um resumo dos resultados obtidos com as pesquisas realizadas.

Tabela 1 - Resultados da pesquisa nas bases de dados

Bases de Conhecimento	Expressões	Resultados
Web of Science	“low code” or “low-code”	**223**
	“no code” or “no-code”	**323**
Scopus	(TITLE (“low code”) or TITLE (“low-code”))	**398**
	(TITLE (“no code”) or TITLE (“no-code”))	**510**
IEEE Xplore	“low code” or “low-code”	**165**
	“no code” or “no-code”	**77**
ACM Digital Library	“low code” or “low-code”	**588**
	“no code” or “no-code”	**1891**

As expressões de pesquisa foram formuladas usando as expressões consideradas relevantes. A estrutura de cada uma das expressões foi comum a todas as bases de dados.

Nos casos em que se verificaram resultados inviáveis de serem processados dado o elevado número de resultados (o caso de ACM Digital Library), foram analisamos apenas os primeiros 500 resultados, tendo em conta a ordem de relevância produzida pela base de dados.

Além destas publicações, foram também considerados documentos, relatórios e publicações de organizações ligadas ao setor, enquadradas no âmbito da literatura cinzenta. Assim sendo, foram selecionados dois artigos da *Forrester*, dois trabalhos da *Quidgest* e ainda duas publicações disponíveis no website da *OutSystems*.

2.2. Seleção de artigos

Todos os artigos foram selecionados para análise tendo em conta o seu título e as suas palavras-chave. Após essa seleção, o resumo dos artigos foi igualmente lido com o objetivo de acrescentar profundidade à análise.

No entanto, em alguns casos foi necessário realizar uma leitura integral das publicações para perceber se as mesmas deveriam ser integradas ou excluídas.

Foram selecionadas as publicações que cumprissem pelo menos um dos seguintes critérios:

- Apresentassem uma definição para o conceito *low-code* e/ou *no-code* ou outros termos equivalentes;
- Discutissem as vantagens e benefícios da tecnologia *low-code*;

- Abordassem as barreiras de adoção da tecnologia *low-code*;
- Discutissem os fatores de sucesso dos projetos realizados com tecnologia *low-code*;
- Explicitassem os principais impactos da tecnologia *low-code*;
- Discutissem as desvantagens e limitações da tecnologia *low-code*;
- Apresentassem as competências requeridas para a utilização da tecnologia *low-code*;
- Abordassem perspetivas futuras de evolução da tecnologia *low-code*;
- Explicitassem e utilizassem outros conceitos equivalentes e complementares do *low-code*, tais como, *no-code*, *Visual Programming Languages* (VPL), linguagens de modelação, *Model Driven Software Development,* etc.;
- Discutissem a produtividade da tecnologia *low-code*.

2.3. Extração e síntese de dados

A fim de alcançar os objetivos de investigação e para dar suporte à extração e análise de dados, foi construída a tabela disponível no apêndice 1, onde se estabelece uma relação entre as publicações selecionadas e os conceitos relevantes.

Também foi desenvolvida a tabela disponível no apêndice 2, com as questões de investigação ou objetivos ou problemas/oportunidades das publicações selecionadas, assim como, a tabela disponível no apêndice 3, com as principais conclusões da literatura realizada.

Na próxima secção é apresentado um resumo de cada uma das publicações consideradas.

2.4. Síntese da literatura

Segundo Sanchis et al. (2020), o *low-code* deve ser encarado como um promotor e catalisador da transformação digital, com o objetivo aumentar a capacidade de fornecer respostas rápidas às constantes mudanças de expectativas e desejos do mercado. Este tipo de tecnologia permite que pessoas com fraca ou mesmo inexistente experiência em codificação de aplicações, as possam desenvolver e implementar. Associado a esta tecnologia existe um conjunto de benefícios: (1) a privacidade, tendo em conta que o desenvolvimento das aplicações pode ser executado internamente, sem existir a necessidade de recorrer a

terceiros, tornando os projetos mais confidenciais; (2) a rapidez, uma vez que grande parte da configuração das aplicações é realizada com facilidades gráficas; (3) a redução de custos, que se encontra, segundo a visão dos autores, relacionada com o benefício anterior, onde uma redução no tempo de desenvolvimento se traduz numa redução de custos; (4) o envolvimento de perfis de negócio no desenvolvimento, que permite a alguns dos utilizadores finais tornarem-se responsáveis pelo desenvolvimento dos produtos e serviços; e (5) a minimização de requisitos instáveis e inconsistentes, como consequência da construção facilitada de produtos mínimos fiáveis que permite a validação das ideias de negócio e a incorporação do feedback dos utilizadores, evitando o desperdício de recursos. Sanchis et al. (2020) destacam, ainda, um conjunto de limitações: (1) a escalabilidade, uma vez que na sua opinião este tipo de tecnologia se destina a aplicações de menor dimensão; (2) a fragmentação tendo em conta que os paradigmas de desenvolvimento *low-code* estão dependentes do seu fornecedor e do seu modelo de programação exclusivo; (3) a diversidade das áreas dos responsáveis pelo desenvolvimento, que podem incluir os seus próprios formalismo e linguagens na codificação; e (4) a segurança, pois as aplicações são desenvolvidas por profissionais cuja área de especialização não está diretamente ligada, na maioria das vezes, ao desenvolvimento de software. A plataforma *Visual Factory Open Operating Systems (vf-OS)* apresentada surge para facilitar o processo de desenvolvimento de soluções e não ser dependente de software e fornecedores terceiros. Este *framework* multifacetado é capaz de gerir a rede de um ambiente colaborativo de produção e logística, que possibilita a comunicação e a interoperabilidade entre humanos, aplicações e dispositivos (*Internet of Things*). O carácter transversal da plataforma é algo que deve ser tido em conta no futuro. Por fim, expõem os próximos desafios a considerar: a integração de diferentes ambientes de desenvolvimento num único *Integrated Development Environmental* (IDE), o desenvolvimento de sistemas que auxiliem os responsáveis pelo desenvolvimento na construção das soluções, a integração da análise de código seguro, e a incorporação da tecnologia IoT nas plataformas *low-code* para responder à Indústria 4.0.

Dushnitsky and Stroube (2021) apresentam as soluções desenvolvidas com tecnologia *low-code*, como soluções baseadas em software onde a necessidade de escrever código manualmente é praticamente inexistente. Na última década verificou-se um aumento da disponibilidade destas plataformas, as quais vieram alterar a forma como os empreendedores

manuseiam os recursos humanos e financeiros. A título de exemplo, a plataforma *Shopify* permite ao utilizador comum desenvolver o seu negócio online e a sua plataforma sem que tenha de ter conhecimentos sobre codificação. Os autores focaram assim a sua atenção em dados que recolheram de *start-ups* de *e-commerce* lançadas na década de 2010, e principalmente, no impacto da *Shopify* no desempenho das *start-ups* que usam esta plataforma como base. Como principais resultados, e por comparação homóloga com outras *start-ups* concluíram, que as *start-ups* que utilizam como base a *Shopify* iniciam o seu percurso de vida com menos recursos humanos e financeiros. Em termos de recursos humanos, as empresas baseadas em *Shopify* possuem no primeiro ano, em média, 5.9 funcionários em tempo integral, contra os 7.6 funcionários em tempo integral das *start-ups* não *Shofify*. No que toca aos recursos financeiros, apenas 76% das *start-ups* que usam *Shopify* registam uma primeira ronda de financiamento, contra 86% de *start-ups* não *Shopify*. Além disso, as *start-ups Shopify* retêm também mais capital. Outros tópicos a destacar são: o financiamento total das *start-ups Shopify*, que é cerca de um terço do valor do financiamento das *start-ups* não *Shopify*, e o número atual de funcionários em tempo integral, que corresponde, nas *start-ups Shopify*, a metade do número atual de funcionários a tempo integral das *start-ups* não *Shopify*. Para concluir, os autores defendem também que a plataforma fornece um conjunto de funcionalidades bastante abrangente e que a disponibilização destas plataformas na última década contribuiu para o lançamento de mais negócios, uma vez que o custo de lançamento de soluções é mais baixo, o que tornou essa realidade bastante mais acessível a todas as pessoas.

da Cruz et al. (2021) definem *low-code* como um conceito emergente que transforma representações gráficas em software funcional, permitindo que qualquer pessoa possa assumir o papel de responsável pelo desenvolvimento de software, independentemente da sua formação. A crescente omnipresença de aplicações, impulsionada pela facilidade de acesso a um computador ou a um telemóvel, aumentam a procura de desenvolvimento de software para a construção de aplicações, o que conduz as organizações a encontrar formas de tornar o desenvolvimento de software mais rápido acessível e económico. Sempre que uma nova aplicação surge no mercado, é natural que sistemas semelhantes sejam criados e introduzidos como concorrentes. Contudo, a entrada destes concorrentes é feita, normalmente, muitos anos mais tarde, verificando-se um atraso na sua introdução no

mercado. Este atraso deve-se ao reconhecimento do mercado potencial, à alocação de recursos para o projeto, e ao tempo de desenvolvimento. O *low-code* pode ser encarado como um elemento capaz de proporcionar uma entrada mais rápida das soluções no mercado. No desenvolvimento de software puro, sendo uma tarefa complexa, podem surgir diversas dificuldades: (1) na criação de interfaces do utilizador intuitivas e responsivas; (2) na aplicação das regras de negócio; (3) na criação da equipa responsável pelo desenvolvimento da solução; e (4) na estimação do tempo total do projeto. O *low-code* ajuda a ultrapassar estes problemas. Além disso, o *low-code* é comparado com outros conceitos, nomeadamente, *Model Driven Software Development* (MDSD) e IDEs. Os autores consideram que o MDSD é semelhante ao *low-code*, uma vez que este se destina à geração automática de código fonte através de modelos. No entanto, existe uma característica que os distingue, que é o *low-code* funcionar como uma caixa preta onde os responsáveis pelo desenvolvimento não têm praticamente acesso a código-fonte, *frameworks*, linguagens de programação, etc. Já voltando o nosso olhar para os IDEs, o *low-code* distingue-se dos mesmos, na medida em que os IDEs servem apenas para simplificar a escrita de código. Os autores realçam alguns aspetos positivos: (1) a simplificação da construção de software complexo, permitindo que a organização se adapte rapidamente; (2) a curva de aprendizagem mais rápida em comparação com a codificação normal; (3) a capacidade de qualquer pessoa poder assumir o desenvolvimento; (4) a construção mais rápida de protótipos, que se traduz em feedback, e como consequência, uma melhor experiência do cliente; (5) a redução de custos de desenvolvimento. Por outro lado, destacam também os aspetos menos positivos a ter em conta: (1) a dificuldade de personalização da interface gráfica; (2) a menor complexidade de desenvolvimento de aplicações, que pode ser prejudicial nos casos em que as plataformas *low-code* são descontinuadas e o acesso ao código-fonte é dificultado; (3) a ausência de código-fonte, que no caso de uma má otimização do código por parte das plataformas *low-code*, impossibilita por vezes a correção dessas insuficiências; (4) o custo associado à utilização de plataformas *low-code*; (5) a dependência face a terceiros; (6) os riscos de segurança, uma vez que os responsáveis pelo desenvolvimento de software têm acesso a dados e informações confidenciais; (7) a vulnerabilidade das soluções causada pelos responsáveis pelo desenvolvimento, que têm pouca experiência e conhecimento, não prestando atenção a alguns pormenores tais como cifragem de palavras-passe, entre outros. Tendo por base as

plataformas *low-code* e o objetivo de desenhar e desenvolver uma plataforma *low-code* própria, é possível definir um conjunto de requisitos funcionais e não funcionais associadas às mesmas. Elencando primeiramente os requisitos não funcionais: (1) a facilidade de uso, devendo as plataformas ser intuitivas, permitindo que os utilizadores sejam capazes de desenvolver soluções com mínimo de formação e esforço; (2) a flexibilidade, podendo as plataformas acomodar casos de uso avançados; (3) a extensibilidade, onde as plataformas devem fornecer ferramentas para que os utilizadores possam limar e ajustar funcionalidades; (4) a interoperabilidade, permitindo a integração com sistemas externos; (5) a segurança, com o fornecimento de mecanismos de segurança fáceis de aplicar; (6) a privacidade dos dados e das informações; (7) a escalabilidade, permitindo, quer o escalonamento vertical quer horizontal, adaptando-se a procuras cada vez maiores; (8) a manutenção, o que significa que a plataforma desenvolvida deve ser construída de forma a que a sua manutenção seja fácil; (9) a compatibilidade com versões anteriores, permitindo diferentes atualizações.

Além dos requisitos não funcionais, são também mencionados os requisitos funcionais fundamentais. Por fim, da Cruz et al. (2021) defendem que o próximo passo do *low-code* é o desenvolvimento de software com auxílio a soluções de inteligência artificial (IA), onde os onde os utilizadores expressam verbalmente o que pretendem e o desenvolvimento do software é feito de forma automática.

Sahay, Indamutsa, Di Ruscio, and Pierantonio (2020) afirmam que as plataformas *low-code* são ambientes gráficos fáceis de utilizar e estão cada vez mais a ser introduzidas e promovidas com o objetivo de qualquer pessoa conseguir desenvolver e construir software. Normalmente, estas plataformas são fornecidas através da *cloud*, como PaaS. Por vezes, a tomada de decisão sobre qual a plataforma de *low-code* a utilizar, assumem os autores, pode ser uma tarefa difícil e desafiadora. Por outro lado, declaram que as *low-code platforms* (LCP) vêm ajudar a lidar com a escassez de profissionais de TI e permitem que os responsáveis pelo desenvolvimento fiquem isentos da responsabilidade de gerir manualmente aspetos técnicos, focando a sua atenção na aplicação das regras e lógica de negócio. As fases típicas de desenvolvimento de aplicações através de LCP identificadas são: (1) a modelação de dados, com a configuração da estrutura de dados, estabelecendo entidades, relacionamentos, restrições, dependências; (2) a definição da interface do utilizador, através da configuração de formulários e páginas, da gestão de funções do utilizador e mecanismos de segurança; (3)

a especificação de regras de negócio e fluxos de trabalho; (4) a integração de serviços externos, recorrendo a *Application Programming Interfaces* (APIs) de terceiros; e (5) a implementação da aplicação. É possível, ainda, definir um conjunto de áreas que devem ser consideradas na seleção e comparação as LCP, sendo elas: (1) a interface gráfica do utilizador, onde são fornecidas funcionalidades essenciais para suportar as interações com o cliente; (2) o suporte de interoperabilidade com serviços externos e fontes de dados; (3) o suporte de segurança das aplicações desenvolvidas através da LCP selecionada; (4) o suporte ao desenvolvimento colaborativo, permitindo a interação de utilizadores localizados em áreas geográficas distintas; (5) o suporte à reutilização fornecido por cada LCP, que pode possibilitar a reutilização de artefactos já desenvolvidos; (6) o suporte à escalabilidade, criando condições para adaptar as soluções desenvolvidas tendo em conta diferentes necessidades; (7) os mecanismos de especificação de regras de negócio; (8) os mecanismos de construção das soluções; e (9) o suporte à implantação das soluções. Um último tópico a ser considerado é a relação existente das LCP com metodologias de desenvolvimento como *Agile* e *Scrum*.

Tal como defendem outros autores, Al Alamin et al. (2021) consideram que o desenvolvimento *low-code* é um conceito emergente, que combina código-fonte mínimo com interfaces gráficas para promover o rápido desenvolvimento de aplicações. O *low-code software developmen*t (LCSD) é um novo paradigma que permite a criação de aplicações de software com codificação manual mínima, com recurso a interfaces gráficas e design baseado em modelos. Alguns dos benefícios associados, do seu ponto de vista, prendem-se com: (1) a flexibilidade e a agilidade; (2) o menor tempo de desenvolvimento que permite uma resposta mais rápida à procura e necessidades do mercado; (3) a redução de erros das aplicações; (4) a manutenção mais facilitada; (5) a incorporação do feedback do utilizador na aplicação a desenvolver, devido à facilidade de construção de produtos mínimos fiáveis. Os autores salientam o *Stack Overflow*, uma plataforma popular entre os responsáveis pelo desenvolvimento, com o propósito de encontrar soluções e diferentes caminhos para a resolução de problemas técnicos sendo visíveis bastantes publicações com discussões sobre os problemas técnicos associados ao desenvolvimento de soluções com recurso às plataformas LCSD. Grande parte das publicações abrangem quatro categorias principais: (1) a personalização; (2) a adoção de plataformas; (3) a gestão de base de dados, e (4) a integração de terceiros. De todas categorias, aquela se assume maior importância é a personalização,

com 40% dos problemas técnicos publicados, seguindo-se a gestão de base de dados com 33%, a adoção de plataformas com 22%, e a integração de terceiros com 17%. Isto deve-se ao facto de os recursos das plataformas LCSD assentarem em padrões, e qualquer customização desses recursos que não são diretamente suportadas pelos padrões disponíveis nas mesmas, torna-se num desafio. Grande parte das questões levantadas prende-se com a fase de desenvolvimento das soluções. No entanto, existe ainda um número considerável de questões ligadas aos desafios que surgem com os testes automatizados das soluções. Por último, é ainda evidenciada a relação existente entre a metodologia ágil e as LCP, uma vez que o objetivo principal de ambas é a satisfação do cliente e a entrega incremental contínua.

Sahinaslan et al. (2021) consideram que as *low-code development platforms* (LCDPs) são ferramentas de desenvolvimento de aplicações mais rápidas, que exigem pouco conhecimento em programação em comparação com as abordagens tradicionais, onde através de uma interface gráfica e ferramentas auxiliares é possível construir aplicações. Estas plataformas são fundamentais para lidar com a dependência da tecnologia, a transformação digital, a IoT, a indústria 4.0, mudanças estas que obrigam as empresas a rever os seus modelos de negócio e a se adaptar à procura com a qualidade e a velocidade desejadas. As LCDP remontam às linguagens de quarta geração, existentes nos anos 1990 e 2000. As LCDP são baseadas no design orientado por modelos, geração automática de código e princípios de programação gráfica. Um dos seus objetivos, referem os autores, é envolver os utilizadores finais na resposta às suas necessidades de desenvolvimento de software. Os principais motivos pelas quais as LCDP têm ganho espaço de adoção à codificação tradicional são: (1) a transparência, tendo em conta que é acessível a todos e fornece um ambiente de desenvolvimento de aplicações direto ao utilizador; (2) a escalabilidade, sendo as plataformas desenvolvidas a partir de uma perspetiva global com recursos mais escalonáveis; (3) a velocidade de desenvolvimento mais rápida; (4) a segurança, que é da responsabilidade da plataforma, não do responsável pelo desenvolvimento; (5) a experiência do utilizador, com o fornecimento de ferramentas de design de páginas gráficas para o utilizador, com diferentes temas que podem ser utilizados. É, ainda, destacada pelos autores a LCDP *SetXRM*, utilizada quer por empresas do setor público como também do setor privado, que contém recursos tais como: (1) design de módulos e relatórios ilimitados; (2) validação e desenho das regras e fluxos de negócio; (3) notificações e modelos de formulários, etc.

Butler (2020) começa por efetuar uma caracterização das plataformas *low-code* recorrendo à definição divulgada por um analista da *Forrester*, que defende que as LCP são plataformas que proporcionam o rápido desenvolvimento e a rápida entrega de aplicações com pouca ou residual codificação manual, envolvendo um investimento inicial em configuração, formação e implantação igualmente reduzidos. Ao contrário do desenvolvimento padronizado em cascata tradicional, estas plataformas focam-se na realização de testes e na aprendizagem contínua que daí advém, é dada uma maior atenção à investigação por contraponto com o desenvolvimento, e são utilizadas práticas de arquitetura mínimas. O sucesso das soluções desenvolvidas é mensurado pelos resultados obtidos pelo cliente e a adoção destas práticas agiliza a entrega das soluções através da disponibilização de ferramentas gráficas para rápida criação de formulários. Normalmente as LCP são encaradas como ferramentas que permitem a criação de aplicações por utilizadores com poucas ou nenhumas competências de programação. É referido pelo autor que grande parte das organizações está a adotar as LCP. No que toca à tarefa de estimativa de um projeto com *low-code*, devem ser considerados três aspetos: (1) tamanho; (2) esforço, e (3) cronograma. Em linguagens clássicas, o tamanho normalmente é medido tendo em conta as linhas de código ou os pontos de função, sendo este elemento fundamental para o posterior cálculo do esforço e do cronograma. No entanto, algumas LCP não fornecem acesso ao código fonte, o que torna a tarefa de estimar mais difícil, impossibilitando em alguns casos o uso de metodologias tradicionais, podendo criar impactos no sucesso do projeto. Após a análise de um conjunto de metodologias existentes aplicadas a um projeto hipotético, foi possível concluir que três apresentaram estimativas bem-sucedidas, sendo elas: COCOMO II, CORADMO e *Work Breakdown Structure*. O autor considera que a metodologia mais eficaz para a estimativa de um projeto *low-code* será semelhante à análise de pontos de função, que considere um conjunto de elementos, tais como: número de entradas do utilizador, número de conexões externas, complexidade do fluxo de trabalho, personalizações, etc.

Waszkowski (2019) apresenta as plataformas *low-code* como um conjunto de ferramentas destinadas a vários públicos, desde programadores profissionais, a pessoas que não possuem nenhum tipo de formação em codificação. As plataformas permitem a entrega rápida de soluções, com o mínimo de esforço depositado na codificação tradicional, bem como com um mínimo de esforço colocado na instalação, configuração, formação e implementação das

soluções. Assume que este paradigma de desenvolvimento pode dar resposta às necessidades das organizações de uma forma mais rápida e mais barata, que necessitam de acompanhar as rápidas mudanças verificadas no mercado, nomeadamente, dos seus concorrentes, fornecedores, etc. Este tipo de tecnologia é considerado como uma técnica de programação de quarta geração que utiliza interfaces gráficas para projetar e produzir aplicações, permitindo ao programador focar a sua atenção na solução, no projeto, nas funcionalidades, colocando de lado aspetos mais técnicos como a sintaxe de programação, minimizando o tempo gasto na resolução de problemas e na implementação. Por outo lado, tal como foi mencionado anteriormente, permite rapidamente implementar e introduzir mudanças, para além de ser possível incorporar mais facilmente a perspetiva do cliente na solução que está a ser desenvolvida. O autor apresenta o sistema de gestão de processos de negócio *AureaBPM*, que fornece suporte à modelação, automação, gestão e otimização dos processos de negócio, permitindo efetuar uma gestão integrada dos processos de negócio através de um repositório central, o acesso remoto dos utilizadores, sistemas de segurança, modelação de processos de negócio, etc.

Kukesh and Brandenburg (2018) defendem que a colaboração entre as organizações e as TI é fundamental para que a organização consiga alcançar o sucesso. Com o objetivo de fornecer aplicações, cada vez mais e de forma mais rápida, o processo iterativo das metodologias ágeis torna-se o mais adequado. Estas metodologias permitem o desenvolvimento iterativo e permitem às organizações fornecer dados e moldar o produto, ao longo do processo, antes de este chegar ao seu ponto de desenvolvimento final. Esta forma de desenvolvimento enquadra-se nas plataformas *low-code*, sendo que neste artigo é mencionada a plataforma *Mendix*, que permite o rápido desenvolvimento de formulários, abrangendo várias camadas de desenvolvimento de aplicações, ocultando os componentes mais técnicos. O foco de quem utiliza a plataforma é mais direcionado para a conceção da solução ao invés de focar a atenção na aprendizagem e na sintaxe de uma linguagem de programação tradicional.

Segundo Arora, Ghosh, and Mondal (2020), o desenvolvimento de aplicações recorrendo ao método tradicional, um modelo desenvolvimento *full-stack*, é um método já ultrapassado. As *low-code application development platforms* (LCADP) contribuem para que as organizações possam criar soluções de software com o mínimo de codificação manual. Os autores

apresentam uma LCADP que fornece uma interface gráfica que permite o desenvolvimento de aplicações de grande dimensão, sendo que qualquer pessoa a pode usar para o desenvolvimento de soluções, de nível empresarial, com um conjunto de vantagens associadas: (1) a velocidade ou rapidez de desenvolvimento, (2) a redução de custos associados à construção das soluções, (3) um melhor rendimento do investimento. Tendo em conta as pressões que se estão a verificar sobre os responsáveis pelo desenvolvimento de aplicações e software, devido à digitalização massiva que estamos a assistir nos últimos anos, as LCADP tornam-se essenciais para dar resposta a estas necessidades, permitindo a construção de aplicações de forma bastante rápida e a um custo reduzido. A plataforma apresentada pelos autores, a *Sagitec Software Studio* (S3), oferece um ambiente de desenvolvimento que permite ao utilizador criar aplicações de nível empresarial com uma quantidade muito reduzida de codificação. Esta plataforma foi desenvolvida sobre um *framework*, o *Sagitec Framework*, que contém uma arquitetura multicamada e os seus serviços estão ocultos à visão do utilizador, permitindo que este se concentre na solução a desenvolver e no negócio. A *Sagitec Software Studio* (S3) é utilizada sobretudo para criar, modificar e implementar aplicações Web. A quantidade de código desenvolvido manualmente, usando a plataforma durante o desenvolvimento de uma solução, decresce em cerca de 70% em comparação com os métodos tradicionais. Além desta plataforma, é apresentada a *Sagitec Test Studio* (STS) e a *NeoAnalytics*. A primeira tem como objetivo reduzir o número de casos de teste a serem executados manualmente, reduzindo o ciclo de desenvolvimento, aumentando a velocidade de execução do teste, e a cobertura do teste fornecendo relatórios fáceis e robustos. A segunda, preocupa-se com a deteção de fraude e é facilmente adaptada sem programação de software complexa.

Pichidtienthum, Pugsee, and Cooharojananone (2021) apresentam as plataformas *low-code* como ferramentas que agilizam e tornam o desenvolvimento de software mais rápido, tendo em conta a reduzida programação manual, sendo o desenvolvimento realizado com o auxílio de uma interface gráfica. Os responsáveis pelo desenvolvimento não são limitados pelos conhecimentos e experiências técnicas, uma vez que qualquer pessoa, com ou sem competências em programação, pode construir soluções. Outras vantagens associadas são: (1) a rápida disponibilização de uma solução para o cliente; (2) a capacidade de focar a atenção dos responsáveis pelo desenvolvimento na solução e não em aspetos técnicos; (3) a

capacidade das plataformas conectarem diferentes bases de dados. Os autores sugerem que as LCP podem não ser adequadas a todo o tipo de casos, sendo que em processos mais complexos a sua utilização pode ficar comprometida. Neste estudo foi abordado o software ERP *Odoo open-source*. O desenvolvimento de software realizado com auxílio desta plataforma requer algum tempo de aprendizagem, dada a complexidade do *framework*. Tendo em conta esta questão, foi proposta uma plataforma *low-code* (*Module Generator*) que auxilia os utilizadores na criação das soluções. Como parte integrante do estudo, foi realizado um teste para comparar o *Module Generator* com o método de codificação tradicional. Foi possível concluir que houve uma redução de 20% no tempo gasto para a criação de um módulo. O uso do *Module Generator* permite também que os responsáveis pelo desenvolvimento se concentrem na construção dos módulos, ao invés de se focarem na codificação, e que utilizadores pouco experientes possam usar a plataforma para desenvolvimento de um módulo.

Bock and Frank (2021a) acreditam que o sucesso alcançado nos últimos anos pelas plataformas de desenvolvimento *low-code* se deve aos ganhos de eficiência que advém da sua utilização. A limitação de mecanismos de codificação tradicional em detrimento do design e especificação são uma das características destas plataformas. Associadas a estas plataformas, os autores elencam um conjunto de vantagens tais como: (1) a redução significativa de custos de desenvolvimento de software; (2) são úteis para os profissionais de software e para utilizadores não familiarizados com linguagens de programação, permitindo que estes últimos sejam capazes de criar aplicações; (3) o aumento da produtividade; (4) a adaptabilidade e o empoderamento dos utilizadores; (5) a minimização da codificação. Os autores efetuam uma análise de sete LCDP disponíveis no mercado. Como resultado, afirmam que as tecnologias que compõem estas plataformas não são totalmente novas nem inovadoras, como, por exemplo, os componentes de modelação de dados, modelação de processos, sistemas de gestão de base de dados, etc. Um ponto que desvia as LCDP das infraestruturas clássicas é facto de incorporarem todas ou a maioria de ferramentas e componentes necessários, uma vez que as ferramentas são todas integradas num ambiente, não existindo a necessidade de alternar entre diferentes sistemas, nem a necessidade de manter e integrar artefactos produzidos por tecnologias externas. Em termos de produtividade, os ganhos obtidos surgem, por exemplo, da redução do esforço em tarefas rotineiras e do fornecimento de

implementações reutilizáveis para tarefas genéricas. Outro elemento fundamental é o facto de poucos fornecedores apresentarem as suas plataformas como soluções para engenharia de software orientada a modelos, utilizando outras designações. Por fim, a reutilização é mencionada num nível de arquitetura mais genérico, sendo apenas reutilizadas pelas LCP estruturas genéricas muitas vezes implícitas para soluções específicas.

Lourenço, Ferreira, and Seco (2021) caracterizam as plataformas *low-code* como plataformas que possibilitam, a pessoas não especialistas e especialistas no desenvolvimento de software, criar e desenvolver sistemas e aplicações complexas. A maior vantagem da utilização destas plataformas prende-se com a reutilização de componentes criados, onde são capturados padrões comuns em todas as camadas das aplicações, nomeadamente, na interface do utilizador, na camada de dados e na integração com sistemas externos. Os autores destacam que a produtividade é um dos elementos fulcrais nos dias de hoje no que diz respeito ao desenvolvimento de software, podendo esta ser medida de diversas maneiras: ou através do tempo utilizado para criar um MVP, ou através da capacidade de adaptação à mudança, ou em atividades de manutenção, etc. Tendencialmente estas medidas são mais positivas no desenvolvimento via LCP. A plataforma *OutSystems* é apresentada como uma plataforma de desenvolvimento e entrega orientada por modelos gráficos que permitem o desenvolvimento de aplicações móveis e Web de nível empresarial. Esta plataforma dispõe de um IDE que integra num único local todos os aspetos essenciais ao desenvolvimento de soluções de software. Além da diminuição da complexidade do desenvolvimento, é ainda permitida a reutilização segura de artefactos de código abstrato. Os autores descrevem uma linguagem de *templates* para a plataforma *OutSystems* denominada de *OSTRICH*. Esta abordagem estende o meta-modelo *OutSystems*, adicionando elementos tais como: parâmetros de modelo, anotação de prioridade, anotação de iteração, e anotação condicional.

Overeem and Jansen (2021) afirmam que as plataformas de desenvolvimento *low-code* aceleram o desenvolvimento de software, auxiliando os utilizadores finais no que diz respeito à programação. Os responsáveis pelo desenvolvimento, com ou sem formação na área da programação, têm a oportunidade de desenvolver aplicações e soluções, que se estão a tornar cada vez mais complexas, com base no desenvolvimento orientado a modelos. Além da complexidade associada às aplicações, as LCDP estão a tornar-se também mais complexas,

incluindo desde serviços empresarias, a IoT, a promotores e facilitadores de transformações na indústria da produção, etc. Esta evolução acarreta consigo um conjunto de desafios com os quais os engenheiros de software já se deparavam antes, e, agora, o responsável pelo desenvolvimento, com ou sem formação na área de software, também se depara com eles. Esta complexidade torna as organizações mais dependentes das soluções criadas, sendo também prestada mais atenção à qualidade das soluções. Aumenta, ainda, a interdependência entre a organização, o responsável pelo desenvolvimento e os profissionais e engenheiros de software. A análise de impacto fornece a todos os profissionais o feedback necessário sobre como as mudanças têm impacto no sistema de software, para migrar corretamente dados existentes, planear atualizações, etc. A análise proposta e avaliada contempla três tipos de profissionais, os responsáveis pelo desenvolvimento das soluções, os fornecedores e responsáveis pelo desenvolvimento da LCDP e os engenheiros de operações responsáveis por manter as LCDP e as soluções funcionais. Além disso, três subsistemas e três artefactos também são considerados. O primeiro subsistema é o ambiente IDE da LCDP fornecido ao utilizador, que permite a criação do primeiro artefacto meta-modelo. Posteriormente, são criados os modelos projetados (segundo artefacto) utilizados pelo subsistema de transformação de modelo para criar um modelo de tempo de execução. Por fim, o subsistema da plataforma contém os recursos necessários para executar o modelo de tempo de execução. Os resultados gerados permitem perceber quais os impactos que as mudanças têm no sistema, auxiliando os profissionais na tomada de decisão.

Di Ruscio et al. (2021) caracterizam a perspetiva atual das plataformas *low-code*, onde as plataformas *low-code* assentes em *cloud* são cada vez mais populares tendo em conta a crescente procura por produtos e/ou serviços personalizados, económicos, seguros e confiáveis. Outros fatores que estão por detrás do sucesso das LCP são a disponibilidade de infraestruturas de *cloud computing* e a grande procura verificada de profissionais de desenvolvimento de software. As soluções de software são desenvolvidas com o auxílio de uma interface gráfica, o que reduz substancialmente a quantidade de codificação manual tradicional, proporcionando que indivíduos sem experiência e competências em programação possam participar diretamente e ativamente no processo de desenvolvimento de uma solução de software.

ElBatanony and Succi (2021a) caracterizam o *no-code* como um paradigma responsável por capacitar indivíduos sem formação e competências no desenvolvimento de software, com o objetivo de estes serem capazes de desenvolver soluções de software. As ferramentas de *no-code* são vistas como ferramentas fáceis e rápidas de utilizar tendo em conta que não exigem nenhum conhecimento especializado e específico e fornecem um alto nível de abstração, nomeadamente, na grande parte dos problemas associados a atividades do desenvolvimento de software, tais como, a gestão, a compilação, a implantação, os testes, etc. O *no-code* é encarado como algo que vai romper com as barreiras entre os indivíduos e o software, sendo salientado um aspeto importante relativo a que este tipo de tecnologia não irá colocar em causa o papel do responsável pelo desenvolvimento de software tradicional e que pode ainda ser benéfico para este tipo de profissionais, possibilitando a criação de novas soluções, inovadoras e disruptivas. Os autores mencionam um conjunto de vantagens associadas a este paradigma: (1) custos de desenvolvimento mais reduzidos, (2) tamanho das equipas mais reduzido, (3) tempo de lançamento no mercado mais rápido. Os resultados do estudo permitem também afirmar que usar uma *no-code development platform* (NCDP) para auxiliar os responsáveis pelo desenvolvimento nas fases iniciais de um projeto, contribuiria para a redução do número de funcionalidades a desenvolver em quase 50%, possibilitando que os responsáveis pelo desenvolvimento focassem a sua atenção em tarefas críticas e que trouxessem valor competitivo.

ElBatanony and Succi (2021b) defendem que uma nova era de desenvolvimento de software está a chegar, onde qualquer pessoa será capaz de desenvolver soluções de software sem ter a necessidade de se preocupar em escrever código. Os autores expõem as plataformas *no-code* como ferramentas destinadas a pessoas que normalmente não são profissionais em desenvolvimento de software e fornecem uma experiência ligeiramente mais restritiva quando comparadas com as abordagens tradicionais de desenvolvimento de software. O desenvolvimento de software tem sofrido grandes evoluções ao longo dos anos, sendo que o que se espera que aconteça no futuro é que qualquer pessoa seja capaz de desenvolver e implementar soluções de software, independentemente da sua experiência e do nível de conhecimento. Os autores acreditam que o primeiro passo na evolução no desenvolvimento de software terá como base uma ferramenta de código aberto, capaz de desenvolver aplicações multiplataforma usando componentes da interface do utilizador. Esta ferramenta

implementaria o desenvolvimento orientado à configuração para construção das soluções. As funcionalidades das aplicações, o *back-end*, os armazenamentos de dados, seriam realizados em *cloud* e as funcionalidades dependeriam de APIs. Numa fase mais posterior de evolução, a inclusão de inteligência artificial, *chat bots* e *natural language processing* (NPL), seria o próximo passo a dar.

Khorram et al. (2020) apresentam as plataformas de desenvolvimento *low-code* como sendo um software em *cloud* destinado a não profissionais de desenvolvimento de software, que contribui para que estes possam criar soluções sem ter o conhecimento especializado. É visível a migração do desenvolvimento tradicional, recorrendo a linguagens de programação, para um desenvolvimento que tem como suporte uma interface gráfica. As soluções são criadas tendo em conta altos níveis de abstração, possibilitando uma mais rápida entrega das soluções e também um menor custo de desenvolvimento. O que contribui para o sucesso destas plataformas é a facilidade de uso e a simplicidade percecionada pelos utilizadores que usam estas plataformas e que não possuem nenhuma experiência em programação. Independentemente da plataforma utilizada para desenvolver a solução, a solução criada deve ser testada para aumentar a probabilidade de sucesso, e o papel do responsável pelo desenvolvimento assume uma elevada importância neste momento. Contudo, a falta de experiência e de conhecimento destes utilizadores, na sua grande maioria das vezes, acarreta consigo um conjunto de desafios, mais especificamente, no que diz respeito à realização de testes das soluções desenvolvidas com auxílio das LCDP, que é uma área ainda muito pouco explorada. Os autores propõem assim uma base de 16 recursos personalizados para testes *low-code*, divididos em cinco categorias, sendo elas: (1) geral, que envolve uma estrutura de teste, uma escala de teste suportada, um suporte de verificação e uma abertura a outras plataformas e ferramentas de teste; (2) design do teste, que envolve o papel do designer de testes, a colaboração no projeto de teses, as técnicas de projetos de testes, os artefactos usados e produzidos no projeto de teste e a reutilização; (3) a geração do teste, que engloba a automação de geração de testes e a linguagem script de teste; (4) a execução do teste, com a automação da configuração, a distribuição, a ferramenta/serviço de execução de teste e a plataforma de execução de teste; (5) a avaliação do teste, com a técnica de avaliação do resultado do teste.

Cabot (2020) considera que a chave para a transformação digital da sociedade passa pela tecnologia *low-code*. Estas plataformas aceleram a entrega de aplicações, uma vez que reduzem substancialmente a quantidade de codificação manual. Segundo a perspetiva do autor, o *low-code* pode ser equiparado ao *model-driven development*, no entanto como uma visão mais restrita. Além disso, é ainda mencionado que frequentemente nos deparamos com ferramenta *low-code* /*no-code*, onde o *no-code* aparece como uma variante do *low-code*. No entanto, a abordagem apresentada pelo *no-code* é diferente em comparação com o *low-code*, uma vez que neste paradigma os responsáveis pelo desenvolvimento devem criar e implementar as soluções sem recorrer em nenhum momento à codificação manual, o que pode limitar e condicionar o tipo de soluções que podem ser desenvolvidas. Em oposição, no *low-code*, apesar de grande parte do código ser gerado automaticamente, em alguns momentos pode ser necessário personalizar as aplicações através do ajuste do código gerado.

Luo et al. (2021) assumem que as empresas estão a sofrer cada vez mais pressões do seu ambiente e que, por isso, é necessário que estas tomem decisões rápidas e eficazes, e que adotem respostas resilientes às mudanças das necessidades do mercado. Tendo em conta que a procura por sistemas de informação está a aumentar a um ritmo muito superior em comparação com a resposta que os departamentos de TI podem fornecer, e tendo em conta a dificuldade verificada em contratar engenheiros de software uma vez que a procura supera a oferta, é necessário encontrar soluções que respondam a estes problemas. Uma das soluções que pode ser adotada são as plataformas *low-code development* (LCD), que surgem como ferramentas que permitem que qualquer profissional seja capaz de construir soluções de software sem que tenha necessariamente experiência e conhecimento em codificação, sendo o processo de desenvolvimento acelerado, permitindo que as empresas acompanhem mais facilmente, eficazmente e eficientemente as necessidades do mercado. É possível afirmar, segundo os autores, que o processo de desenvolvimento pode ser acelerado em 5 a 10 vezes. Como principais resultados do estudo e após a análise das publicações feitas no *Stack Overflow*, é possível concluir que vários profissionais usam diferentes termos para descrever as plataformas LCD, nomeadamente, low-code, arrastar e soltar, programação gráfica, *what you see is what you get* (WYSIWYG), etc. Além disso foi possível identificar 21 plataformas LCD que são mencionadas nas publicações, 14 das quais comerciais e as restantes 7 de código aberto. Foram identificadas linguagens de programação usadas no LCD como Java,

JavaScript, C#, Python, entre outras. No que toca às unidades de implementação em LCD, temos em grande destaque as APIs, os *templates*, os componentes e os serviços. Já relativamente às aplicações desenvolvidas com auxílio deste paradigma, encontram-se em lugar de destaque as aplicações móveis e web. Alguns dos benefícios reconhecidos e resultantes da investigação realizada associados às plataformas LCD são: (1) rápido desenvolvimento de soluções; (2) facilidade de uso; (3) custo mais reduzido, tendo em conta que não é necessário contratar profissionais especializados para o desenvolvimento das soluções; (4) abrangência graças à existência vários módulos e de fácil instalação; (5) interfaces amigáveis para os responsáveis pelo desenvolvimento, que muitas vezes não possuem competências técnicas e que acabam por criar software de forma fácil, intuitiva e familiar; (6) alta qualidade das soluções desenvolvidas, sendo seguras e escaláveis; (7) grande capacidade de integração e expansão, que requer um mínimo de esforço e uma boa capacidade de personalização; (8) são mais intuitivas e flexíveis; (9) fornecem uma boa experiência ao utilizador, disponibilizando uma interface gráfica; (10) facilidade de implementação; etc. Por outro lado, as limitações salientadas são: (1) a curva de aprendizagem, que pode ser em alguns casos mais íngreme; (2) elevado custo associados às plataformas LCD e à sua utilização; (3) a personalização, que é mais restritiva em comparação com o modelo de desenvolvimento tradicional; (4) menos poderosas em relação às linguagens de programação tradicionais; (5) a necessidade de codificação em questões que são consideradas mais complexas e críticas; (6) a dependência face aos fornecedores; (7) a dificuldade de manutenção; etc.

Di Sipio, Di Ruscio, and Nguyen (2020) acreditam que os sistemas de recomendação estão a assumir um papel cada vez mais fundamental na engenharia de software, tendo em conta a heterogeneidade de elementos, potenciando a capacidade dos responsáveis pelo desenvolvimento na construção de soluções através do fornecimento de informações e sugestões. No entanto, a criação destes sistemas é uma tarefa que se revela complexa, nomeadamente a personalização dos sistemas tendo em consideração necessidades do utilizador. Tendo em conta estes fatores, o *low-code* surge como um paradigma capaz de responder a esses desafios. Os autores sugerem que o desenvolvimento *low-code* se baseia na combinação de duas abordagens, mais especificamente, a engenharia orientada a modelos e o desenvolvimento rápido de aplicações. As LCPD são vistas como sistemas que

disponibilizam uma interface gráfica, fácil de utilizar, que permite que qualquer utilizador possa criar soluções de software. As LCPD estão cada vez mais presentes no mercado, tendo em conta que têm ambientes fáceis de utilizar e aceleram o desenvolvimento de soluções de software que consideram as restrições de tempo típicas e que se verificam nas organizações atualmente. Algumas vantagens associadas a estas plataformas são: (1) o aumento da inovação digital; (2) fácil e rápido desenvolvimento de soluções; e (3) a redução de custos durante o processo de desenvolvimento. O propósito principal destas plataformas é permitir que qualquer pessoa, com pouca ou nenhuma formação em programação, possa desenvolver as suas próprias soluções. É de salientar, ainda, e segundo os autores, que é importante ter alguma competências técnicas e formação para que o desenvolvimento das aplicações seja adequado e completo. Assim sendo, apresentam uma proposta de desenvolvimento de sistemas de recomendação recorrendo a tecnologia *low-code*.

Bragança et al. (2021) apresentam as plataformas *low-code* como ferramentas que permitem a qualquer pessoa construir aplicações de software de forma autónoma, mais concretamente, aplicações Web e aplicações móveis. Uma boa parte destas plataformas oferece suporte para a reutilização, a fim de facilitar o desenvolvimento de soluções semelhantes. Os autores referem, ainda, que estas soluções são incapazes de atingir o nível industrial, sendo atualmente mais destinadas ao desenvolvimento de pequenas aplicações. Apesar de as plataformas não exigirem nenhum tipo de conhecimento ou formação para serem utilizadas, podendo qualquer pessoa assumir a função de responsável pelo desenvolvimento de software, quando utilizadas por profissionais de software, estas permitem que estes adequem o seu esforço e a sua atenção a tarefas realmente importantes e críticas, abstraindo-se de pormenores e aspetos mais técnicos. A utilização destas plataformas está a verificar um aumento exponencial, quer por parte de profissionais qualificados quer por profissionais não qualificados, o que conduz à conclusão de que as soluções desenvolvidas serão mais complexas e terão como grande objetivo resolver problemas e necessidades mais exigentes. Os ganhos em produtividade e a facilidade de uso promovem uma maior adoção das *low-code application platforms* (LCAPs). A principal limitação das LCAP é que raramente fornecem a possibilidade de acesso ao meta-modelo, aos modelos e ao código das aplicações. A abordagem apresentada não requer nenhuma alteração nas LCAP, sendo apenas necessário que estas tenham um mecanismo para importar

e exportar modelos de aplicação. A solução está assente num conjunto de *domain specific languages* a serem utilizadas num *software product line* IDE.

Horváth, Horváth, and Wimmer (2020) consideraram que as plataformas *low-code* surgiram como a próxima geração de plataformas colaborativas disponíveis em *cloud*. Estas plataformas permitem que qualquer pessoa possa conceber uma aplicação de software completa, num curto espaço de tempo, proporcionando ciclos *time-to-market* e *time-to-value* mais reduzidos. Os responsáveis pelo desenvolvimento constroem modelos de software que vão sendo refinados em diagramas com diferentes níveis de abstração, usam transformações de modelos para obter o código-fonte, testes, etc. Outra vertente abordada é a *Model-Based Systems Engineering* (MBSE), que abrange todo o ciclo de vida de modelação e que combina modelos de vários domínios e oferece uma plataforma colaborativa para engenheiros de diferentes áreas. Para beneficiar das características colaborativas das *low-code platforms* em engenharia de sistemas, é necessário adequar estas plataformas aos desafios da *Model-Based Systems Engineering* (MBSE) e o seu desempenho deve ser ajustado em dois aspetos, nomeadamente: (1) o número de utilizadores que estão simultaneamente a trabalhar no modelo e (2) a escalabilidade dos modelos. Os autores apresentam, assim, um conjunto de requisitos que estas plataformas devem ter num futuro próximo, nomeadamente: (1) devem oferecer um ambiente *multi-tenant* para gerir o trabalho colaborativo; (2) devem fornecer um paradigma de processamento de modelos que pode ser dimensionado para vários milhões de elementos; e (3) devem fornecer aos engenheiros um conjunto de critérios de seleção para escolher o mecanismo de transformação do modelo certo em ambientes de execução *multi-tenant*.

Almonte, Cantador, Guerra, and de Lara (2020) afirmam que as LCDP contribuem para que as pessoas com fraco ou inexistente conhecimento técnico possam construir soluções de software completas, recorrendo a interfaces gráficas, diagramas e linguagens declarativas. Estas plataformas habitualmente são disponibilizadas na *cloud*, como um serviço PaaS. Já o *Model Driven Environment* (MDE), segundo os autores, é uma abordagem que permite o desenvolvimento de software através de um elevado nível de abstração. Os modelos gerados podem ser usados para especificar, analisar, testar, gerar código e manter sistemas, contribuindo para a eficiência e a qualidade do trabalho realizado. Os IDEs são vistos como ambientes que ajudam os programadores a serem mais eficientes recorrendo ao

preenchimento automático de código, a correções rápidas, etc. Como já referido anteriormente, o foco do estudo está assente nos responsáveis pelo desenvolvimento de soluções que usam LCDP, que são na sua grande maioria profissionais que não possuem competências em programação, daí ser importante integrar mecanismos que contribuam para ajudar estes profissionais no desenvolvimento das aplicações. Os sistemas de recomendação associados às LCDP são, assim, encarados como um mecanismo importante, uma vez que fornecem informações e sugestões aos utilizadores que são geradas de acordo com o tipo de solução que se pretende criar e implementar. No entanto, o desenvolvimento destes sistemas de recomendação exige bastante tempo pois é necessário escolher e implementar o método de recomendação adequado, bem como configurá-lo tendo em conta o problema em questão, e ainda avaliar a precisão das suas recomendações. Como tal, é apresentado um *framework* para automatizar e auxiliar a construção de sistemas de recomendação que contribuem com informações importantes para o responsável que usa as LCDP e que não possui conhecimentos em programação.

Sahay, Di Ruscio, and Pierantonio (2020) mencionam que as plataformas de desenvolvimento de *low-code* conferem a capacidade a qualquer pessoa, com pouca ou sem experiência em programação, de desenvolver soluções de software. Caracterizadas por serem ambientes gráficos, estas plataformas permitem definir fluxos de trabalho que consistem em execuções sequenciais ou paralelas de serviços, que são disponibilizados na própria plataforma ou em entidades externas. Em momentos em que é necessário definir fluxos de trabalho complexos, os responsáveis pelo desenvolvimento devem estar conscientes dos fornecedores de serviços com os quais a LCDP é capaz de ser integrado, e os meios que podem ser utilizados para desenvolver o processo idealizado. Especificar fluxos de trabalho envolvendo diferentes LCDP e serviços pode ser uma tarefa difícil e complexa. Os autores propõem uma abordagem que contempla vários componentes: (1) meta-modelo de especificação de metas, onde seria empregada uma técnica já conhecida para encadear transformações de modelos, sendo a abordagem capaz de identificar possíveis cadeias de transformação; (2) meta-modelo LCDP, para especificar as características dos modelos LCDP suportados, definindo os modelos de *workflow* que podem ser executados pelo LCP correspondente; (3) conectores LCDP, que são componentes de software que permitem que o sistema se conecte às diferentes LCDP; (4) *composition reasoner*, que verifica a viabilidade

do objetivo de entrada em relação aos serviços fornecidos pelas LCDP que o sistema é capaz de conectar; (5) transformação *Goal2Workflow*, componente este que gera possíveis *workflows* compatíveis com os objetivos especificados; e (6) *workflow engine*, que é componente capaz de executar os modelos gerados.

Philippe, Coullon, Tisi, and Sunyé (2020) afirmam que as plataformas de desenvolvimento de *low-code* estão a assumir cada vez mais um papel de destaque na engenharia orientada a modelos, conduzindo a novos desafios como a eficiência e a escalabilidade. As LCDP propõem interfaces gráficas para o desenvolvimento de soluções de software, fornecidas com um serviço PaaS, que tem como um dos seus objetivos minimizar a codificação manual. Na sua generalidade, as LCDP são baseadas na descrição do comportamento das soluções recorrendo a modelos, tal como acontece na engenharia orientada a modelos. O desempenho das operações realizadas sobre estes modelos assume-se como um tema de investigação de grande importância, mais concretamente, a gestão destes modelos nas LCDP expressa uma necessidade significativa de operações automáticas e transparentes, eficientes e escaláveis para a atualização, consulta e análise dos modelos, sendo que existem três razões que estão na base dessa necessidade. A primeira razão prende-se com facto das LCDP terem a necessidade de fornecer interfaces gráficas de desenvolvimento complexas com baixo tempo de resposta, que é considerado um fator de qualidade que tem influência no conforto e na eficiência do responsável pelo desenvolvimento das soluções. Em segundo lugar, surge o problema da escalabilidade em modelos de dados de grande dimensão, sendo necessário fornecer uma solução eficiente. Por fim, o número de operações simultâneas que podem ser efetuadas numa LCDP, cria a necessidade de uma gestão eficiente. Apesar do grande número de operações realizadas em paralelo, tendo em conta que as LCDP são fornecidas como PaaS, permitindo que vários utilizadores consultem modelos, acedam a servidores e base de dados ao mesmo tempo, a eficiência e a escalabilidade não devem ficar comprometidas.

Di Ruscio et al. (2022) assumem que nos últimos anos se verificou um crescimento expressivo das plataformas de desenvolvimento *low-code*. Estas plataformas são encaradas como plataformas de desenvolvimento gráfico, habitualmente disponíveis em *cloud*, que reduzem a necessidade de codificação manual e destinadas também a indivíduos não profissionais em desenvolvimento de software. Os principais propósitos das LCDP são a redução do esforço de desenvolvimento e de manutenção para entregar e construir

determinado tipo de aplicações e permitir que qualquer pessoa, com ou sem experiência em programação, possa contribuir diretamente para o processo de desenvolvimento de software. Segundo os autores, o termo *low-code* surgiu pela primeira vez em 2014 através da *Forrester* (Richardson & Rymer, 2014), onde as LCDP foram definidas como plataformas que permitem a entrega rápida de aplicações de negócio com um mínimo de codificação manual e com um mínimo de investimento inicial em configuração, formação e implementação. Com o passar dos anos, a definição foi aprimorada e, em 2017, apresentava-se como produtos e/ou serviços em *cloud*, destinados ao desenvolvimento de aplicações que empregam técnicas visuais declarativas em vez de programação e que estão disponíveis para os clientes a um custo reduzido, exigindo igualmente um reduzido tempo de formação, com os custos a aumentar na proporção do valor para o negócio. O foco está voltado para as interfaces gráficas e técnicas declarativas. Já as plataformas de desenvolvimento *no-code* (NCDP), são apresentadas como ferramentas que eliminam totalmente a necessidade de programação, usando linguagens visuais, interfaces gráficas do utilizador e configuração. Algumas características associadas as LCDP são: (1) a experiência proporcionada ao utilizador através de interfaces gráficas avançados que, permitem o desenvolvimento das soluções; (2) o desenvolvimento colaborativo, que permite a interação entre responsáveis pelo desenvolvimento que se encontram em espaços físicos diferentes; (3) o foco no tipo de solução a desenvolver; (4) o fornecimento de artefactos pré-definidos, que podem ser utilizados como ponto de partida para os responsáveis pelo desenvolvimento. O desenvolvimento de sistemas de software recorrendo a LCDP engloba diversas fases, sendo elas: (1) modelação de domínio, onde são disponibilizados aos utilizadores elementos para que estes possam representar as relações e os conceitos que sustentam a solução que pretendem desenvolver; (2) definição da interface do utilizador, onde são definidas as formas e as páginas de dados para criar, editar, e visualizar os dados que a solução desenvolvida vai gerir; (3) a especificação da lógica de negócio, onde através de diversos mecanismos o utilizador define o controlo e os fluxos de dados do sistema; (4) a integração com serviços externos e fontes de dados; (5) a criação e implementação das aplicações idealizadas, (6) a manutenção das soluções desenvolvidas. É, ainda, realizada uma comparação entre a MDE e as LCDP. Na MDE os modelos são utilizados para especificar, testar, simular, manter, gerar, etc., código para o sistema, contudo nem todos processos MDE terminam com a geração de código. O seu objetivo é aumentar a produtividade através da

automatização de diferentes etapas no desenvolvimento de software recorrendo à construção de modelos. Por outro lado, as LCDP permitem a construção de soluções através da utilização de interfaces gráficas, onde a codificação manual é reduzida ou mesmo nula. Outras características que distinguem os paradigmas são: as LCPD estarem disponíveis em *cloud*, em comparação com a MDE que, na sua grande maioria, são soluções baseadas em desktop; as LCPD estão presentes noutros domínios como a IoT, *chatbots*, enquanto que a MDE se dedica a outras áreas como a engenharia de energia, sistemas *cyber-physical*, etc. Finalmente, as razões por detrás do sucesso das LCPD são: (1) a implementação baseada em *cloud*, (2) a força de trabalho nativa, tendo em conta que, atualmente, profissionais, de várias áreas têm contacto com múltiplas ferramentas de software, (3) a ausência da necessidade de configuração, uma vez que as ferramentas são de fácil instalação e utilização, (4) a procura de profissionais de TI inferior à capacidade de oferta, e (5) os meios para aquisição de competências, com a utilização de plataformas digitais que permitem a partilha de vários conteúdos atualizados e o contacto com outros utilizadores das LCDP.

Bock and Frank (2021b) apresentam as plataformas de desenvolvimento *low-code* (LCP), plataformas de aplicação *low-code* (LCAP), e plataformas de desenvolvimento *low-code* (LCDP), como ambientes de desenvolvimento de software que potenciam o aumento da produtividade no que diz respeito ao desenvolvimento de software e que promovem novas formas de alinhamento das TI de negócio e o *empowerment* do utilizador. Os autores afirmam, num primeiro momento, que as LCP são capazes de concretizar os objetivos que têm estado no foco da investigação em SI, nomeadamente: o aumento da produtividade, a redução de custos de desenvolvimento e de manutenção dos sistemas, incrementando a capacidade de adaptação das organizações de acordo com os requisitos que estão em constante mutação. A falta de profissionais na área da TI, importantes nas iniciativas de transformação digital, a ineficiência de alguns projetos de software e a grande expectativa do posicionamento das LCP no mercado, contribuem para o aumento da popularidade destas plataformas. Um dos resultados apresentado é que grande parte das soluções atuais advém de outras soluções passadas já existentes que utilizavam outro tipo de designação, tais como, desenvolvimento rápido de aplicações, plataformas como serviço (PaaS), plataformas de aplicações orientadas a modelos, etc. Ao longo do estudo foram também identificadas características referentes às LCP. As características mais transversais a todas as plataformas enquadram-se numa

perspetiva estrutural. Todos os produtos e serviços possuem um componente para a definição de estruturas de dados, tendo um recurso comum relacionado, sendo este a capacidade de aceder a fontes de dados externas através de APIs. Outro recurso presente em todas as plataformas é a componente de criação de interface de utilizador, que contribui para a criação das interfaces gráficas e para a sua integração com outros artefactos de implementação. Além destas, as LCP oferecem especificações funcionais básicas, tais como, linguagens para regras de decisão, biblioteca de operações padrão genéricas, invocação e integração de funções externas por meio de APIs, etc. Quase todas as plataformas oferecem suporte avançado para a implementação das aplicações, assim como são disponibilizados componentes para definir funções e direitos do utilizador.

Por fim, algumas características ocasionais são a disponibilização de elementos de codificação avançados ou tradicionais, e a disponibilização de um componente de modelação de fluxo de trabalho e um mecanismo de fluxo de trabalho. As LCP integram vários componentes clássicos de desenvolvimento de software, no entanto, estas diferem das estruturas tradicionais tendo em conta que integram a maioria das ferramentas e componentes necessários para o desenvolvimento de soluções de software. A reutilização é abordada nas plataformas num nível de arquitetura genérico. Em termos dos ganhos de produtividade, estes decorrem da redução do esforço de tarefas rotineiras.

Bhattacharyya and Kumar (2021) focam o conceito de aplicações *low-code* e *no-code* e estudam a sua utilização para Web design, o desenvolvimento rápido de aplicações e a digitalização da cadeia de valor. A realização da revisão de literatura efetuada por estes autores permitiu perceber que o número de publicações que abordam as tecnologias *low-code* e *no-code* e a digitalização da cadeia de valor é reduzido. Após a realização do estudo, os autores concluíram que as aplicações *low-code* e *no-code* podem ser utilizadas em toda a cadeia de valor, promovendo, por exemplo, uma maior transparência para que os gestores tomem decisões com base nos dados. O desenvolvimento rápido de aplicações com o auxílio do *low-code*, é encarado como algo que pode reduzir substancialmente a dependência das organizações face aos programadores e profissionais de software. Além disso, foram igualmente destacados alguns benefícios, tais como: (1) a redução de custos, (2) o desenvolvimento mais rápido, (3) transparência e simplificação de processos, (4) capacidade de qualquer pessoa, com ou sem formação em programação, se tornar responsável pelo

desenvolvimento das soluções, (5) aumento da velocidade de lançamento de soluções no mercado, (6) a independência de terceiros (fornecedores e responsáveis pelo desenvolvimento de soluções de software). Em especial, e no que diz respeito às PME, as aplicações *low-code* e *no-code* económicas e fáceis de manter, podem contribuir para que estas se tornem mais competitivas no mercado. Algumas barreiras que limitam a utilização de forma mais generalizada são: (1) a falta de estratégias de adoção adequadas disponíveis para as organizações, (2) a pobre e a fraca perceção cultural e a falta de consciencialização sobre as tecnologias.

Ihirwe, Di Ruscio, Mazzini, Pierini, and Pierantonio (2020) afirmam que o desenvolvimento de sistemas IoT enfrenta vários desafios, tendo em conta a grande diversidade de dispositivos e componentes, daí surgirem ferramentas de apoio ao desenvolvimento IoT, neste caso com enfoque nas LCP. Os autores apontam que a tendência atual das plataformas de desenvolvimento *low-code* capta a atenção das organizações, uma vez que proporcionam um desenvolvimento mais facilitado de aplicações totalmente funcionais e baseadas em *cloud*. O principal objetivo destas plataformas é contribuir para que pessoas, com ou sem experiência e/ou formação em engenharia de software, possam construir aplicações de negócio através de interfaces gráficas do utilizador simples. O projeto apresentado "*Lowcomote*", pretende impulsionar o avanço do *low-code*, para uma Era mais técnica e sofisticada, por via da fusão de técnicas de engenharia orientada a modelos, *cloud computing* e *machine learning*. As LCP têm revelado os seus pontos fortes no que toca ao desenvolvimento de software em vários segmentos de mercado, como, por exemplo, as aplicações de base de dados e aplicações móveis e um segmento de mercado próximo será a IoT. Após a análise de 16 plataformas MDE e *low-code,* foi possível identificar um conjunto de pontos fracos das plataformas: (1) a falta de padrões para apoiar o desenvolvimento de sistemas IoT, onde cada plataforma tem a sua própria forma de desenvolvimento impossibilitando a interoperabilidade entre plataformas; (2) o suporte limitado à modelação de múltiplas visualizações, onde grande parte das abordagens usa uma visão específica para toda a modelação; (3) o suporte limitado para engenharia baseada em modelos em *cloud*; (4) suporte limitado para testes e análises.

Segundo Alsaadi et al. (2021), as plataformas de desenvolvimento *low-code* foram introduzidas para combater a lacuna existente entre a alta procura de aplicações e o baixo número de profissionais de software disponíveis para atender a essa procura, permitindo que

qualquer pessoa possa assumir a função de responsável pelo desenvolvimento, podendo construir aplicações rápidas, eficientes, escaláveis com uma necessidade mínima de programação. Além disso, o crescente número de aplicações móveis e Web e a transformação digital, podem contribuir para o desenvolvimento destes paradigmas. Existe um conjunto de fatores que diferenciam o *low-code,* do *no-code* e da abordagem tradicional de desenvolvimento, sendo eles: o tamanho das equipas, os recursos necessários, prototipagem, o tempo de desenvolvimento dos produtos, o custo, risco de investimento, etc. Enquanto na abordagem tradicional, a tecnologia de codificação é totalmente ajustada à estrutura da tarefa e mecanismos, e é eficiente quando utilizada por uma equipa que possui conhecimento em programação, no caso do *low-code*, considerado como a quarta geração de linguagem de programação, o ambiente de desenvolvimento reflete-se numa interface gráfica, baseado na geração automática de código e no design orientado por modelo. O *low-code* permite que qualquer pessoa, com ou sem formação técnica, possa criar e entregar aplicações com o mínimo de esforço, e focar a atenção dos profissionais responsáveis pelo desenvolvimento de software em operações mais críticas do desenvolvimento de aplicações como a instalação, a configuração, teste da solução, etc. O *no-code* é visto por alguns como parte integrante do mercado das LCDP, contudo este concentra-se no desenvolvimento de soluções de software sem que exista a necessidade de escrever qualquer linha de código, soluções essas que apresentam funcionalidades mais limitadas. Um aspeto que também caracteriza a abordagem tradicional é a necessidade de escrever uma grande quantidade de código, na maioria das vezes, a partir do zero, o que tem um custo bastante elevado em termos de tempo, custo e esforço, sendo considerada uma abordagem complexa onde os conhecimentos técnicos são fundamentais. Por outro lado, o *low-code*, apesar de permitir um desenvolvimento mais rápido, que pode ser realizado por qualquer pessoa, tem também algumas limitações, associadas à falta de comentários no código, aos nomes usados e atribuídos a variáveis que podem ser vagos, problemas de segurança, etc. A utilização das LCDP permite, ainda, que os profissionais de TI foquem a sua atenção em tarefas consideradas mais complexas e críticas. As LCDP têm associadas a si outras designações, tais como engenharia de software do utilizador final, programação do utilizador final, e meta-design. Os resultados do estudo revelam que mais de metade dos participantes usam as LCDP para a criação de aplicações e sites de comércio eletrónico. Outro resultado interessante, é que os benefícios mais

relevantes obtidos com o uso das LCDP, na perspetiva dos participantes, são a facilidade de desenvolvimento e a redução do tempo de desenvolvimento. O principal desafio enfrentado no uso das LCDP é a personalização de alguns aspetos das aplicações criadas. Uma boa parte dos participantes também considerou que a qualidade das soluções desenvolvidas com recurso às LCDP é igual à qualidade usando abordagens tradicionais. Por outro lado, alguns afirmam que as LCDP não se destinam a soluções complexas, e que a integração das soluções desenvolvidas com LCDP com outros sistemas não apresenta problemas. Umas das limitações apontada que impede o uso das LCDP é a sua escalabilidade. Mais de metade da amostra do estudo considera que tem a intenção de vir a usar LCDP no futuro e tem noção de que o uso de LCDP reduzirá o tempo de desenvolvimento. Cerca de 43% da amostra considera que as soluções desenvolvidas com LCDP apresentam boa qualidade e desempenho.

Metrôlho, Ribeiro, and Araújo (2020) afirmam que, tendo em conta a elevada procura de profissionais com competências na área das TI, a incapacidade de as instituições de ensino superior responderem a essa necessidade, e o aumento do número de iniciativas de transformação digital das organizações, será fundamental satisfazer essas necessidades recorrendo a pessoas que não tenham como área de formação as TI. As plataformas de desenvolvimento *low-code* podem desempenhar um papel importante, uma vez que permitem que profissionais de outras áreas possam assumir as funções de responsável pelo desenvolvimento de soluções digitais e de software. Alguns benefícios associados às plataformas *low-code* são: (1) os recursos de segurança, o suporte multiplataforma, e a integração de dados, já se encontram disponíveis podendo ser facilmente personalizados; (2) não é necessário ter conhecimentos sobre linguagens de programação específicas e não é necessário ter experiência para usar as plataformas; (3) possibilitam a construção de aplicações para múltiplas plataformas simultaneamente; (4) permitem atualizações e entrega de novos recursos num curto espaço de tempo. A transformação digital traz como consequência a requalificação de profissionais de diversas áreas, tendo em conta que a força de trabalho é substituída em alguns casos por software ou máquinas. Neste trabalho, foram realizados alguns estudos de caso, onde uma LCP foi utilizada para requalificar profissionais. Os autores apresentaram dois cursos, onde a reconversão e a requalificação dos profissionais foi efetuada, com o auxílio da plataforma da *OutSystems*. Em cada uma das descrições dos cursos, foi apresentado o plano do curso, assim, como os seus principais resultados. O

primeiro curso foi destinado a profissionais desempregados ou que estavam com dificuldades em exercer na sua área de formação. Após a requalificação dos profissionais, nove dos 21 profissionais foram contratados pela empresa onde efetuaram o seu estágio, e após seis meses apenas seis permaneciam desempregados. O segundo curso foi destinado a profissionais que estavam no mercado de trabalho e que pretendiam aperfeiçoar as suas competências. Todos os profissionais concluíram o curso e a sua adaptação à plataforma *low-code* foi fácil e rápida.

Silva, Vieira, Campos, Couto, and Ribeiro (2021) apresentam as plataformas de desenvolvimento *low-code* como ferramentas capazes de responder à necessidade de maior produtividade no desenvolvimento de software, contribuindo para combater o problema de escassez de profissionais de desenvolvimento de software, através de nível de abstração mais alto onde o software é desenvolvido, automatizando tarefas de mais baixo nível e rotineiras. Estas plataformas reduzem a barreira de entrada para o desenvolvimento de software, ocultando detalhes mais técnicos. Estas plataformas podem assumir também a designação de plataformas de desenvolvimento do utilizador final. Além do termo acima mencionado, existem outros termos equivalentes e associados ao desenvolvimento de software através da utilização das LCDP, tais como, programação do utilizador final, engenharia de software do utilizador final, e meta-design. As LCDP suportam o desenvolvimento de aplicações de software através da codificação manual mínima com o objetivo de capacitar diferentes utilizadores, com diversas áreas de formação, para que este possam criar as suas soluções de maneira fácil e rápida. Os autores propuseram a utilização de um Modelo Cognitivo Descritivo, designado PRECOG, para estudar a interação entre programadores com muita experiência e programadores menos experientes e as LCDP, e identificar problemas. Foi possível observar que os programadores com mais experiência enfrentaram vários tipos de problemas, mas que estes decorrem com menos frequência. Já os programadores menos experientes enfrentaram menos tipos de problemas, mas que ocorreram com mais frequência, assumindo-se como problemáticos para este perfil. Outro resultado que teve impacto nos valores mencionados anteriormente foi o facto de os programadores mais experientes executarem mais tarefas em comparação com os menos experientes. A abordagem PRECOG assume-se, assim, como promissora, uma vez que foi capaz de identificar grande parte dos problemas enfrentados

pelos diferentes tipos de programadores. Esta metodologia pode ser usada eficazmente para prever, entender e mitigar erros de uso e interações irregulares nas LCDP.

Henriques, Lourenço, Amaral, and Goulão (2018) mencionam que as *Modelling Languages* estão a ser cada vez mais adotadas na indústria. Através destas linguagens, os autores acreditam ser possível melhorar a experiência do responsável pelo desenvolvimento, e a sua fácil adoção pode tornar os responsáveis pelo desenvolvimento mais produtivos, criando um impacto económico. A plataforma *OutSystems* é um exemplo disso, onde através da mesma podem ser criadas aplicações móveis e Web com um conjunto de *domain-specific languages* (DSLs) integradas. Estas DSLs são linguagens visuais que permitem a construção de soluções com um elevado nível de abstração, ocultando detalhes de mais baixo nível. Em comparação com as abordagens tradicionais, a plataforma da *OutSystems* possibilita desenvolvimentos mais rápidos e com maior qualidade. Esta plataforma possui uma DSL denominada *Business Process Technology*, que é utilizada por profissionais de programação e de modelação de processos de negócio como meio de comunicação com os gestores de negócio. Contudo, a DSL não estava a ter a adoção esperada, daí ser essencial identificar as falhas existentes e realizar as alterações necessárias. Os autores, tendo em conta estes fatores, decidiram desenvolver uma versão aprimorada da DSL. A comparação entre as duas versões da DSL permitiu concluir que a versão aprimorada possui uma classificação de usabilidade mais alta, uma transparência semântica mais elevada, entre outras vantagens.

Ragusa and Henriques (2018) afirmam que o processo de revisão de código é uma prática corrente na indústria do software e normalmente este processo é conduzido através de ferramentas disponibilizadas em *cloud*, o que contribui para melhorar a compreensão e a manutenção do código. No entanto, no caso das linguagens de *low-code*, ou seja, das *Visual Programming Languages* (VPLs), esse processo não se encontra prontamente disponível, o que acarreta consigo um desafio para a eficácia do processo. Apesar da presença das VPLs no mercado ter aumentado significativamente, os seus mecanismos para potenciar e melhorar a qualidade do software desenvolvido não evoluíram. As soluções desenvolvidas não são suportadas por ferramentas de linguagens de programação existentes, o que provoca constrangimentos e dificuldades no processo de revisão. Tendo por base estes fatores, foi criada uma ferramenta para auxiliar as revisões de código de VPLs. A arquitetura da solução pode ser decomposta em dois componentes principais: (1) os serviços principais responsáveis

por converter e armazenar os artefactos gráficos; e a (2) ferramenta que é capaz de calcular a diferença entre versões e apresentá-las lado a lado de forma visual. Os utilizadores têm apenas acesso à interface, onde focam a sua atenção na deteção de erros, melhorando a qualidade do código.

Hurlburt (2021) assume que a programação funcional foi o precursor do *low-code* e do *no-code*. Comparando o *low-code* com o *no-code*, é possível identificar um conjunto de diferenças. Enquanto no *low-code* é necessário por vezes possuir alguns conhecimentos em programação, o *no-code* caracteriza-se por permitir que qualquer pessoa (nomeadamente, pessoas sem formação ou com formação residual em programação) consiga desenvolver aplicações personalizadas. A inteligência artificial, o *machine learning* (ML), e a *robotic process automation*, estão a conduzir o *low-code* para a perspetiva da criação de software de forma cada vez mais autónoma, sendo que o autor defende que o *low-code* pode eventualmente ser "eclipsado" pelo *no-code*. São, ainda, colocadas as seguintes questões: "o papel do profissional de TI está em causa dentro das organizações?", e "o profissional de TI é capaz de reter valor para a organização?". A resposta, segundo o autor, é afirmativa às duas questões e baseada em alguns fatores. Em primeiro lugar, apesar da eficiência ser maior com o *low-code* e o *no-code*, no que toca à segurança, estes paradigmas são mais frágeis. As ameaças e os ataques informáticos são cada vez mais frequentes, além disso a dependência existente em relação a redes, dados e outras aplicações, pode criar condições para a introdução de valores e dados incompletos e vulnerabilidades. É importante assumir uma ação pró-ativa com o objetivo de combater estas situações e melhorar continuamente os mecanismos de segurança, tornando-os mais eficazes, sendo aqui relevante o profissional de TI. Em segundo lugar, algumas aplicações *low-code* e *no-code* podem não ser interoperáveis e a sua integração com outras entidades e APIs pode tornar-se uma tarefa complexa, sendo necessário ter competências especificas, características do profissional de TI. Com o avanço da tecnologia, como, por exemplo, a utilização da inteligência artificial, do *machine learning*, e o aparecimento da computação quântica, o papel do profissional de TI torna-se mais importante na implementação destas tecnologias, tendo em conta que alguns destes elementos não estão totalmente assentes em padrões e muitas vezes são considerados "opacos", o que pode trazer riscos associados. O objetivo do profissional de TI é minimizar esses riscos, que podem ser prejudiciais para a empresa. Por fim, na transformação digital, a sua gestão, a harmonização

e a interdependência de vários processos e a visão impulsionadora devem ser garantidas pelo profissional de TI que tem essa capacidade de afetar e promover a mudança.

Segundo Oteyo et al. (2021), o setor da agricultura atravessa uma grande evolução com a adoção, em grande número, de aplicações que permitem automatizar os processos agrícolas. Estas aplicações são denominadas por aplicações inteligentes de agricultura e a sua implementação é efetuada recorrendo a profissionais com experiência em programação, o que conduz a um aumento da procura por estes profissionais. Contudo, as *low-code development tools* (LCDT) são vistas como potenciais ferramentas que permitem reduzir a complexidade de desenvolvimento destas aplicações, permitindo que agricultores, sem nenhuma base de formação em programação, possam construir soluções adequadas às suas necessidades. As LCDT são centradas em interface gráficas do utilizador, em oposição ao modo tradicional de desenvolvimento de software, tornando o processo de desenvolvimento de software mais intuitivo, focando a atenção na solução e não nos aspetos técnicos. Além disso, as LCDT assumem-se, também, como ferramentas que permitem a construção e a entrega rápidas de soluções com o mínimo de esforço, quer para profissionais especializados em desenvolvimento de software, quer para profissionais de outras áreas de formação. As LCDT oferecem diferentes capacidades e recursos, e a tomada de decisão sobre qual a LCDT a usar para a resolução do problema e/ou necessidade identificada pode ser uma tarefa difícil, exaustiva e complexa. Como já mencionado anteriormente, a construção e a implementação de aplicações inteligentes de agricultura requerem competências avançadas em programação. Uma solução para este problema é o uso de LCDT, que são capazes de dar resposta a requisitos tais como: (1) a configuração das aplicações para culturas diversas, em diferentes estações do ano; (2) a alteração da recolha de dados para diferentes culturas; e (3) a utilização em aspetos agrícolas modernos. No entanto a tarefa de selecionar a LCDT mais adequada às necessidades pode ser uma tarefa difícil e trabalhosa. Tendo em conta esta questão, os autores definiram um conjunto de critérios de base para a comparação das diferentes LCP, sendo eles: (1) o domínio de foco, (2) as técnicas de desenvolvimento de soluções das aplicações, (3) o tipo de aplicações, (4) a representação da aplicação, (5) o tipo de ferramenta, (6) o suporte para APIs, (7) distribuição, e (8) reconfiguração. Após a análise de várias LCDT, a *DisCoPar* salientou-se como a plataforma que combina diferentes técnicas de desenvolvimento de soluções, além de se apresentar como uma ferramenta mais gráfica

que permite a modelação, gráficos de fluxo para representação de aplicações, recursos gráficos distribuídos, reconfiguráveis, etc.

Woo (2020) apresenta um projeto desenvolvido pelas autoridades de Nova Iorque que pretendiam rastrear a COVID-19 e fornecer serviços adequados e de forma eficiente. A velocidade de desenvolvimento da solução era um requisito chave, por isso, em cerca de três dias, um portal foi desenvolvido sem ser necessário recorrer à escrita de código e, posteriormente, a solução foi replicada noutra região. A solução foi desenvolvida com o auxílio de uma plataforma de desenvolvimento *no-code*. Estas plataformas, juntamente com as plataformas *low-code*, que não exigem praticamente conhecimento em programação, estão atualmente em grande ascensão. Alguns dos estudos considerados no artigo afirmam que este crescimento se deve à crescente digitalização dos negócios e à crescente procura por aplicações personalizadas. Uma vez que a escassez de profissionais de TI também se está a verificar, as organizações voltam-se para estas plataformas. Algumas das plataformas começam a incorporar também a IA, com o objetivo de antecipar as necessidades e os desejos do utilizador, contudo esta tarefa pode revelar-se um grande desafio. Estas tecnologias trazem consigo algumas limitações, tais como: (1) a atualização das soluções pode ser comprometida, uma vez que são desenvolvidas na sua maioria por profissionais externos à área de TI; (2) o tempo de execução das soluções pode verificar um pior desempenho e pode ser igualmente difícil integrar as soluções com outros sistemas das organizações; (3) problemas de segurança associados às soluções; (4) a dificuldade em entender o código gerado, tendo em conta que este não vai possuir comentários, e tendo em conta que as variáveis podem assumir nomes bastante incongruentes. Por fim, é ainda mencionado que, apesar da presença destas plataformas no mercado, o papel do profissional de TI não será substituído e será igualmente essencial, permitindo que estes foquem a sua atenção em tarefas mais críticas.

Strømsted, Marquard, and Heuck (2018) apresentam uma notação de processo declarativo de gráficos de *Dynamic Condition Response* (DCR) como uma proposta para o desenvolvimento *low-code*. Estes recursos permitem a criação de soluções funcionais de gestão de casos sem ser necessário recorrer à codificação tradicional. A utilização de DCR permitiu a redução da quantidade de código produzido. Ao invés da produção de um modelo de processo de negócio tradicional que serve de base aos programadores para codificarem o

sistema, a introdução de DCR permitiu a criação de um modelo de processo de negócio executável, reduzindo o tempo de codificação e a necessidade de possuir competências e experiência específica de programação para a produção do mesmo. Este processo permitiu ainda adquirir uma perceção sobre os fluxos de trabalho. A fase de desenvolvimento do modelo de processo de negócio é dividida em diversas iterações e em cada iteração, o modelo de processo é testado, e o feedback incorporado evitando um desfasamento entre os requisitos e as necessidades dos clientes, bem como evitando a redundância de tarefas e trabalho. O uso de gráficos DCR permitiu que os *stakeholders* se envolvessem no processo de desenvolvimento, focando a atenção no comportamento do processo de negócio, nos fluxos de trabalho, deixando em segundo plano questões mais técnicas. Ao longo do desenvolvimento dos sistemas, a existência e a presença de profissionais de software foi nula.

Varajão (2021) afirma que, tendo em conta aquela que é a situação atual em que vivemos caracterizada por uma elevada incerteza, imprevisibilidade e instabilidade, as tecnologias e os sistemas de informação assumem um papel fundamental para lidar com esta realidade. É descrito um projeto de desenvolvimento de software, recorrendo a tecnologia *low-code*, que foi capaz de, num curto espaço de tempo (cerca de três semanas) e envolvendo gestão ágil de projetos e rápida tomada de decisão, dar resposta a uma necessidade decorrente da situação pandémica, COVID-19. O sistema de informação desenvolvido foi considerado robusto, confiável e capaz de evoluir continuamente, incorporando as mudanças consideradas necessárias. Um dos elementos de sucesso do projeto foi a utilização de tecnologia *low-code*, que permitiu ter em consideração a volatilidade dos requisitos e a necessidade de fornecer uma solução totalmente funcional, em tempo útil. Esta tecnologia permite a criação de aplicações, recorrendo a codificação manual mínima. Além disso, é possível apresentar sistematicamente produtos mínimos viáveis (MVPs), o que possibilita a incorporação de feedback do cliente, aumentando a qualidade da resposta dada. Em último lugar, é perspetivado que o desenvolvimento de software recorrendo a *low-code*, extreme *low-code* e/ou *no-code* apoiado por tecnologias disruptivas e inovadoras como a inteligência artificial, terá uma adesão crescente e rápida, tendo uma grande influência e funcionado como catalisador da transformação digital.

Segundo a *Quidgest* (Quidgest, 2019b), a modelação e a geração automática de software, suportada por aprendizagem recursiva e processos de *machine learning*, estão a ser

consideradas como uma das áreas de maior crescimento no futuro, sendo necessário adotar soluções onde a agilidade verificada é exponencialmente maior. A engenharia de software está a sofrer transformações, onde por contraste com a abordagem dita tradicional, a utilização de modelos e da inteligência artificial tem ganho um maior destaque. A codificação manual acarreta consigo um conjunto de desvantagens, tais como: (1) perda de produtividade, (2) perda de agilidade, (3) perda de tempo, etc. Além disso, a complexidade do desenvolvimento de software tem vindo também a aumentar, tornando difícil e exaustiva. Por outro lado, os consumidores finais estão cada vez mais interessados em se envolver no processo de desenvolvimento. É realçado o atraso verificado na engenharia de software em relação à utilização de modelos, em comparação com outras engenharias, bem como o atraso identificado na aplicação da IA para o desenvolvimento de software.

A *Quidgest* (Quidgest, 2019b) realiza uma comparação entre a capacidade de um programador *code-based*, as plataformas *low-code* e a sua própria plataforma, o *Genio*, onde o programador *code-based* apenas consegue executar 16 pontos de função/mês, com as plataformas *low-code* 175 pontos de função/mês, e o *Genio* 1330 pontos de função/mês, o que reflete a rapidez e a capacidade de desenvolvimento das plataformas *low-code* e da *Genio*. Além disso, afirma que as soluções desenvolvidas são entregues ao consumidor final em 1/10 do tempo e com apenas 1/10 dos recursos que são habitualmente utilizados pelo desenvolvimento de software tradicional, de notar que esta referência não tem base científica.

Noutro estudo apresentado pela Quidgest (2019c), a mesma afirma que as plataformas *low-code* são destinadas a aplicações de menor dimensão, onde apenas 4% de todas as aplicações analisadas e desenvolvidas com *low-code* são de grande dimensão. Contudo, 75% das aplicações desenvolvidas pela sua própria plataforma, a *Genio*, correspondem a aplicações de grande dimensão. É dada uma perspetiva da indústria atual de desenvolvimento de software, que ainda exige um grande número de programadores que se tornam cada vez mais escassos, daí que a adoção das plataformas *low-code* e da *Genio* permitam o desenvolvimento de software por parte de pessoas que possuem pouco ou nenhum conhecimento e competências em programação/codificação manual, com uma rápida curva de aprendizagem associada. Por fim, a *Quidgest* não considera a sua plataforma como uma plataforma de *low-code* tradicional tendo em conta que algumas das LCP produzem código

que apenas pode ser executado nas próprias plataformas. Pelo contrário, a plataforma *Genio* produz código padrão, que pode ser utilizado e implementado fora da plataforma em que é gerado. Este aspeto traz uma grande independência face ao fornecedor da plataforma, o que nem sempre é visível nas LCP.

A *Forrester* (Rymer & Appian, 2017) em 2017 procedeu à avaliação de 13 plataformas de desenvolvimento *low-code,* avaliação essa baseada em 26 critérios. Nesse ano, já se verificava por parte dos profissionais e das organizações uma necessidade de encontrar soluções que permitissem a entrega rápida de aplicações e a inovação contínua. Tendo em conta estes fatores, as plataformas de desenvolvimento *low-code* foram encaradas como uma forma de entregar mais rapidamente soluções que conquistam, atendem e retêm as necessidades dos clientes. Um dos resultados deste relatório é que as 41 organizações de desenvolvimento e entrega de aplicações consideradas que usam LCP destacam que o seu objetivo é aumentar exponencialmente a capacidade das suas equipas responderem a tempo à elevada procura de soluções verificada. Além disso, foi ainda possível concluir que as LCP são uteis para aplicações de grande escala e complexidade, e a sua adoção contribui para a migração das organizações de desenvolvimento e entrega de aplicações para *public clouds*. Segundo a *Forrester*, as LCP podem ser definidas como produtos e/ou serviços em *cloud*, destinados ao desenvolvimento de aplicações que empregam técnicas visuais declarativas em vez de programação, e que estão disponíveis para os clientes a um custo reduzido, exigindo igualmente um reduzido tempo de formação, com os custos a aumentar na proporção do valor de negócio das plataformas.

Em 2019, a *Forrester* (Rymer & Koplowitz, 2019) apresentou outro relatório de avaliação de LCP. Desta vez a avaliação foi decomposta em 28 critérios e teve como um dos seus objetivos ajudar os profissionais a selecionar a plataforma mais adequada às suas necessidades. Tendo em conta a investigação realizada, foi possível concluir que no ano anterior ao estudo, em 2018, 23% dos responsáveis pelo desenvolvimento de soluções de software globais usaram as plataformas *low-code* e outros 22% referiram que o planeavam fazer dentro de um ano. O que despoleta estes valores é a procura das empresas digitais por software cada vez mais rápido.

Alexander (2021b) perspetiva o *low-code* como uma abordagem de desenvolvimento de software que permite a entrega mais rápida de aplicações, com codificação manual mínima.

As LCPs são um conjunto de ferramentas que permitem o desenvolvimento visual de aplicações por meio de modelação e interface gráfica, acelerando o processo tradicional de colocar uma aplicação em produção. O autor assume que o *low-code* permite resolver problemas, tais como: (1) o aumento contínuo da pressão sobre as organizações de TI para fornecer soluções inovadoras; (2) os recursos financeiros e humanos para atender a procura de mercado que são escassos para grande parte das empresas; (3) os enormes atrasos provocados pela falta de recursos humanos qualificados, que cada vez são mais solicitados a fazer mais com menos; (4) a complexidade de desenvolvimento de software, que é reduzida permitindo uma maior produtividade e velocidade do responsável pelo desenvolvimento; (5) trabalho repetitivo e desnecessário, que é restringido uma vez que a construção de MVPs é mais facilitada, tal como a incorporação do feedback do cliente, o que permite um maior alinhamento entre a solução e o cliente. Por outro lado, existe um conjunto de limitações associadas: (1) a escalabilidade da aplicação, que fica comprometida pela integração com sistemas existentes e pela utilização da aplicação em condições extremas; (2) a capacidade de atualização, que é dificultada nestas plataformas tendo em conta a dependência existente para com o fornecedor.

Alexander (2021a) faz uma reflexão sobre *low-code* e *no-code*, destacando as principais diferenças entre as mesmas e em que situação devem ser usadas. Começa por referir que o *low-code* é uma forma de os responsáveis pelo desenvolvimento projetarem aplicações rapidamente e com pouca codificação manual. O foco desta tecnologia está na criação de algo novo e valioso, sendo que os responsáveis pelo desenvolvimento não são prejudicados pela codificação repetitiva ou trabalho de duplicação. O autor enumera um conjunto de vantagens: (1) a velocidade de desenvolvimentos de aplicações e de produtos mínimos viáveis a serem apresentados aos *stakeholders*; (2) a disponibilidade de mais recursos, uma vez que qualquer pessoa pode ser responsável pelo desenvolvimento; (3) o baixo custo, pois os processos de segurança já estão integrados e podem ser facilmente personalizados. Por outro lado, aponta um conjunto de desvantagens: (1) algumas LCP não são de fácil utilização, o que pode criar constrangimentos a quem possui um conhecimento superficial sobre desenvolvimento de software; (2) a escalabilidade, o elevado desempenho e a elevada qualidade, que nem sempre são tidos em conta com o *low-code* e em muitos casos nem existe a possibilidade de os alterar. Já o *no-code* apresenta um desenvolvimento totalmente visual, destinado a pessoas que

normalmente não têm nenhum conhecimento sobre uma linguagem de programação, mas que desenvolvem ou pretendem desenvolver uma aplicação para um caso específico. Esta tecnologia é ideal para o desenvolvimento de aplicações simples e de pequena dimensão, oferece liberdade aos utilizadores para atender a uma necessidade imediata, sem desviar a atenção do departamento de TI. As plataformas *no-code* são plataformas que exigem pouca formação e qualquer pessoa pode assumir a responsabilidade pelo desenvolvimento de aplicações. No entanto, o desenvolvimento das aplicações é feito, tipicamente, sem supervisão e podem surgir questões de segurança, problemas de conformidade, problemas de integração podem fazer com que aplicações usem mais recursos do que aqueles realmente necessários.

Efetuando a comparação entre os dois tipos de tecnologia, Alexander (2021a) afirma que o *low-code* é mais destinado ao desenvolvimento de aplicações móveis e Web independentes e portais. O *no-code* deve apenas ser aplicado a casos de uso *front-end*. É possível concluir que o desenvolvimento *no-code* é mais simples que o *low-code*. No entanto, o *low-code* oferece um nível de simplicidade para colocar as aplicações em funcionamento muito satisfatório, para além de ser mais rápido que a codificação manual. Além disso, e em comparação com o *no-code*, o *low-code* requer mais conhecimento de programação, logo, tendencialmente as aplicações serão desenvolvidas de uma forma mais correta e aprimorada.

2.5. Literatura relacionada

Na literatura foram identificados três estudos de caso focados na tecnologia *low-code* e *no-code*:

- (Varajão, 2021) – um estudo de caso que envolve tecnlogia *low-code*;
- (Unqork) – dois estudos de caso que envolvem tecnologia *no-code*.

No primeiro estudo de caso (Varajão, 2021), a situação pandémica foi encarada como uma oportunidade de mercado. Inicialmente foi definido um plano para a criação de uma solução software juntamente com uma lista inicial de requisitos. A solução criada foi designada por VIRVI, um sistema de informação destinado a suportar a monitorização e o controlo de dados da pandemia. Além disso, a solução foi capaz de evoluir ao longo do tempo. Os aspetos focados neste estudo de caso foram: os pontos de função da solução, o horizonte temporal

do projeto, a constituição da equipa de trabalho, a produtividade da equipa de trabalho, a abordagem de desenvolvimento seguida, e os fatores de sucesso do projeto. No que toca aos pontos de função, inicialmente, a solução tinha 207 pontos de função, a quantidade de funcionalidades de negócio que um sistema de informação (como um produto), fornece aos utilizadores), três semanas depois, o número cresceu significativamente para 1365, o que comprova a rápida velocidade de evolução dos requisitos. O arranque do projeto ocorreu no dia 12 de março e o primeiro protótipo funcional foi apresentado no dia 22 de março a várias entidades com o intuito de recolher feedback. Três semanas após o início do projeto, a solução VIRVI estava pronta a ser lançada, o que atesta a rapidez de desenvolvimento. Outro aspeto considerado foi a equipa de projeto que, nas duas primeiras semanas, era constituída por: dois analistas/*developers* de sistemas a tempo integral, um gestor de projeto a tempo integral, um designer UX a tempo parcial (25%), um especialista em *development-technology* em tempo parcial (50%), cinco elementos da equipa de marketing a tempo parcial (25%) e quatro elementos em tempo parcial da equipa de vendas (25%). Após esse período de tempo a equipa de trabalho sofreu algumas alterações: dois analistas/*developers* de sistemas a tempo parcial (50%), um gestor de projeto a tempo parcial (50%), um especialista em *development-technology* em tempo parcial (25%), um consultor de saúde pública em tempo parcial (50%), quatro elementos da equipa de marketing a tempo parcial (5%) e quatro elementos em tempo parcial da equipa de vendas (20%). No total, na equipa de desenvolvimento houve 3.25 FTEs (*full-time equivalents*) e na de marketing e vendas 1.84 FTEs. Tendo em conta os 1365 pontos de função, a produtividade semanal dos analistas/*developers* situou-se nos 220 pontos de função. Foi seguida uma abordagem ágil de gestão de projetos tendo em conta o carácter volátil dos requisitos. A ferramenta Microsoft Teams foi utilizada para dar suporte ao processo. Um quadro *Kanban* foi construído contendo seis estados diferentes e podendo ser alterado por todos os membros da equipa: (1) tarefas em análise, (2) pendentes, (3) em execução, (4) executadas, (5) em teste e (6) concluídas. Além disso, foram também efetuadas reuniões frequentes para rever as tarefas em relação a prioridades, obsolescência e esclarecimento de requisitos. Por fim, os fatores que contribuíram para o sucesso do projeto foram: (1) o suporte da direção de topo, (2) a agilidade (na decisão e gestão), (3) a compreensão e o comprometimento da equipa do projeto, e (4) a tecnologia utilizada.

No segundo estudo de caso, foi utilizada a tecnologia *no-code* para o desenvolvimento de uma solução (Unqork). Os aspetos salientados neste estudo de caso foram: o perfil profissional do responsável pelo desenvolvimento da solução, o horizonte temporal do projeto, a integração da solução criada com serviços externos, e os benefícios obtidos através da utilização da solução. O perfil do responsável pelo desenvolvimento da solução foi totalmente funcional, sendo que a sua função profissional, dentro da organização, era como gestor de património/riqueza. Este gestor usou a tecnologia *no-code* para transformar o ciclo de vida dos seus clientes, conseguindo-o fazer em doze semanas. A área de atuação da empresa era a gestão de património/riqueza e verificava um abrandamento na geração de receita como consequência da baixa produtividade do consultor e do tempo prolongado para integrar e servir os clientes. As margens da organização estavam a ser afetadas devido ao alto custo operacional e aos riscos impulsionados por processos e controlos manuais. Tendo em conta esses fatores, o gestor utilizou a ferramenta *no-code* da *Unqork* para construir uma solução digital que fosse suficientemente abrangente para abarcar todos os processos. A solução gerada foi também integrada com sistemas de manutenção de registos e com serviços de terceiros. O tempo de implementação e de desenvolvimento da solução foram cruciais, no entanto destacaram-se outros benefícios como: (1) uma aceleração de 60% no tempo de integração dos clientes, (2) a redução do risco operacional em 70% devido à automatização, (3) a redução em 40% do custo de operações e propriedade, e (4) o aumento da receita em 20%.

O último estudo de caso considerado está relacionado com as autoridades da cidade de Nova-Iorque que pretendiam mapear a propagação da COVID-19 e fornecer serviços com eficiência aos cidadãos mais necessitados (Unqork). Os aspetos levados em consideração neste estudo de caso foram: a rapidez de desenvolvimento proporcionada pela tecnologia *no-code*, o número de soluções desenvolvidas, bem como as suas principais características e funcionalidades, e a comparação entre o desenvolvimento recorrendo a tecnologia *no-code* e as abordagens de desenvolvimento tradicionais. Tendo em conta que as abordagens tradicionais não seriam suficientemente capazes de acompanhar o ritmo acelerado da doença, a equipa de resposta à COVID-19 optou por usar a plataforma *no-code* da *Unquork* para o desenvolvimento das soluções. No total foram criados quatro portais, mais concretamente:

- *COVID-19 Engagement Portal* – construído em 72 horas, disponível em onze idiomas, este portal permitiu que os cidadãos com a doença fornecessem informações sobre si, sendo possível mapear o impacto do vírus e conectar cidadãos a serviços críticos;
- *PPE Donation Portal* – tendo em conta que as taxas de infeção cresceram rapidamente, alguns sistemas de saúde enfrentaram a escassez de recursos e, através deste portal, quer cidadãos, quer organizações, podiam fazer doações de equipamento médico;
- *GetFoodNYC Delivery Portal* – antes da pandemia, vários cidadãos beneficiavam de programas de auxílio na alimentação; com a pandemia o número de cidadãos dependentes desses programas cresceu, e a *Unqork* em união com a cidade, desenvolveu este portal que proporcionou que os motoristas da *Taxi Limousine Commission* fossem remunerados em troca da entrega de comida aos cidadãos mais vulneráveis;
- *Project Cupid* – de modo a permitir que os cidadãos pudessem estabelecer uniões reconhecidas legalmente, foi criado um centro de processamento de pedidos de casamento remotamente.

Caso os portais desenvolvidos estivessem dependentes das abordagens de desenvolvimento tradicionais, o seu ciclo de desenvolvimento seria mais extenso em termos de tempo e mais complexo.

3. FRAMEWORK DE DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO COM TECNOLOGIA LOW-CODE

Através da revisão da literatura, foi possível identificar os principais conceitos e resultados do desenvolvimento de sistemas de informação com recurso a tecnologias *low-code*, os quais se encontram organizados no *framework* apresentado na figura 1. Nas próximas secções, são abordadas as cinco dimensões do *framework*: tecnologia, *developers*, projetos, desafios e literatura.

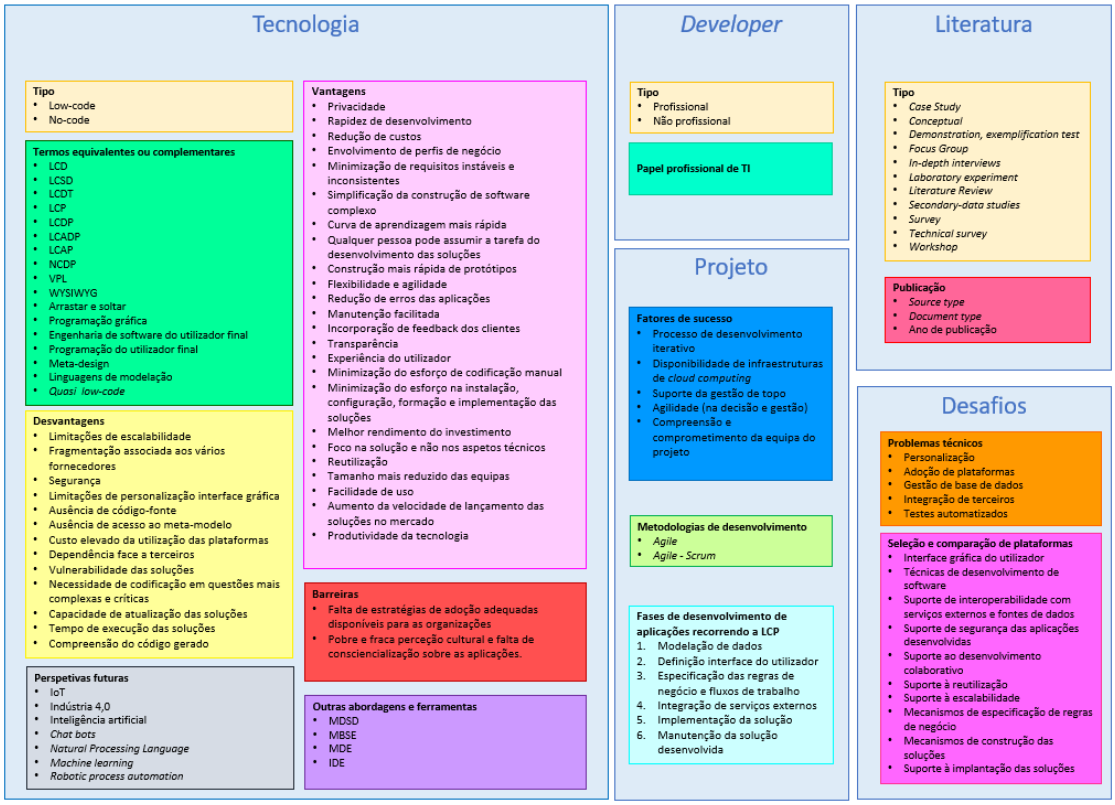

Figura 1 - Framework

3.1. Dimensão tecnologia

Na primeira dimensão, a tecnologia abordada pode ser classificada em dois tipos diferentes, tecnologia *low-code* (Al Alamin et al., 2021; Almonte et al., 2020; Alsaadi et al., 2021; Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança

et al., 2021; Butler, 2020; Cabot, 2020; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Ruscio et al., 2021; Di Sipio et al., 2020; Dushnitsky & Stroube, 2021; Henriques et al., 2018; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa & Henriques, 2018; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019; Woo, 2020) ou tecnologia *no-code* (Alsaadi et al., 2021; Bhattacharyya & Kumar, 2021; Cabot, 2020; Di Ruscio et al., 2022; ElBatanony & Succi, 2021a, 2021b; Hurlburt, 2021; Woo, 2020). Em ambos os casos, podem ser utilizados e mencionados vários termos relacionados, nomeadamente, LCD (Luo et al., 2021), LCSD (Al Alamin et al., 2021), LCDT (Oteyo et al., 2021), LCP (Alsaadi et al., 2021; Bock & Frank, 2021b), LCDP (Almonte et al., 2020; Bock & Frank, 2021b; Di Ruscio et al., 2022; Di Sipio et al., 2020; Khorram et al., 2020; Overeem & Jansen, 2021; Philippe et al., 2020; Sahinaslan et al., 2021; Silva et al., 2021), LCADP (Arora et al., 2020; Sahay, Di Ruscio, et al., 2020), LCAP (Bragança et al., 2021), NCDP (Bock & Frank, 2021b; Di Ruscio et al., 2022), VPL (Ragusa & Henriques, 2018), WYSIWYG (Luo et al., 2021), Arrastar e soltar (Luo et al., 2021), Programação gráfica (Luo et al., 2021), Engenharia de software do utilizador final (Alsaadi et al., 2021; Silva et al., 2021), Programação do utilizador final (Alsaadi et al., 2021; Silva et al., 2021), Meta-design (Alsaadi et al., 2021; Silva et al., 2021), Linguagens de modelação (Henriques et al., 2018), *Quasi low-code* (Varajão, 2021).

A tecnologia apresenta diversas vantagens, tais como: a privacidade (Sanchis et al., 2020), a rapidez de desenvolvimento (Al Alamin et al., 2021; Alsaadi et al., 2021; Arora et al., 2020; Butler, 2020; Cabot, 2020; Di Ruscio et al., 2022; Di Sipio et al., 2020; ElBatanony & Succi, 2021a; Horváth et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Luo et al., 2021; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Sanchis et al., 2020; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), a redução de custos (Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Sipio et al., 2020; Dushnitsky & Stroube, 2021; ElBatanony & Succi, 2021a; Henriques et al., 2018; Khorram et al., 2020; Luo et al., 2021; Metrôlho et al., 2020; Sanchis et al., 2020; Waszkowski, 2019), o envolvimento de perfis de negócio (Sanchis et al.,

2020; Strømsted et al., 2018), a minimização de requisitos instáveis e inconsistentes (Sanchis et al., 2020; Varajão, 2021), a simplificação da construção de software complexo (da Cruz et al., 2021; Oteyo et al., 2021), uma curva de aprendizagem mais rápida (da Cruz et al., 2021), qualquer pessoa pode assumir a tarefa do responsável pelo desenvolvimento das soluções (Almonte et al., 2020; Alsaadi et al., 2021; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a; Bragança et al., 2021; da Cruz et al., 2021; Di Ruscio et al., 2021; Di Sipio et al., 2020; ElBatanony & Succi, 2021a, 2021b; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Woo, 2020), a construção mais rápida de protótipos (da Cruz et al., 2021), a flexibilidade e a agilidade (Al Alamin et al., 2021; Luo et al., 2021), a redução de erros das aplicações (Al Alamin et al., 2021), manutenção facilitada (Al Alamin et al., 2021; Bock & Frank, 2021b; Metrôlho et al., 2020), incorporação de feedback dos clientes (Al Alamin et al., 2021; da Cruz et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), transparência (Bock & Frank, 2021b; Sahinaslan et al., 2021), experiência do utilizador (Bock & Frank, 2021a; da Cruz et al., 2021; Di Ruscio et al., 2022; Henriques et al., 2018; Luo et al., 2021; Sahinaslan et al., 2021), a minimização do esforço de codificação manual (Al Alamin et al., 2021; Alsaadi et al., 2021; Arora et al., 2020; Bock & Frank, 2021a; Butler, 2020; Cabot, 2020; Di Ruscio et al., 2022; Di Ruscio et al., 2021; Dushnitsky & Stroube, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Silva et al., 2021; Varajão, 2021; Waszkowski, 2019), a minimização do esforço na instalação, configuração, formação e implementação das soluções (Di Ruscio et al., 2022; Luo et al., 2021; Waszkowski, 2019), um melhor rendimento do investimento (Arora et al., 2020), o foco na solução e não nos aspetos técnicos (Bragança et al., 2021; Henriques et al., 2018; Kukesh & Brandenburg, 2018; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahay, Indamutsa, et al., 2020; Strømsted et al., 2018; Waszkowski, 2019), a reutilização (Bock & Frank, 2021a, 2021b; Bragança et al., 2021; Lourenço et al., 2021), o tamanho mais reduzido das equipas (Alsaadi et al., 2021; ElBatanony & Succi, 2021a), a facilidade de uso (Bragança et al., 2021; Luo et al., 2021), o aumento da velocidade de lançamento das soluções no mercado (Bhattacharyya & Kumar, 2021; Butler, 2020; da Cruz et al., 2021; ElBatanony & Succi, 2021a; Horváth et al., 2020; Pichidtienthum et al., 2021), e a produtividade associada à tecnologia

low-code (Bock & Frank, 2021a, 2021b; Lourenço et al., 2021; Silva et al., 2021). São igualmente identificadas desvantagens: limitações de escalabilidade (Alsaadi et al., 2021; Sanchis et al., 2020), a fragmentação associada aos vários fornecedores (cada um com um modelo de programação exclusivo) (Sanchis et al., 2020), a segurança (Alsaadi et al., 2021; da Cruz et al., 2021; Sanchis et al., 2020; Woo, 2020), limitações de personalização da interface gráfica (da Cruz et al., 2021; Luo et al., 2021), a ausência de código-fonte (Bragança et al., 2021; Butler, 2020; da Cruz et al., 2021), a ausência de acesso ao meta-modelo (Bragança et al., 2021), o custo elevado da utilização das plataformas (da Cruz et al., 2021; Luo et al., 2021), a dependência face a terceiros (da Cruz et al., 2021; Luo et al., 2021; Woo, 2020), a vulnerabilidade das soluções causada pelos responsáveis pelo desenvolvimento (da Cruz et al., 2021; Di Sipio et al., 2020), a necessidade de codificação em questões mais complexas e críticas (Cabot, 2020; Luo et al., 2021), a capacidade de atualização das soluções (Woo, 2020), o tempo de execução das soluções (Woo, 2020), e a compreensão do código gerado (Alsaadi et al., 2021; Woo, 2020).

Foram associadas à tecnologia outras abordagens e ferramentas como MDSD (da Cruz et al., 2021), MBSE (Horváth et al., 2020), MDE (Almonte et al., 2020; Di Ruscio et al., 2022; Ihirwe et al., 2020), IDE (Almonte et al., 2020; da Cruz et al., 2021). Alguns estudos apontam algumas barreiras ao uso da tecnologia: a falta de estratégias de adoção adequadas disponíveis para as organizações (Bhattacharyya & Kumar, 2021), a pobre e a fraca perceção cultural e a falta de consciencialização sobre as tecnologias (Bhattacharyya & Kumar, 2021).

Por fim, são referidas as perspetivas futuras de evolução da tecnologia em diversas áreas: IoT (Di Ruscio et al., 2022; Dushnitsky & Stroube, 2021; Ihirwe et al., 2020; Overeem & Jansen, 2021; Sahinaslan et al., 2021; Sanchis et al., 2020), Indústria 4.0 (Sahinaslan et al., 2021; Sanchis et al., 2020), Inteligência artificial (ElBatanony & Succi, 2021b; Hurlburt, 2021; Varajão, 2021; Woo, 2020), *Chat bots* (Di Ruscio et al., 2022; ElBatanony & Succi, 2021b), *Natural Processing Language* (ElBatanony & Succi, 2021b), *Machine learning* (Hurlburt, 2021; Ihirwe et al., 2020), *Robotic Process Automation* (Hurlburt, 2021).

Na figura 2 é apresentado o número de publicações que suportam os elementos que compõem a dimensão *tecnologia* do *framework*.

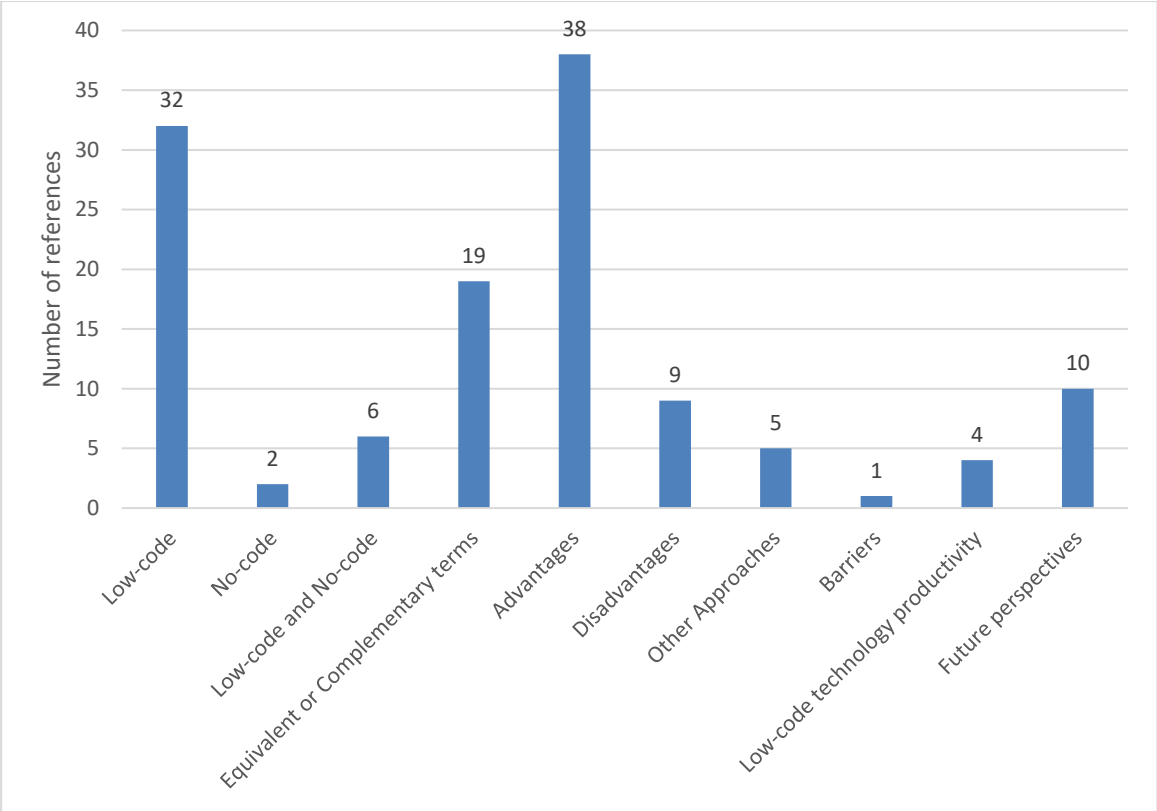

Figura 2 - Dimensão Tecnologia

Após a análise das publicações é possível concluir que 32 publicações abordam a tecnologia *low-code* exclusivamente, e apenas duas abordam a tecnologia *no-code* também de forma exclusiva. As publicações que abordam os dois tipos de tecnologia são seis no seu total.

Além disso, 19 publicações utilizam termos equivalentes ou complementares para se referirem a *low-code*, 38 elencam um conjunto de vantagens e um grupo mais restrito de publicações (apenas nove publicações) aponta um conjunto de desvantagens.

No que se refere a outras abordagens, cinco publicações fazem referência e estabelecem pontos divergentes entre a tecnologia *low-code/no-code* e outras abordagens. As barreiras à adoção foram identificadas apenas em uma publicação e a produtividade da tecnologia *low-code* é realçada em quatro publicações.

Em último lugar, as perspetivas futuras foram apresentadas em dez publicações.

3.2. Dimensão projeto

Na segunda dimensão são destacados os vários aspetos ligados aos projetos de desenvolvimento de soluções com recurso à tecnologia *low-code*. Em primeiro lugar, são abordados os fatores de sucesso: (1) processo de desenvolvimento iterativo (Butler, 2020), (2) disponibilidade de infraestruturas de *cloud computing* (Di Ruscio et al., 2022; Di Ruscio et al., 2021), (3) suporte da gestão de topo (Varajão, 2021), (4) agilidade (na decisão e gestão) (Varajão, 2021), e (5) compreensão e comprometimento da equipa do projeto (Varajão, 2021).

As metodologias de desenvolvimento, como *Agile* (Al Alamin et al., 2021; Kukesh & Brandenburg, 2018; Sahay, Indamutsa, et al., 2020; Varajão, 2021) e *Scrum* (Sahay, Indamutsa, et al., 2020) são apontadas como as mais comuns no desenvolvimento de soluções recorrendo a LCP.

As fases de desenvolvimento de aplicações recorrendo a LCP são igualmente enumeradas em: (1) Modelação de dados (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020), (2) Definição da interface do utilizador (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020), (3) Especificação das regras de negócio e fluxos de trabalho (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020), (4) Integração de serviços externos (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020), (5) Implementação da solução (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020) e (6) Manutenção da solução desenvolvida (Di Ruscio et al., 2022).

Na figura 3 é apresentado o número de publicações que suportam os elementos que compõem a dimensão *projeto* do *framework*.

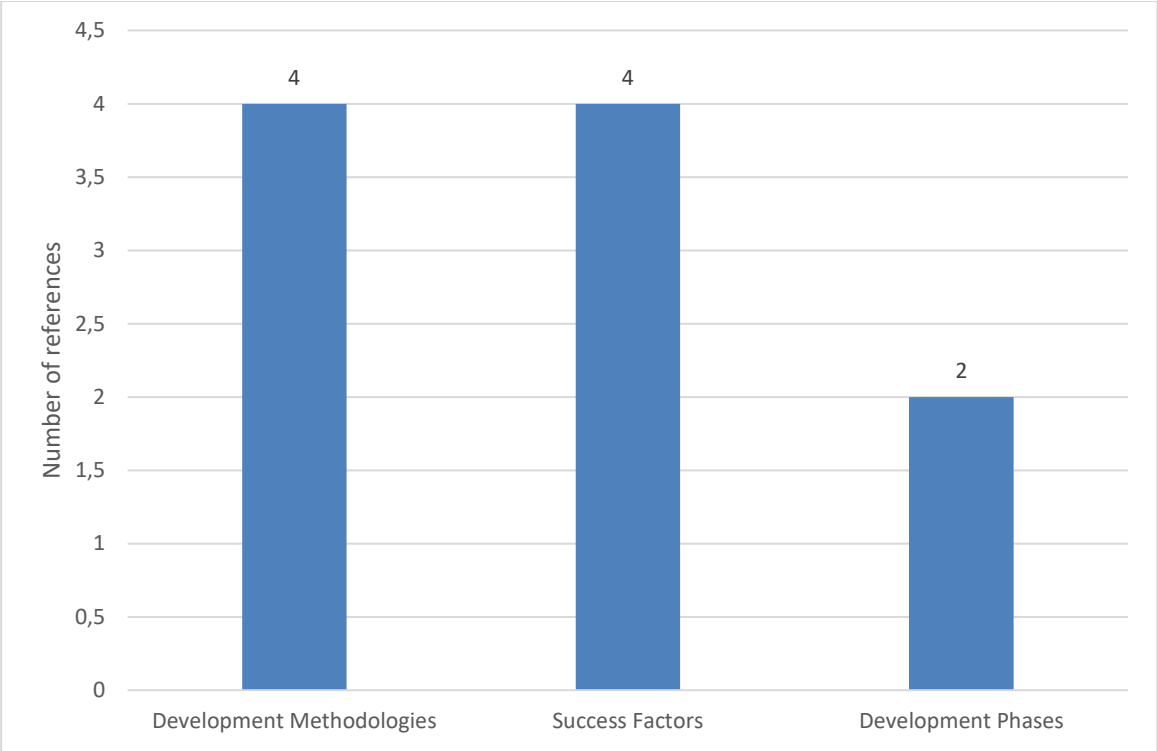

Figura 3 - Dimensão Projeto

No que toca às metodologias de desenvolvimento, existem quatro publicações que as mencionam. Os fatores de sucesso da tecnologia são salientados em quatro publicações.

As fases de desenvolvimento de aplicações recorrendo a LCP foram destacadas somente em duas publicações.

3.3. Dimensão developer

A terceira dimensão apresenta os vários elementos relacionados com *developers* que utilizam a tecnologia *low-code/no-code* para o desenvolvimento das suas soluções. Em primeiro lugar, é possível dividir os *developers* em dois tipos, os *developers* profissionais, cuja sua área de formação está diretamente relacionada com as TI e os SI (Al Alamin et al., 2021; Almonte et al., 2020; Alsaadi et al., 2021; Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança et al., 2021; Cabot, 2020; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Sipio et al., 2020; Henriques et al., 2018; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa &

Henriques, 2018; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Silva et al., 2021; Varajão, 2021; Waszkowski, 2019; Woo, 2020), e os *developers* não profissionais, cuja sua formação de base não contempla as áreas das TI e dos SI (Al Alamin et al., 2021; Almonte et al., 2020; Alsaadi et al., 2021; Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança et al., 2021; Butler, 2020; Cabot, 2020; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Ruscio et al., 2021; Di Sipio et al., 2020; Dushnitsky & Stroube, 2021; ElBatanony & Succi, 2021a, 2021b; Henriques et al., 2018; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa & Henriques, 2018; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019; Woo, 2020).

O papel do profissional de TI é abordado em algumas publicações, sendo que todas são unânimes em afirmar que o papel do profissional de TI não está ameaçado pela utilização desta tecnologia, pelo contrário, pode ser benéfico já que este se pode focar em tarefas mais críticas e ser capaz de continuar a acrescentar valor para a organização (ElBatanony & Succi, 2021a; Hurlburt, 2021; Woo, 2020).

Na figura 4 é apresentado o número de publicações que suportam os elementos que compõem a dimensão *developer* do *framework*.

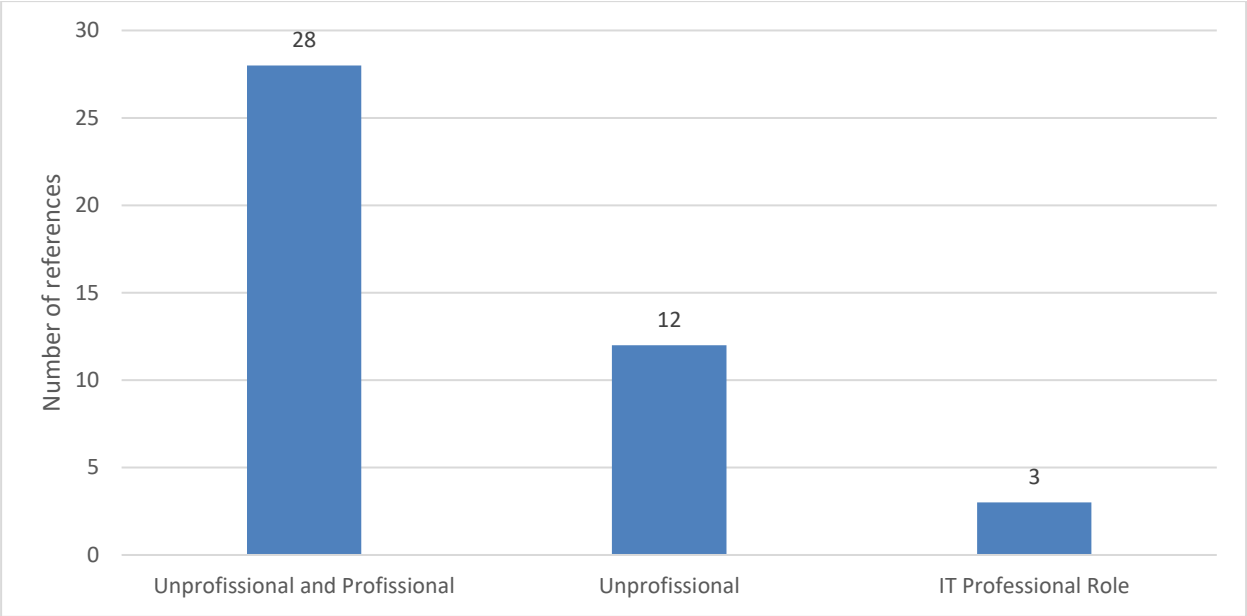

Figura 4 - Dimensão Developer

28 publicações focaram a sua atenção em profissionais na área das TI e dos SI e profissionais de outras áreas de formação, e 12 publicações restringiram o seu foco apenas a profissionais provenientes de outras áreas de formação.

O papel do profissional de TI dentro das organizações é destacado em três publicações.

3.4. Dimensão desafios

A quarta dimensão contempla os desafios associados ao desenvolvimento de soluções com recurso às LCP. Os problemas técnicos com os quais os *developers* se deparam no desenvolvimento das suas soluções recorrendo às LCP são, mais especificamente, a personalização (Al Alamin et al., 2021; da Cruz et al., 2021; Luo et al., 2021), a adoção de plataformas (Al Alamin et al., 2021), a gestão de base de dados (Al Alamin et al., 2021), a integração de terceiros (Al Alamin et al., 2021), os testes automatizados (Al Alamin et al., 2021) e a revisão do código (Henriques et al., 2018).

A seleção e comparação de plataformas LCP é outra questão que inquieta os *developers*, e como já mencionado anteriormente, pode-se tornar uma tarefa difícil e complexa. Assim sendo, existem fatores que devem ser tidos em conta, tais como: a interface gráfica do utilizador (Sahay, Indamutsa, et al., 2020), as técnicas de desenvolvimento de software (Oteyo et al., 2021), o suporte de interoperabilidade com serviços externos e fontes de dados (Oteyo

et al., 2021; Sahay, Indamutsa, et al., 2020), o suporte de segurança das aplicações desenvolvidas (Sahay, Indamutsa, et al., 2020), o suporte ao desenvolvimento colaborativo (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020), o suporte à reutilização (Sahay, Indamutsa, et al., 2020), o suporte à escalabilidade (Sahay, Indamutsa, et al., 2020), os mecanismos de especificação de regras de negócio (Sahay, Indamutsa, et al., 2020), os mecanismos de construção das soluções (Oteyo et al., 2021; Sahay, Indamutsa, et al., 2020), e o suporte à implantação das soluções (Oteyo et al., 2021; Sahay, Indamutsa, et al., 2020).

Na figura 5 é apresentado o número de publicações que supram os elementos que compõem a dimensão *desafios* do *framework*.

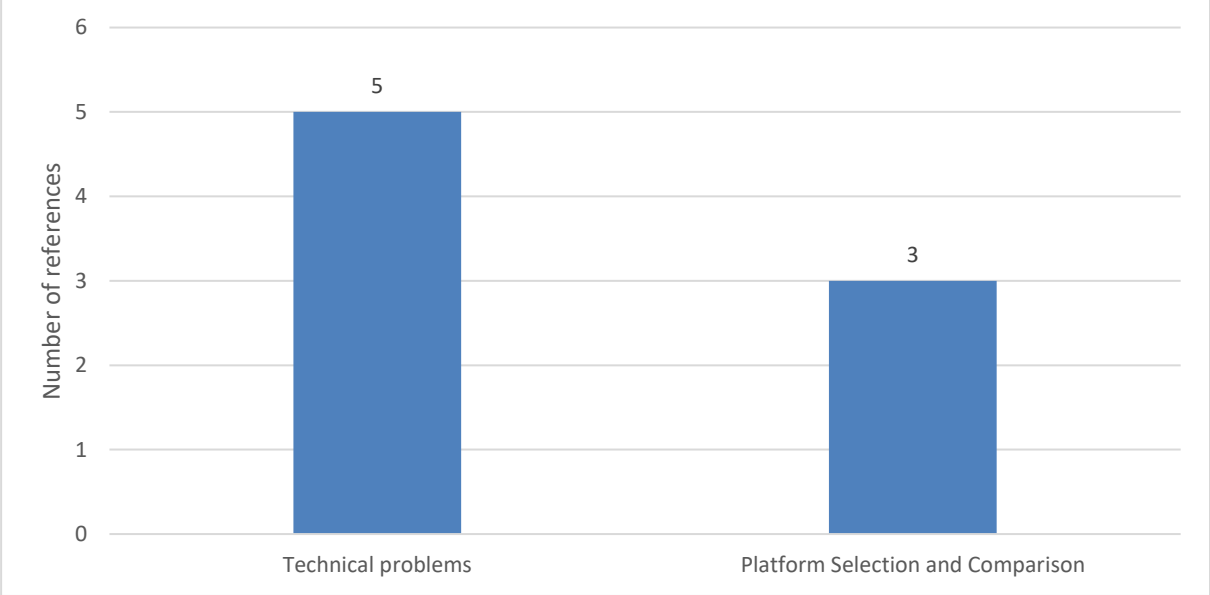

Figura 5 - Dimensão Desafios

No que diz respeito aos problemas técnicos apontados pelos profissionais, apenas cinco publicações fazem referência aos mesmos.

A tarefa de seleção e comparação de plataformas, vista como uma das tarefas mais importantes, é mencionado exclusivamente em três publicações.

3.5. Dimensão literatura

A última dimensão está relacionada com os estudos. Num primeiro momento o tipo de literatura é distribuído por 11 categorias: *Case Study* (Bragança et al., 2021; Metrôlho et al., 2020; Overeem & Jansen, 2021; Sahinaslan et al., 2021; Sanchis et al., 2020; Strømsted et al., 2018; Varajão, 2021; Woo, 2020)*, Conceptual* (Bock & Frank, 2021a, 2021b; Cabot, 2020; ElBatanony & Succi, 2021b; Horváth et al., 2020; Hurlburt, 2021; Khorram et al., 2020; Philippe et al., 2020; Sahay, Di Ruscio, et al., 2020; Waszkowski, 2019; Woo, 2020), *Demonstration and/or exemplification test* (da Cruz et al., 2021; Di Sipio et al., 2020; Henriques et al., 2018; Lourenço et al., 2021; Oteyo et al., 2021; Ragusa & Henriques, 2018)*, Focus Group* (Di Ruscio et al., 2022)*, In-depth Interviews* (Bhattacharyya & Kumar, 2021)*, Laboratory Experiment* (Almonte et al., 2020; Pichidtienthum et al., 2021; Silva et al., 2021), *Literature Review* (Ihirwe et al., 2020)*, Secondary-data studies* (Al Alamin et al., 2021; Dushnitsky & Stroube, 2021; ElBatanony & Succi, 2021a; Luo et al., 2021)*, Survey* (Alsaadi et al., 2021; Bock & Frank, 2021b; Henriques et al., 2018)*, Technical Survey* (Sahay, Indamutsa, et al., 2020), Workshop (Arora et al., 2020; Di Ruscio et al., 2021).

Além disso, também são considerados o *source type*, *Conference Proceedings* (Al Alamin et al., 2021; Almonte et al., 2020; Arora et al., 2020; Bock & Frank, 2021a; Bragança et al., 2021; Cabot, 2020; Di Ruscio et al., 2021; Di Sipio et al., 2020; ElBatanony & Succi, 2021a, 2021b; Henriques et al., 2018; Horváth et al., 2020; Ihirwe et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa & Henriques, 2018; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Strømsted et al., 2018; Waszkowski, 2019) e *Journal* (Alsaadi et al., 2021; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021b; da Cruz et al., 2021; Di Ruscio et al., 2022; Dushnitsky & Stroube, 2021; Hurlburt, 2021; Kukesh & Brandenburg, 2018; Sanchis et al., 2020; Silva et al., 2021; Varajão, 2021; Woo, 2020), e o *document type, Conference Paper* (Al Alamin et al., 2021; Almonte et al., 2020; Arora et al., 2020; Bock & Frank, 2021a; Bragança et al., 2021; Cabot, 2020; Di Ruscio et al., 2021; Di Sipio et al., 2020; ElBatanony & Succi, 2021a, 2021b; Horváth et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Overeem & Jansen, 2021; Pichidtienthum et al., 2021; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Waszkowski, 2019)

(Henriques et al., 2018; Ihirwe et al., 2020; Metrôlho et al., 2020; Oteyo et al., 2021; Philippe et al., 2020; Ragusa & Henriques, 2018; Strømsted et al., 2018) e *Article.* (Alsaadi et al., 2021; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021b; da Cruz et al., 2021; Di Ruscio et al., 2022; Dushnitsky & Stroube, 2021; Hurlburt, 2021; Sanchis et al., 2020; Silva et al., 2021; Varajão, 2021; Woo, 2020).

Por fim, é ainda identificado o ano correspondente a cada publicação, 2018 (Henriques et al., 2018; Kukesh & Brandenburg, 2018; Ragusa & Henriques, 2018; Strømsted et al., 2018), 2019 (Waszkowski, 2019), 2020 (Almonte et al., 2020; Arora et al., 2020; Butler, 2020; Cabot, 2020; Di Sipio et al., 2020; Horváth et al., 2020; Ihirwe et al., 2020; Khorram et al., 2020; Metrôlho et al., 2020; Philippe et al., 2020; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sanchis et al., 2020; Woo, 2020), 2021 (Al Alamin et al., 2021; Alsaadi et al., 2021; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança et al., 2021; da Cruz et al., 2021; Di Ruscio et al., 2021; Dushnitsky & Stroube, 2021; ElBatanony & Succi, 2021a, 2021b; Hurlburt, 2021; Lourenço et al., 2021; Luo et al., 2021; Oteyo et al., 2021; Overeem & Jansen, 2021; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Silva et al., 2021; Varajão, 2021), 2022 (Di Ruscio et al., 2022).

Na figura 6 é apresentado o número de publicações que suportam o elemento *type of literature*, que compõe a dimensão *literatura* do *framework*.

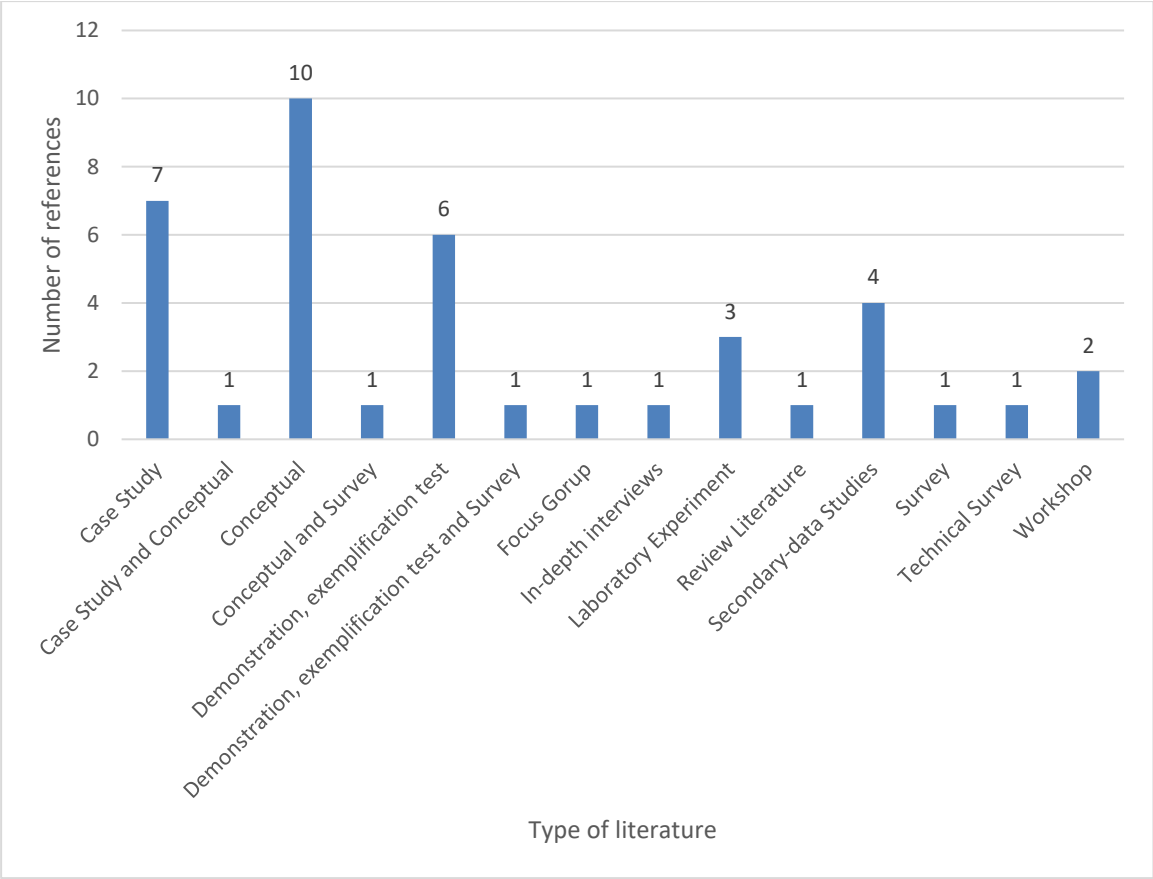

Figura 6 - Dimensão Literatura (1)

Na figura 7 é apresentado o número de publicações que suporta os elementos *source type* e *document type* que compõem a dimensão *literatura* do *framework*.

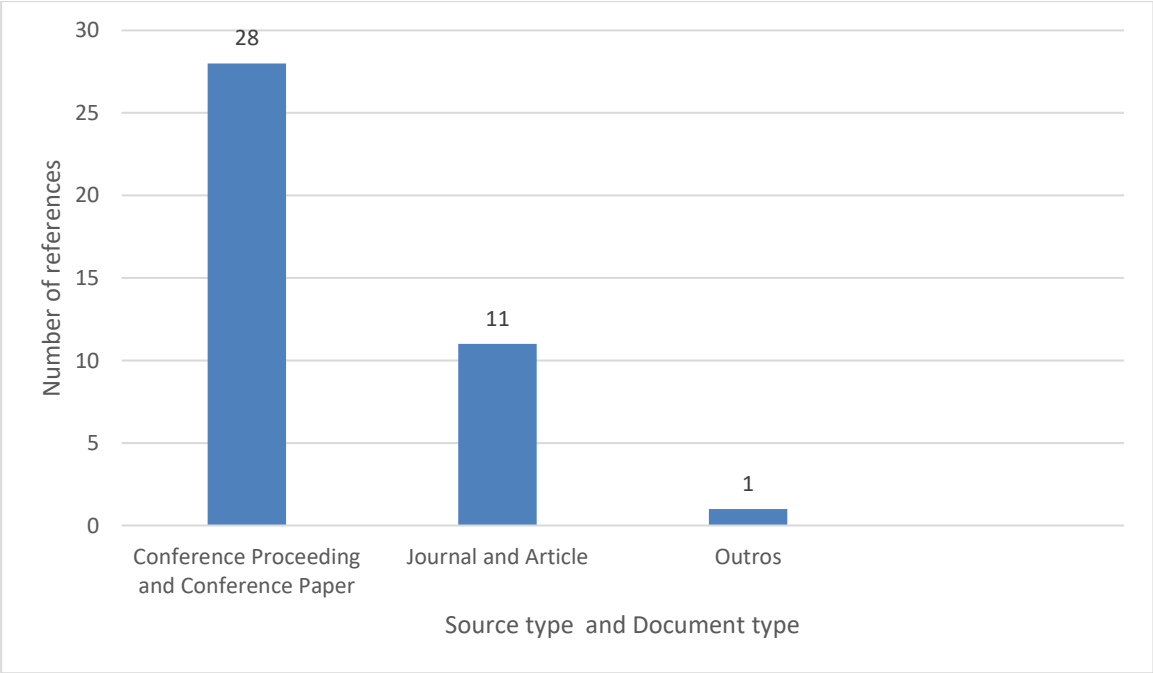

Figura 7 - Dimensão Literatura (2)

Na figura 8 é apresentado o número de publicações que suportam o elemento ano, que compõe a dimensão *literatura* do *framework*.

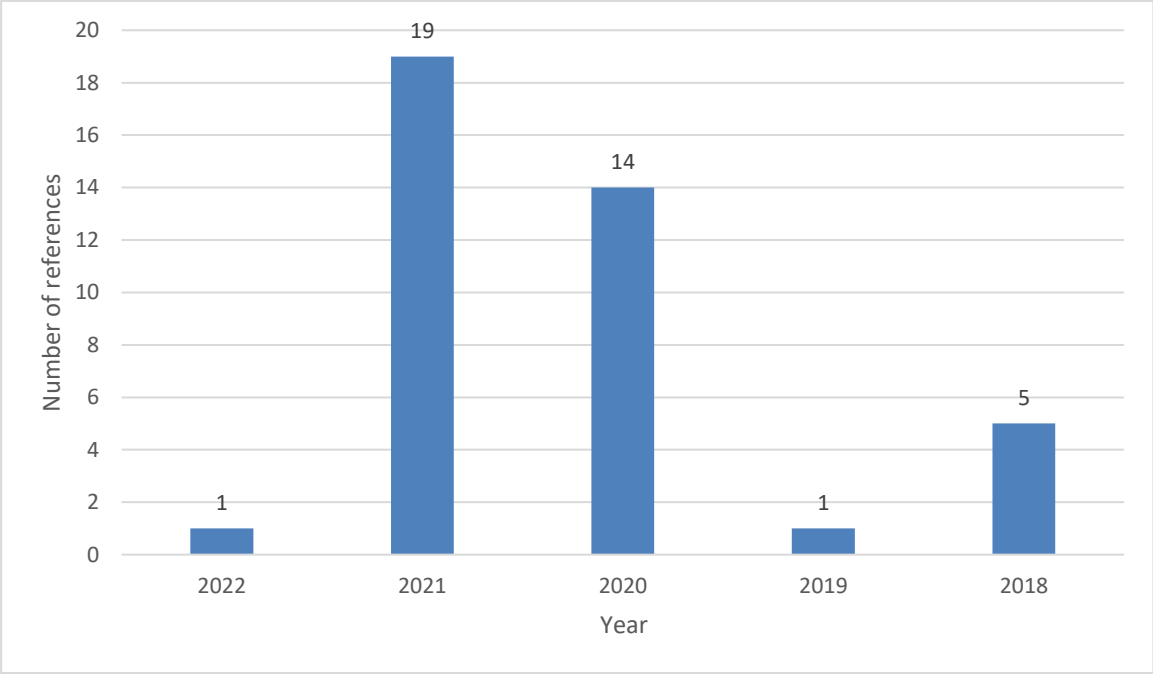

Figura 8 - Dimensão Literatura (3)

Sete publicações foram classificadas como *Case Study*, uma como *Case Study and Conceptua*l, dez como *Conceptual*, sendo este o tipo de literatura com mais publicações, uma *Conceptual and Survey*, seis como *Demonstration, exemplification test*, uma *Demonstration, exemplification test and Survey*, uma como *Focus Group*, uma *In-depth interviews*, três como *Laboratory Experiment*, uma como *Review Literature*, quatro como *Secondary-data Studies*, uma *Survey*, uma *Technical Survey* e dois Workshops.

No que toca ao *source type* e ao *document type*, os valores refletem a ligação que existe entre estes dois parâmetros, neste contexto específico, em que caso o *source type* de uma publicação fosse identificado como *Conference Proceeding*, o seu *document type* seria *Conference Paper*, caso o *source type* fosse *Journal*, o *document type* seria *Article*. Daí o resultado de 28 *Conference Proceding* e *Conference Paper,* 11 *Journals* e *Articles*.

O último ponto a ter em conta foi o ano das publicações. Grande parte da literatura considerada foi publicada entre 2021 e 2020, com 19 publicações e 14 publicações

respetivamente. O ano que se seguiu foi o ano de 2018 com 5 publicações. Os dois anos com menos publicações associadas é o ano de 2022 e de 2019 com apenas 1 publicação cada um. Nota-se, assim, um interesse crescente da comunidade científica neste fenómeno.

4. MÉTODO DE INVESTIGAÇÃO

Os métodos de investigação estão na base da produção de conhecimento em qualquer área e são capazes de moldar a linguagem que usamos para descrever o mundo, e por sua vez, a linguagem molda a forma como pensamos o mundo (Dubé & Paré, 2003). Nesta secção, o método de investigação adotado (Estudo de caso), é descrito à luz da literatura existente, bem como apresentada a razão pela qual este método foi escolhido e considerado o mais adequado. Também é descrito o estudo realizado.

4.1. Descrição e justificação do método de investigação adotado

Yin (2003) define Estudo de caso (*Case Study*) como sendo uma investigação empírica que aborda um fenómeno contemporâneo, dentro do seu contexto da vida real, especialmente quando os limites entre o fenómeno e o contexto não são evidentes.

Segundo Paré (2004), a investigação recorrendo ao método Estudo de caso e a sua aceitação na área dos SI e das TI tem vindo a aumentar.

Afirma também que este método de investigação é útil quando um fenómeno é amplo e complexo, quando o corpo de conhecimento existente é reduzido, quando é necessária uma investigação holística e aprofundada, e quando um fenómeno não pode ser estudado fora do contexto em que ocorre.

A utilização do método de investigação Estudo de caso é bastante frequente nas áreas dos SI e das TSI, tendo em conta que os objetos de investigação são os sistemas de informação nas organizações (Benbasat, Goldstein, & Mead, 1987).

O Estudo de caso permitir capturar a dinâmica e a mutualidade dos SI e das TSI, e o seu impacto nas organizações, permite construir um melhor entendimento das várias interações entre organizações, tecnologias e pessoas, permite identificar novas oportunidades, desafios e problemas enfrentados pelos profissionais de TI e pelas organizações, e permite definir hipóteses, testar hipóteses e encontrar explicações.

Paré (2004) apresenta uma estrutura a seguir para a realização de um Estudo de caso (*Case Study*), estando esta dividida em quatro etapas diferentes [1]:

1. *Design of the Case Study;*
2. *Conduct of the Case Study;*
3. *Analysis of the Case Study evidence;*
4. *Writing up the Case Study report;*

Na primeira etapa, *Design of the Case Study*, é fundamental definir inicialmente as questões de investigação, especificar a priori os constructos ou teoria, definir a unidade de análise, definir o número de casos e proceder à sua seleção, e utilizar um protocolo.

As questões de investigações devem possuir determinadas qualidades, nomeadamente, serem claras, simples, óbvias, viáveis tendo em conta o tempo e o número de recursos disponíveis, oportunas, cientificamente relevantes, etc.

No que toca à especificação prévia da teoria, esta pode ser realizada seguindo duas abordagens diferentes. Uma em que o investigador efetua uma seleção de conceitos e relações entre eles, outra em que investigador não fica limitado pela teorização anterior e olha para o desenvolvimento de teorias, hipóteses e conceitos relevantes como um propósito do projeto.

A definição da unidade de análise deve ter em conta os seguintes fatores: cada unidade de análise deve ser o mais específica possível, cada caso deve ser encarado como um sistema limitado, cada unidade de análise deve estar intimamente relacionada com as questões de investigação, e a literatura deve ser utilizada como input.

A decisão sobre o número de casos a considerar e a selecionar, deve ser feita de forma a maximizar o potencial da investigação, no período de tempo disponível, o que significa que várias estratégias podem ser seguidas:

- Caso crítico – esta estratégia permite a generalização lógica e a aplicação da informação a outros casos;
- *Theory-based* – onde se procura encontrar exemplos de uma construção teórica, procedendo à sua elaboração e exame;

[1] Optou-se por manter a expressão original em inglês, de modo a não se perder significado com a tradução.

- *Confirming and disconfirming cases* – onde são elaboradas análises iniciais com o objetivo de encontrar variações e exceções;
- Caso extremo ou divergente – esta estratégia tem como propósito retirar conclusões a parir de manifestações altamente incomuns do fenómeno;
- Caso típico – onde se pretende verificar o que é comum, normal e médio;
- Intensidade – em que são abordados casos ricos em informação que manifestam o fenómeno de forma intensa;
- Critério – onde os casos selecionados têm em consideração algum critério, o que pode ser útil para garantir a qualidade;
- Conveniência – esta estratégia permite economizar tempo, esforço e recursos em troca da perda de informação e de credibilidade.

Um protocolo de um Estudo de caso deve conter quatro componentes:

1. Uma visão abrangente do Estudo de caso (objetivos, questões de investigação, etc.);
2. Procedimentos de campo (credenciais, acesso a sites, fontes de informação);
3. Guias de entrevistas e/ou instrumentos de investigação;
4. Um guia para a construção do relatório do Estudo de caso.

A segunda etapa, *Conduct of the Case Study*, inicia-se com os métodos de recolha de dados, onde geralmente são combinados vários métodos, podendo ser recolhidos vários dados de diferentes fontes.

No que diz respeito às fontes de informação, é possível distingui-las em seis tipos diferentes (Yin, 2003):

- Documentação – cartas, documentos administrativos, artigos de jornais, relatórios anuais, boletins informativos, etc.;
- Registos de arquivo – documentos de serviço, registos organizacionais, listas de nomes, dados de investigação, etc.;
- Entrevistas (que são o elemento mais comum na maioria dos Estudos de Caso);
- Observação direta – que ocorre quando são realizadas atividades de campo durante o Estudo de caso;

- Observação participante – onde o investigador assume um papel ativo nos eventos que são estudados;
- Artefactos físicos – ferramentas, instrumentos ou outro elemento que pode ser recolhido no estudo de campo.

A triangulação de dados, que é o passo que se segue, permite que múltiplas perceções sejam utilizadas para esclarecer o significado, é realizada com o objetivo de encontrar repetições de observações e/ou de interpretações.

Um Estudo de caso não necessita de ter um processo de recolha de dados altamente intensivo, uma vez que nem sempre a quantidade se traduz em qualidade. O ponto ideal a atingir é a saturação teórica, onde a partir de determinado ponto, os dados qualitativos não acrescentam valor para o conhecimento.

A etapa 3, *Analysis of the Case Study evidence*, encontra-se dividida em três passos: *Early Steps in Data Analysis, Within-Case Analysis e Cross Case Analysis*.

No primeiro passo, é realizada uma análise dos dados qualitativa dos dados onde é possível utilizar várias técnicas, sendo destacadas duas por Paré (2004): a codificação e o desenvolvimento de uma base de dados do Estudo de caso.

A codificação permite a redução e a organização dos dados recolhidos. Alguns investigadores utilizam códigos pré-definidos, geralmente baseados em investigações anteriores ou considerações, teóricas. Outros preferem desenvolver os códigos somente após alguma exploração inicial dos dados, e em alguns casos é utilizada uma versão intermédia das duas perspetivas anteriores.

A codificação permite, assim, organizar segmentos de texto semelhantes ou relacionados para uma melhor interpretação.

O desenvolvimento de uma base de dados do Estudo de caso permite a separação entre os dados recolhidos e as interpretações feitas pelo investigador. Este repositório de dados pode conter desde dados recolhidos, a notas de campo, documentos, materiais criados pelos investigadores, considerações feitas pelo investigador, etc.

O passo *Within-Case Analysis*, envolve a adoção de um modo dominante de análise de dados, o uso de representações visuais e as revisões do projeto.

As duas estratégias mais adotadas na análise dos dados são a correspondência de padrões e a construção de explicações. Na correspondência de padrões, é comparado um padrão

empírico com um previsto, onde a validade é reforçada caso os padrões coincidam. Na construção de explicações é desenvolvida uma descrição do caso que permite estruturar o Estudo de caso. As representações visuais são outro elemento na análise qualitativa onde se pretende descobrir conexões entre segmentos codificados através de gráficos, matrizes e figuras. Nas revisões do projeto, o investigador recolhe as opiniões de pessoas que estão envolvidas no Estudo de caso, para aumentar e comprovar a credibilidade das interpretações e das descobertas alcançadas.

No último passo, *Cross Case Analysis*, procura-se encontrar padrões entre casos, permitindo aumentar a generalização, a relevância e a aplicabilidade das descobertas a outras configurações semelhantes, aprofundar a compreensão e a explicação do caso.

A última etapa, *Writing up the Case Study report*, concentra-se no aspeto do relatório do Estudo de caso e o quão importante ele é do ponto de vista do leitor. Tendo por base a literatura, existem quatro critérios a ter em consideração e que abordam a qualidade dos relatórios (Paré, 2004): a ressonância, a retórica, a capacitação e a aplicabilidade.

Os critérios de ressonância medem o grau de ajuste entre o relatório do Estudo de caso e o conjunto de convicções que sustentam o paradigma que o investigador escolheu seguir. Os critérios retóricos focam-se na avaliação da forma, estrutura e características de apresentação do Estudo de caso, tais como, a unidade, a coerência, a simplicidade e clareza. Os critérios de capacitação avaliam a capacidade de o Estudo de caso provocar e facilitar a ação por parte do leitor. Estes critérios incluem a justiça, a educação, a capacidade de ação, etc. Os critérios de aplicabilidade aferem a capacidade de o estudo de caso facilitar a construção de inferências pelo leitor.

Após a análise de vários Estudos de caso, Dubé and Paré (2003) apontam três áreas como sendo fundamentais para a elaboração de um Estudo de caso, nomeadamente:

- Área 1: *Design Issues*;
- Área 2: *Data Collection*;
- Área 3: *Data Analysis*.

Para cada uma das áreas, os autores definiram uma lista de recomendações a serem seguidas com o objetivo de incrementar a qualidade dos Estudos de caso.

Na primeira área, *Design Issues*, os investigadores devem definir questões ou objetivos de investigação claros, especificar as razões para a seleção de um caso único ou de um vários casos, aproveitar os casos piloto para aprimorar o design e os planos de recolha de dados, realizar Estudos de caso mais longitudinais para tirar partido dos vários métodos de recolha de dados ao estudar os fenómenos ao mesmo tempo que estes se desenrolam, e considerar teorias opostas ou alternativas para aumentar a validade e o poder preditivo dos Estudo de caso.

Na área 2, *Data Collection*, devem ser fornecidas informações detalhadas sobre os métodos de recolha de dados (entrevistas, questionários, observação direta, etc.) e os procedimentos (estratégias para amostras, número de entrevistas e entrevistados, guia das entrevistas, etc.), utilizadas tabelas para resumir as informações sobre o processo de recolha de dados, e usar a triangulação de dados para aumentar a validade interna das descobertas e fornecer explicações objetivas sobre como o processo de triangulação é alcançado.

Na última área, *Data Analysis*, é essencial fornecer descrições claras sobre os métodos e procedimentos analíticos (principalmente o modo dominante de análise) e fornecer aos *stakeholders* externos informações cruciais o suficiente para que possam acompanhar a criação das evidências das questões de investigação iniciais até às conclusões e vice-versa, fazer maior uso de técnicas e ferramentas de análise de dados preliminares, incluindo notas de campo, codificação, demonstrações visuais de dados para criar reflexões sobre os dados, apresentar citações suficientes para que os *stakeholders* externos possam fazer uma apreciação independente da análise, e comparar os resultados obtidos com a literatura para incrementar a confiança nos resultados.

Tendo em conta estes fatores, é possível concluir que o método Estudo de caso, é o método mais indicado para a realização deste estudo pois: (1) o corpo de conhecimento sobre os trabalhos de natureza científica focados em perceber, por exemplo, a produtividade possibilitada pelas tecnologias *low-code*, são ainda escassos, (2) o fenómeno não pode ser estudado fora do contexto onde ocorre, neste caso, um projeto desenvolvido com recurso à tecnologia *low-code*, (3) pretende-se efetuar um estudo aprofundado dos elementos que constituem um projeto de *low-code*, incluindo as suas características, limitações, benefícios, etc.

4.2. Descrição e seleção do caso

A *Quidgest* (2019d) é uma organização tecnológica, fundada em 1988, pioneira na geração automática de software. Esta organização possui diversas certificações: (1) o sistema de gestão da qualidade com a norma ISO 9001, (2) o sistema de gestão ambiental com a norma ISO 27001, (3) gestão da segurança da informação com a norma ISO 27001.

A organização possui uma plataforma de desenvolvimento *extreme low-code*, *Genio*, que utiliza para a construção das suas soluções destinadas a várias plataformas, Web, dispositivos móveis e *cloud*.

Segundo a *Quidgest* (2019d), esta forma de desenvolvimento de software é mais ágil e competitiva e contribui para a sua expansão em diversos mercados por todo o mundo, desde Portugal, Alemanha, Timor-Leste, Reino Unido, Brasil, etc.

É possível encontrar diversas soluções no seu portefólio de SI, nas áreas da: (1) gestão publica; (2) gestão de topo e gestão estratégica; (3) proteção de dados e cibersegurança; (4) gestão documental, BPM mobilidade e cidadania, etc.

Além do desenvolvimento de software, a empresa presta também serviços de consultoria de negócio e definição de procedimento, formação, e manutenção técnica das soluções criadas.

A plataforma *Genio*, tal como mencionado, é uma plataforma de desenvolvimento de soluções de software *extreme low-code*. Esta plataforma está assente em padrões e permite o desenvolvimento de soluções software através da construção de modelos.

Segundo a *Quidgest* (2019a), esta plataforma distingue-se de outras pois: (1) combina a independência tecnológica, a padronização de código e a velocidade de entrega em várias linguagens e tecnologias; (2) as soluções criadas evoluem com o tempo; (3) o código gerado é standard e independente da *framework* de geração e (4) a criação e manutenção dos sistemas é independente dos domínios de programação devido à interface gráfica de alto nível de abstração.

A *Quidgest* destaca várias vantagens competitivas da plataforma *Genio*, como o desenvolvimento rápido de soluções, a independência de fornecedores de tecnologias, a atualização das soluções, a maior produtividade, a redução de erros das aplicações, etc.

O Genio suporta as mais recentes arquiteturas e tecnologias Web (ASP.Net MVC, HTML5, C#), bases de dados (SQL Server, Oracle, MySQL) e WCF WebServices.

A seleção do projeto para o estudo de caso deve-se ao facto de a solução criada pela *Quidgest* ter sido desenvolvida em colaboração com uma empresa do setor privado, a QuidCLIENT (designação anonimizada por razões de confidencialidade). A escolha do projeto a analisar recaiu sobre a solução construída para a QuidCLIENT devido às suas características: projeto de grande dimensão; equipa reduzida; utilização da tecnologia *low-code*.

O Grupo QuidCLIENT surgiu na década de 1980 atualmente encontra-se presente em 16 países, sendo que a sua sede está assente na Suíça. Este grupo possui mais de 250 laboratórios de análises, 150 centros de radiologia, 12.600 colaboradores, etc.

Em Portugal, a QuidCLIENT iniciou o seu percurso com a aquisição de 85% de outra empresa e, a partir desse momento, adotou uma estratégia de crescimento e expansão nacional com a aquisição e o estabelecimento de parcerias com outros laboratórios.

A QuidCLIENT possui várias áreas de atuação, nomeadamente, a cardiologia, as análises clínicas, a radiologia, a anatomia patológica, a medicina nuclear, a gastrenterologia e a genética médica.

A gestão dos vários laboratórios é feita de forma centralizada e o objetivo do projeto foi implementar um sistema de gestão documental capaz de fazer a gestão de todo o processo de compras da QuidCLIENT, ou seja, um sistema onde são feitas as requisições de compra, que seguem os *workflows* que são previamente definidos. A solução desenvolvida foi integrada com o ERP da QuidCLIENT, para enviar as requisições. Na solução atual as compras são enviadas aos fornecedores, é recebida a informação de todas as compras enviadas aos fornecedores, são registadas no sistema, e posteriormente são recebidas as faturas.

O projeto arrancou em 2019, com uma primeira fase destinada ao levantamento dos processos de negócio. Na sua totalidade, o projeto envolveu 25000 horas de trabalho dedicadas não apenas à construção da solução, mas também à criação de novos *features* do produto, à especificação e ao levantamento dos processos de negócio, à formação e ao apoio à implementação da solução.

Além da *Quidgest* e da QuidCLIENT, é possível identificar outros dois *stakeholders* adicionais de elevada importância para o projeto, a QuidCasoCOMP1 (designação anonimizada) e a QuidCasoCOMP2 (designação anonimizada). A QuidCasoCOMP1 é a empresa responsável pelo ERP da QuidCLIENT e a QuidCasoCOMP2 é responsável pela digitalização de todos os documentos da QuidCLIENT.

4.3. Recolha de dados

As evidências foram recolhidas, no ano de 2021, através de entrevistas realizadas com a gestora de projeto e o diretor comercial da *Quidgest*, em formato não presencial com uma duração aproximada de uma hora e trinta minutos cada. Os resultados da análise foram discutidos em várias conversas subsequentes.

A gestora de projeto da *Quidgest* foi a principal visada tendo em conta que é quem detém mais informação e conhecimento sobre todo o projeto.

A entrevista foi estruturada em três grandes partes: (1) opiniões e perceções pessoais dos entrevistados sobre o *low-code* e em particular a ferramenta *Genio*; (2) opiniões e perceções pessoais dos entrevistados sobre o sucesso dos projetos no geral envolvendo *low-code* e (3) opiniões, perceções e informações relevantes sobre o projeto desenvolvido com a QuidCLIENT.

No final da entrevista foram ainda efetuadas um conjunto de questões de resposta rápida.

A entrevista foi gravada e transcrita na sua totalidade.

4.4. Análise de dados

A análise dos dados seguiu várias etapas. Num primeiro momento, foi efetuada uma leitura integral das transcrições das entrevistas. Após esta primeira leitura, foi feita uma segunda leitura com o objetivo de identificar expressões que permitissem estabelecer a ligação entre as opiniões e as perceções do entrevistado e as diferentes dimensões (e os seus constituintes) do *framework*.

Na primeira parte das entrevistas, relativa à opinião e às perceções sobre *low-code* e em particular o *Genio*, foi possível estabelecer conexão com a dimensão *developer*, a dimensão tecnologia e a dimensão projeto.

Na segunda parte das entrevistas, foram identificados alguns conceitos que não estão contemplados no *framework*, tais como, os critérios para a avaliação do desempenho e do resultado final do projeto, a avaliação do sucesso, a gestão do sucesso, a gestão dos *stakeholders* e a comunicação do sucesso. Apesar disso, foi ainda possível estabelecer ligação com a dimensão projeto.

Por fim, na última parte das entrevistas foram igualmente focados aspetos que não estão presentes no *framework* como a gestão do risco, a gestão de *stakeholders*, a gestão de projetos, os critérios para a avaliação do desempenho e do resultado final do projeto. Além dos conceitos anteriormente identificados, também foi possível estabelecer a ligação com várias dimensões do *framework*, mais especificamente com a dimensão tecnologia, a dimensão projeto e a dimensão *developer*

5. RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os principais resultados e feita a sua discussão com base no *framework* desenvolvido, a partir da literatura analisada. A apresentação e a discussão dos resultados foram organizadas em três partes, com base nas estruturas das entrevistas realizadas. Na primeira parte o foco é a tecnologia *low-code* e a ferramenta *Genio*, na segunda parte o sucesso dos projetos no geral envolvendo *low-code*, e na última parte o projeto desenvolvido com a QuidCLIENT.

5.1. *Low-code* e *Genio*

O perfil profissional do gestor de projeto e também de um dos *developers* da equipa, enquadra-se num tipo de profissionais presentes na dimensão *developers*, cuja sua área de base de formação de base não contempla as áreas das TI e dos SI (Al Alamin et al., 2021; Almonte et al., 2020; Alsaadi et al., 2021; Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança et al., 2021; Butler, 2020; Cabot, 2020; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Ruscio et al., 2021; Di Sipio et al., 2020; Dushnitsky & Stroube, 2021; ElBatanony & Succi, 2021a, 2021b; Henriques et al., 2018; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa & Henriques, 2018; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019; Woo, 2020). Neste caso, a área de formação de base do gestor de projeto é "uma licenciatura em História, com uma especialização em Ciências Documentais".

No início da sua carreira, para reconversão e requalificação do gestor do projeto foi feita "uma formação disponibilizada pela *Quidgest*, com uma duração de duas semanas, onde participaram vários tipos de profissionais, desde profissionais ligados às áreas de TI e SI até profissionais com um carácter e um perfil mais funcional", o que suporta duas vantagens identificadas na dimensão tecnologia do *framework*: uma curva de aprendizagem mais rápida (da Cruz et al., 2021); e não é preciso formação de base em TI para assumir a tarefa de desenvolvimento das soluções (Almonte et al., 2020; Alsaadi et al., 2021; Bhattacharyya &

Kumar, 2021; Bock & Frank, 2021a; Bragança et al., 2021; da Cruz et al., 2021; Di Ruscio et al., 2021; Di Sipio et al., 2020; ElBatanony & Succi, 2021a, 2021b; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Woo, 2020).

No final da referida formação, "ambos os tipos de profissionais ficaram equilibrados em termos de competências técnicas, no que diz respeito ao desenvolvimento de soluções com a plataforma *low-code* utilizada".

Outro aspeto destacado que reflete uma das desvantagens da tecnologia *low-code* presente no *framework*, é "a necessidade de codificação em aspetos que não são diretamente suportados pelos padrões disponíveis na plataforma" (Cabot, 2020; Luo et al., 2021).

Em termos de produtividade, foi referido que, em comparação com outras tecnologias e outras ferramentas, a capacidade da tecnologia *low-code* é substancialmente maior, sendo a produtividade da tecnologia *low-code* um elemento presente na dimensão tecnologia do *framework* (Bock & Frank, 2021a, 2021b; Lourenço et al., 2021; Silva et al., 2021).

São várias vantagens referidas associadas ao modelo de desenvolvimento *low-code*, e que estão presentes na dimensão tecnologia do *framework*, sendo elas: "a rapidez de desenvolvimento" (Al Alamin et al., 2021; Alsaadi et al., 2021; Arora et al., 2020; Butler, 2020; Cabot, 2020; Di Ruscio et al., 2022; Di Sipio et al., 2020; ElBatanony & Succi, 2021a; Horváth et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Luo et al., 2021; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Sanchis et al., 2020; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), "a simplificação da construção de software complexo" (da Cruz et al., 2021) (Oteyo et al., 2021), uma curva de aprendizagem mais rápida (da Cruz et al., 2021), "qualquer pessoa pode assumir a tarefa do desenvolvimento das soluções" (Almonte et al., 2020; Alsaadi et al., 2021; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a; Bragança et al., 2021; da Cruz et al., 2021; Di Ruscio et al., 2021; Di Sipio et al., 2020; ElBatanony & Succi, 2021a, 2021b; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Woo, 2020), "a

construção mais rápida de protótipos" (da Cruz et al., 2021), "a minimização do esforço de codificação manual" (Al Alamin et al., 2021; Alsaadi et al., 2021; Arora et al., 2020; Bock & Frank, 2021a; Butler, 2020; Cabot, 2020; Di Ruscio et al., 2022; Di Ruscio et al., 2021; Dushnitsky & Stroube, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Silva et al., 2021; Varajão, 2021; Waszkowski, 2019), "a minimização do esforço na instalação, configuração, formação e implementação das soluções" (Di Ruscio et al., 2022; Luo et al., 2021; Waszkowski, 2019), "a facilidade de uso" (Bragança et al., 2021; Luo et al., 2021) e "o aumento da velocidade de lançamento das soluções no mercado" (Bhattacharyya & Kumar, 2021; Butler, 2020; da Cruz et al., 2021; ElBatanony & Succi, 2021a; Horváth et al., 2020; Pichidtienthum et al., 2021).

Foi ainda salientada a "flexibilidade e a agilidade da plataforma e da tecnologia *low-code*", vantagens elencadas na dimensão tecnologia do *framework*, na "produção de soluções", mas principalmente, na "produção de aplicações de sistemas de informação estruturados, com uma base de dados de suporte" (Al Alamin et al., 2021; Luo et al., 2021).

No que toca à qualidade de soluções produzidas foi referido que "a qualidade das soluções é maior se a solução for assente na sua grande maioria nos padrões disponibilizados pela plataforma. Um padrão é algo robusto, testado, partilhado e uniforme, que vai sendo melhorado ao longo do tempo para prever novos casos" e isto traz consigo algumas vantagens abordadas na dimensão tecnologia do *framework*, tais como: a redução de erros das aplicações (Al Alamin et al., 2021) e a reutilização (Bock & Frank, 2021a; Lourenço et al., 2021) (Bock & Frank, 2021b; Bragança et al., 2021).

A metodologias de desenvolvimento são outro elemento elencado. A tecnologia *low-code* e a plataforma utlizada adaptam-se facilmente a uma perspetiva mais tradicional *waterfall*. No entanto, o mesmo reconhece que os ganhos são ainda maiores no contexto de metodologias ágeis pois a tecnologia *low-code* permite "um maior envolvimento dos perfis de negócio" (Sanchis et al., 2020; Strømsted et al., 2018), "a incorporação facilitada do feedback dos clientes"(Al Alamin et al., 2021; da Cruz et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019) através da "construção mais rápida e fácil de protótipos" (da Cruz et al., 2021), e como consequência "a minimização de requisitos instáveis e inconsistentes" (Sanchis et al., 2020; Varajão, 2021), sendo essas vantagens listadas na dimensão tecnologia do *framework*.

O entrevistado afirma que, nos projetos em que está envolvido, a primeira iteração de desenvolvimento tem a duração de duas semanas e, no final dessa iteração é já apresentado ao cliente um primeiro protótipo funcional. A partir deste momento, todos os ajustes que tenham de ser feitos são realizados sobre esse protótipo, o que permite, tal como mencionado anteriormente "um maior envolvimento dos perfis de negócio" (Sanchis et al., 2020; Strømsted et al., 2018), "a incorporação de feedback dos clientes" (Al Alamin et al., 2021; da Cruz et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), "a minimização de requisitos inconsistentes" (Sanchis et al., 2020; Varajão, 2021). Além disso, reflete também "a rapidez de desenvolvimento" (Al Alamin et al., 2021; Alsaadi et al., 2021; Arora et al., 2020; Butler, 2020; Cabot, 2020; Di Ruscio et al., 2022; Di Sipio et al., 2020; ElBatanony & Succi, 2021a; Horváth et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Luo et al., 2021; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Sanchis et al., 2020; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), "a flexibilidade e a agilidade" (Al Alamin et al., 2021; Luo et al., 2021) e "o foco na solução e não nos aspetos técnicos" (Bragança et al., 2021; Henriques et al., 2018; Kukesh & Brandenburg, 2018; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahay, Indamutsa, et al., 2020; Strømsted et al., 2018; Waszkowski, 2019). Todas estas vantagens estão presentes na dimensão tecnologia do *framework*.

No que toca às equipas de desenvolvimento, é "frequente vermos que a constituição destas equipas comtempla profissionais mais técnicos, ligados às áreas das TI e dos SI" (Al Alamin et al., 2021; Almonte et al., 2020; Alsaadi et al., 2021; Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança et al., 2021; Cabot, 2020; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Sipio et al., 2020; Henriques et al., 2018; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa & Henriques, 2018; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Silva et al., 2021; Varajão, 2021; Waszkowski, 2019; Woo, 2020), e "também profissionais mais funcionais, cuja sua área de formação não está diretamente ligada às áreas das TI e dos SI" (Al Alamin et al., 2021; Almonte et al., 2020; Alsaadi et al., 2021; Arora et al., 2020; Bhattacharyya & Kumar, 2021; Bock & Frank, 2021a, 2021b; Bragança et al., 2021; Butler, 2020; Cabot, 2020; da Cruz et al., 2021; Di Ruscio et al., 2022; Di Ruscio et al., 2021; Di Sipio et al., 2020;

Dushnitsky & Stroube, 2021; ElBatanony & Succi, 2021a, 2021b; Henriques et al., 2018; Horváth et al., 2020; Hurlburt, 2021; Ihirwe et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Lourenço et al., 2021; Luo et al., 2021; Metrôlho et al., 2020; Oteyo et al., 2021; Overeem & Jansen, 2021; Philippe et al., 2020; Pichidtienthum et al., 2021; Ragusa & Henriques, 2018; Sahay, Di Ruscio, et al., 2020; Sahay, Indamutsa, et al., 2020; Sahinaslan et al., 2021; Sanchis et al., 2020; Silva et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019; Woo, 2020). Estes dois tipos de profissionais estão destacados na dimensão *developers* do *framework*.

Os especialistas de negócio ou os profissionais funcionais podem "envolver-se e são capazes de participar nas fases de desenvolvimento de aplicações recorrendo a LCP, mais especificamente, nas fases de modelação de dados" (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020) e "definição da interface do utilizador" (Di Ruscio et al., 2022; Sahay, Indamutsa, et al., 2020), duas fases identificadas na dimensão projetos do *framework*. O seu conhecimento sobre o negócio traz consigo diversos benefícios, sendo eles: "uma melhor interação com o cliente, e por consequência um melhor entendimento das suas necessidades" (Al Alamin et al., 2021; da Cruz et al., 2021; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), realçando mais uma vez que o foco está na solução e na satisfação do cliente e não nos aspetos técnicos (Sahay, Indamutsa, et al., 2020; Waszkowski, 2019) (Bragança et al., 2021; Henriques et al., 2018; Kukesh & Brandenburg, 2018; Oteyo et al., 2021; Pichidtienthum et al., 2021; Strømsted et al., 2018).

Um bom exemplo é "um projeto desenvolvido pela *Quidgest* num país da América do Sul, com a duração de seis meses, em que os perfis funcionais foram capazes de realizar o levantamento dos requisitos e a modelação de dados sem a intervenção direta de um perfil mais técnico. Estes seis meses de duração incluíram também a formação e a implementação da solução".

Mais uma vez é possível salientar a rapidez de desenvolvimento associada ao *low-code* (Al Alamin et al., 2021; Alsaadi et al., 2021; Arora et al., 2020; Butler, 2020; Cabot, 2020; Di Ruscio et al., 2022; Di Sipio et al., 2020; ElBatanony & Succi, 2021a; Horváth et al., 2020; Khorram et al., 2020; Kukesh & Brandenburg, 2018; Luo et al., 2021; Oteyo et al., 2021; Pichidtienthum et al., 2021; Sahinaslan et al., 2021; Sanchis et al., 2020; Strømsted et al., 2018; Varajão, 2021; Waszkowski, 2019), a minimização do esforço de instalação, configuração, formação e

implementação da solução (Di Ruscio et al., 2022; Luo et al., 2021; Waszkowski, 2019), bem como o aumento da velocidade de lançamento das soluções no mercado (Bhattacharyya & Kumar, 2021; Butler, 2020; da Cruz et al., 2021; ElBatanony & Succi, 2021a; Horváth et al., 2020; Pichidtienthum et al., 2021).

É possível concluir que ambos os papéis, o do profissional mais diretamente ligado às áreas dos SI e das TI, e o do profissional mais funcional, cuja área de base de formação não comtempla as áreas dos SI e das TI, são igualmente essenciais para o processo de desenvolvimento de soluções com tecnologia low-code.

A constituição típica das equipas de projeto que desenvolvem soluções recorrendo à tecnologia *low-code* varia de projeto para projeto. No caso da empresa em estudo, há projetos em que o peso dos programadores é de cerca de 10% a 20% e o resto da equipa é constituída por elementos mais funcionais. No entanto, em projetos que não totalmente assentes nos padrões da plataforma, onde é necessário um maior esforço de desenvolvimento, uma maior customização, uma maior personalização, o peso dos programadores pode assumir um valor entre os 40% a 50% da equipa.

No que toca ao tamanho das equipas de projeto, "habitualmente as equipas que desenvolvem soluções recorrendo à tecnologia *low-code* são equipas consideravelmente pequenas. Um projeto desenvolvido entre a *Quidgest* e um dos seus clientes contou, apenas com uma pessoa a tempo inteiro e as restantes em tempo parcial". Este aspeto corrobora a vantagem presente na dimensão tecnologia do *framework*, onde é mencionado que a tecnologia *low-code* proporciona equipas de trabalho com um tamanho mais reduzido (Alsaadi et al., 2021; ElBatanony & Succi, 2021a).

Em termos de competências são destacadas, "a criatividade e o raciocínio lógico". Além disso, "os profissionais com carácter mais técnico, os programadores e profissionais de software, devem ter a capacidade de se enquadrar numa lógica de equipa, percebendo as mais-valias e as potencialidades de cada um dos perfis que têm dentro da equipa e ter a capacidade de aproveitar ao máximo as potencialidades da plataforma".

Um dos prolemas enfrentados na contratação dos perfis técnicos, ou seja, profissionais cuja área de formação de base contempla as áreas dos SI e das TI, é "a pobre e fraca perceção cultural e a falta de consciencialização que estes apresentam perante a tecnologia *low-code*", uma das barreiras enquadradas na dimensão tecnologia do *framework* (Bhattacharyya &

Kumar, 2021). “Alguns deles sentem que o seu papel dentro das organizações vai ser menorizado, inferiorizado, desvalorizado por não programar aplicações, o que não é de todo a realidade” (ElBatanony & Succi, 2021a; Hurlburt, 2021; Woo, 2020). Este papel do profissional de TI dentro das organizações é outro elemento destacado no *framework*, na dimensão *developers*.

No caso em que “as organizações compram a ferramenta *low-code*, os profissionais técnicos são aqueles que estão em melhores condições para inicialmente assumir as funções de responsável pelo desenvolvimento de soluções e de projetos”.

No contexto da empresa em estudo, “estes profissionais são igualmente essenciais para o aprimoramento e para a construção de padrões disponíveis na ferramenta”. Isto reforça a ideia de que o papel do profissional de TI nas organizações não é inferiorizado, desvalorizado com o *low-code*, pelo contrário, pode ser reforçado graças às melhorias, por exemplo, de produtividade” (ElBatanony & Succi, 2021a; Hurlburt, 2021; Woo, 2020).

Além desta barreira identificada foi ainda referida “a falta de estratégias de adoção adequadas” (Bhattacharyya & Kumar, 2021). “A maioria das organizações considera que aquilo que o *low-code* pode proporcionar é algo que transcende a realidade. Muitos consideram que só as empresas que são especializadas em *low-code* é que têm capacidade de desenvolver as soluções e que “os comuns mortais” não têm as competências necessárias para o fazer”.

Contudo, “quando confrontados com provas de conceito, com projetos desenvolvidos com recurso a tecnologia *low-code*, a sua opinião muda e reconhecem a grande flexibilidade, agilidade e capacidade desta tecnologia para a construção das soluções” conforme referido por Al Alamin et al. (2021) e Luo et al. (2021).

5.2. Sucesso dos projetos

A definição subjacente ao sucesso dos projetos desenvolvidos com tecnologia *low-code*, “é uma questão complexa e que nos transporta para vários pontos de vista”.

“Segundo a perspetiva do meu superior, o alcance do sucesso traduz-se num simples critério para a avaliação do desempenho e do resultado final do projeto: o recebimento do pagamento do cliente. Apesar deste critério ser um bom indicador, a minha definição de sucesso é mais abrangente e completa. Existem outros critérios que apresentam igual ou

maior relevância, como é o caso da boa implementação da solução, da capacidade de resposta aos requisitos do cliente, e a utilização da solução produzida".

Além destes critérios, foram apontados outros e que vulgarmente servem de base à avaliação do sucesso, mais concretamente, "o cumprimento dos prazos estabelecidos", "o cumprimento do orçamento e "uma boa margem de lucro". Contudo, "estes critérios são bastante redutores, podendo não transpor diretamente e da forma mais completa aquilo que é o sucesso do projeto".

O critério de "utilização da solução produzida é aquele que de facto merece uma maior importância". No caso de muitas das soluções que são criadas, a sua utilização não reflete a capacidade total dos sistemas, sendo apenas utilizada, em alguns casos, uma pequena parte das funcionalidades. Isto acontece porque, por vezes, ocorre uma má implementação, uma má instalação, ou uma má formação, e o cliente acaba por não conseguir usufruir do sistema como deveria. Como consequência, em alguns casos, os sistemas são substituídos por outros, que acabam por oferecer a mesma capacidade de resposta e de opções".

Na empresa em estudo, "os critérios para a avaliação do desempenho e do resultado final do projeto são ajustados e adequados a cada projeto. Muitas vezes, a questão da boa margem de lucro e do orçamento é e pode ser ultrapassada se de facto o projeto for considerado um projeto estratégico para a organização. Noutros casos, logo à partida o prazo estabelecido para o projeto pode não ser realista, e o não cumprimento do prazo pode significar um fracasso sob ponto de vista do cliente. No entanto é fundamental fazer uma boa gestão do projeto, abordar o cliente, perceber o que é ou não possível fazer, reajustar prazos, reajustar recursos, priorizar requisitos, funcionalidade, reduzir âmbitos, etc.".

Esta última parte reflete bem um aspeto do sucesso dos projetos, não elencado na literatura analisada, que é a gestão das expectativas do cliente.

Outro aspeto relevante é a avaliação do sucesso, que "deve ser efetuada de forma regular, ao longo de todo o projeto, uma vez que a perspetiva do sucesso pode sofrer alterações ao longo do tempo". Aqueles que são os critérios para a avaliação do desempenho e do resultado final do projeto "podem assumir diferentes prioridades em diferentes momentos do projeto. Inicialmente, por exemplo, o cumprimento do prazo pode-se assumir como o critério mais importante, no entanto ao longo do projeto essa perspetiva pode mudar para a qualidade da solução, entre outros".

Os participantes da avaliação do sucesso devem "abarcar diferentes *stakeholders*. Desde o gestor do projeto ao cliente, envolvendo também quer a equipa de projeto, quer a equipa do lado do cliente. Em termos de relevância, por vezes a equipa do cliente encontra-se num nível superior ao do gestor do projeto do cliente, tendo em conta que este pode ter vários projetos a seu cargo, pode ser uma pessoa menos presente no projeto, pode apenas assumir a função/o cargo no sentido burocrático, e não ter uma perceção real do projeto."

Tal como já mencionado anteriormente, a avaliação do sucesso é "algo que deve ocorrer de forma regular ao longo do projeto e ser reportado aos diversos *stakeholders*, para que todos possam ter uma perceção da realidade do projeto e para que os diversos pontos possam ser discutidos, analisados. No entanto, na maioria das vezes, isto sucede de uma forma mais informal".

A avaliação formal do sucesso dos projetos "acarreta consigo custos, sendo necessário manter um equilíbrio e entender qual a mais-valia que vamos retirar. No caso da empresa em estudo, "os procedimentos são simplificados e enquadrados de forma a recolher e a tratar a informação necessária sem que esta requeira muito trabalho. Cada vez mais a lógica da gestão de sucesso dos projetos e os seus procedimentos estão a ser enquadrados no trabalho da empresa e as tarefas são integradas para que a informação seja recolhida e tratada."

5.3. Projeto *QuidCLIENT*

A seleção do projeto para estudo deve-se ao facto de o projeto ter sido desenvolvido com uma empresa com uma dimensão considerável, e requerer uma nova área de desenvolvimento, exigindo mais da empresa e da tecnologia.

No projeto, "apesar de existir inicialmente a compreensão de algum risco associado ao projeto, a perceção do risco real só surgiu mais tarde. Os riscos identificados prendem-se "com o facto do cliente do projeto, ser uma organização de grande dimensão, com uma estrutura pesada e burocrática, o que dificultou a condução do projeto".

Em Portugal, o cliente adotou uma estratégia de crescimento e expansão nacional com a aquisição e o estabelecimento de parcerias com outras empresas. Possui várias áreas de atuação, nomeadamente, a cardiologia, as análises clínicas, a radiologia, a anatomia patológica, a medicina nuclear, a gastrenterologia e a genética médica. A gestão das várias

empresas do grupo é feita de forma centralizada, onde é realizada toda a gestão financeira e a organização das diferentes empresas pertencentes ao grupo.

O objetivo do projeto foi implementar um sistema de gestão documental para fazer a gestão de todo o processo de compras, ou seja, um sistema onde são feitas as requisições de compra, que seguem *workflows* que são previamente definidos. A solução incluiu a integração com o sistema financeiro. Em termos de *workflows* que são suportados pela solução, as compras são enviadas aos fornecedores, é recebida a informação dessas cópias todas de compras, são todas registadas no sistema e depois são recebidas as faturas.

Foram implementados vários mecanismos em que é feita a aprovação automática das faturas, ou seja, são minimizados ao máximo os casos em que um ser humano tem de aprovar as faturas, com base numa série de mecanismos automáticos e de integração com a aplicação ERP.

A quantidade de faturas e de outros documentos de despesa era um problema para a organização, devido ao elevado volume de informação, que tinha de ser aprovada pelo mesmo conjunto de pessoas, o que causava problemas em termos de tempo, de alocação de recursos, sendo necessário investir em equipas "gigantescas" para que a resposta fosse dada em tempo útil.

O projeto arrancou em outubro de 2019 e, na primeira fase do projeto, o foco esteve voltado para o levantamento dos processos de negócio. A pandemia trouxe consigo algumas consequências como o atraso o desenvolvimento do projeto, uma vez que a empresa foi um dos laboratórios que mais contribuiu para a criação de centros de vacinação. No ano 2021, o projeto tinha a fase de desenvolvimento concluída e já tinha sido elaborado um plano de "*go out"* para a implementação da solução em todas as empresas do grupo.

O projeto envolveu um pacote de 2500 horas de trabalho, não dedicadas única e exclusivamente ao projeto, mas também à criação de novas *features* do produto. Estas 2500 horas envolveram a especificação e o levantamento dos processos de negócio, a criação da solução, a formação e o apoio à implementação da solução.

Em termos de complexidade, foi um projeto de "complexidade média, tendo em conta que a implementação dos mecanismos automáticos de aprovação foi algo novo, com o qual a empresa nunca tinha trabalhado, o que obrigou ao estudo, à aprendizagem e à testagem". De notar que a solução desenvolvida foi assente num produto que já existia, permitindo a

reutilização de componentes. O desenvolvimento também assumiu uma perspetiva de generalização da solução, para que posteriormente pudesse ser aplicada e reutilizada noutros *workflows*, sendo que este aspeto contribuiu para o incremento do grau de complexidade.

No que diz respeito aos vários *stakeholders* do projeto é possível identificar: a *Quidgest*, a *QuidCLIENT*, e mais dois fornecedores, a QuidCasoComp1 e a QuidCasoCOMP2. O envolvimento dos fornecedores no projeto foi importante, pois estes tiveram um peso substancial no seu desenvolvimento.

A equipa da *Quidgest* foi constituída por dois profissionais a tempo inteiro, um com carácter mais técnico, um programador, e outro com carácter mais funcional. Além destes elementos *core*, existiu ainda um terceiro elemento que deu suporte ao elemento mais funcional da equipa. Isto reflete uma das vantagens presente na dimensão tecnologia do *framework*, que é o tamanho mais reduzido das equipas (Alsaadi et al., 2021; ElBatanony & Succi, 2021a).

Do lado do cliente, a equipa foi constituída pelos vários representantes dos departamentos envolvidos nos *workflows*. Inicialmente a equipa possuía também dois gestores de projetos (um gestor mais técnico e outro gestor mais funcional).

A opção tomada pela utilização da tecnologia *low-code* deve-se ao ajuste entre as características da plataforma utilizada e o tipo de solução a produzir.

No que diz respeito à metodologia de desenvolvimento associada ao projeto, inicialmente o que foi delineado foi a utilização de uma metodologia mais *waterfall*, tendo em consideração a quantidade reduzida de requisitos. No entanto, após o levantamento dos processos de negócio, o número de requisitos identificados cresceu exponencialmente e justificou-se a mudança para uma metodologia mais próxima da *Scrum*, onde os requisitos são identificados, priorizados, desenvolvidos, implementados, e a solução entregue incrementalmente. Existiu uma transição do *deploy* tradicional de um produto para uma lógica de iterações, em que vai sendo feito o *deploy* de uma nova versão. A metodologia de desenvolvimento seguida no projeto vai de encontro às metodologias de desenvolvimento identificadas na dimensão projetos do *framework*: *Agile* (Al Alamin et al., 2021; Kukesh & Brandenburg, 2018; Sahay, Indamutsa, et al., 2020; Varajão, 2021) e *Scrum* (Sahay, Indamutsa, et al., 2020).

No que toca à metodologia de gestão do projeto, a metodologia usada é uma metodologia própria, desenvolvida pela *Quidgest*, que comtempla várias tendências. A metodologia contém elementos de guias da *IPMA*, tendo em conta que a *Quidgest* é certificada em *IPMA*, onde o foco é mais voltado para a gestão da equipa e da relação do cliente e menos para a parte técnica. Além deste referencial, também contém elementos do *PMBOK*, da *ISO9001*, do *CMMI*, etc.

É possível realçar um conjunto de aspetos positivos associados ao projeto, tal como, "o bom relacionamento com o cliente, o bom conhecimento sobre o negócio e os seus processos e a qualidade da solução gerada". Por outro lado, é possível identificar também alguns aspetos mais negativos, como é o caso, da "entrega de versões da solução com alguns erros, de alguns requisitos mais instáveis e inconsistentes".

O reconhecimento da capacidade e da competência da equipa de projeto foram alcançados, o que contribuiu para que novos projetos subsequentes estejam já a ser equacionados, refletindo o sucesso do projeto e o alcance da satisfação do cliente.

Outro elemento que contribuiu para o sucesso, foi o facto "do produto ter crescido substancialmente no sentido do mercado empresarial. Foi realizado um esforço para adicionar novos *features* ao produto, e é possível considerar que este foi sem dúvida um projeto estratégico".

Tendo em consideração a perspetiva do cliente, "os melhores critérios para a avaliação do desempenho e do resultado final do projeto são: a minimização das aprovações manuais e por consequência a diminuição dos prazos dos fluxos de tramitação. Apesar disso, existiam ainda um conjunto de aspetos extra que tinham de ser alinhados para que para que estes critérios se verificassem, como foi o caso de todas as faturas terem de ter uma nota de encomenda. Todo o processo de requisição, nota de encomenda, e fatura, foi refinado o para retirar o maior proveito possível.

Tendo em conta os fornecedores, a QuidCasoCOMP2 pode não ter encarado este projeto como um sucesso, uma vez que a empresa teve dificuldades no cumprimento dos prazos, dos requisitos e dos objetivos que lhe foram propostos. "Possivelmente foi feita uma má gestão das expectativas do cliente". Por outro lado, "a QuidCasoComp1 apesar de ter apresentado algumas falhas, em termos colaborativos é possível considerar que foi um sucesso. De facto,

o seu comprometimento enquanto *stakeholder* no projeto contribuiu quer para o seu sucesso, quer também para o sucesso do projeto".

Além da "boa relação com o cliente", é possível identificar outros fatores críticos de sucesso: (1) "a resposta rápida às solicitações do cliente", (2) "a comunicação regular e eficaz com o cliente", (3) "a experiência da equipa de trabalho". Estes fatores de sucesso não foram identificados na literatura de *low-code* analisada, no entanto, devem ser considerados.

Em temos de âmbito do projeto, "é possível afirmar que o âmbito foi cumprido e excedido, tendo havido com a devida articulação com o cliente, uma alteração não só do âmbito como também do tempo e do custo do projeto". Em termos do cumprimento dos prazos, "para além das diversas condicionantes associada à pandemia COVID-19, os fornecedores, em algumas situações podiam ter efetuado as suas operações de uma forma mais célere, o que prejudicou o cumprimento dos prazos estabelecidos". Do ponto de vista dos custos, tanto do lado do cliente como do lado da *Quidgest*, os mesmos foram excedidos. Da perspetiva do cliente, porque os requisitos excederam os inicialmente pensados, e da perspetiva da *Quidgest*, porque vários componentes do produto foram criados sem terem sido inicialmente equacionados, o que conduziu a uma alteração do âmbito que culminou em assumir a diminuição do lucro para que o produto se desenvolvesse. Existe ainda um contrato de manutenção ou de suporte que arrancou no dia 1 de junho de 2021.

Por fim, foi referido que "os processos de avaliação do sucesso de forma formal não são definidos nos projetos como deviam. Além disso, concorda que em alguns casos os *stakeholders* não participam na avaliação do sucesso dos projetos, como também conforme deveria acontecer". Na sua opinião, o sucesso que é reportado muitas vezes sobre os projetos resulta de "perceções idealizadas e não de avaliações realizadas formalmente".

O gestor de projeto afirma, também, que de um modo geral, os critérios de avaliação do sucesso estão limitados, na sua maioria, "ao cumprimento do âmbito, do tempo e do custo e à satisfação do cliente", no entanto considera que essa perspetiva está a sofrer alterações e a evoluir na empresa. Outro aspeto destacado é a avaliação do sucesso, que segundo a ótica do gestor do projeto, é uma preocupação que "se mantém ao longo de todo o projeto, sendo feito um acompanhamento ao longo de todo o ciclo de vida do projeto. Esta preocupação normalmente é vivida com mais fervor pelo gestor do projeto, principalmente no caso da *Quidgest*, e no caso do cliente, essa preocupação já é mais variável". Existem ainda dois

fatores que influenciam diretamente esta preocupação: (1) “a dimensão do projeto”, (2) “o quão estratégico é o projeto para a organização”.

6. CONCLUSÃO E IMPLICAÇÕES

A investigação foi motivada pela escassez de trabalhos de natureza científica focados em entender, por exemplo, a produtividade possibilitada pelas tecnologias l*ow-code*. Tendo em conta este fator, neste artigo foi feita não só uma caracterização dos estudos realizados até ao momento sobre *low-code*, como também o estudo aprofundado dos aspetos relevantes de um projeto de *low-code*, incluindo as suas características, limitações, benefícios, etc.

Em primeiro lugar, foi realizada uma revisão de literatura, uma caracterização do estado da arte, de trabalhos de natureza científica e não científica sobre a tecnologia *low-code*. Como principais resultados da revisão de literatura são de referir a síntese das publicações selecionadas, bem como a construção de um *framework* que identifica dimensões e conceitos relevantes.

Posteriormente, foi efetuado o estudo de um caso de utilização de tecnologia *low-code* a nível de um projeto. Este estudo de caso permitiu estabelecer a ligação entre as opiniões e as perceções identificadas no estudo de caso e o *framework* e as suas dimensões.

Além disso, foi possível também identificar elementos adicionais que podem impulsionar investigações futuras.

6.1. Contribuições

Em primeiro lugar, foi desenvolvido, com base na análise da literatura, um *framework* abrangente com seis dimensões. A primeira dimensão está diretamente relacionada com a tecnologia *low-code*, onde são comtemplados: (1) os tipos classificação que a tecnologia *low-code* pode ter, (2) as vantagens associadas à tecnologia *low-code*, (3) as desvantagens associadas à tecnologia *low-code*, (4) outras abordagens associadas à tecnologia *low-code*, (5) as barreiras associadas à tecnologia *low-code*, (6) a produtividade da tecnologia *low-code* e (7) as perspetivas futuras associadas à tecnologia *low-code*.

A segunda dimensão está diretamente ligada aos projetos de desenvolvimento de soluções com recursos à tecnologia *low-code*, sendo elencados: (1) os fatores de sucesso, (2) as metodologias de desenvolvimento, e (3) as fases de desenvolvimento de soluções recorrendo a LCP.

A terceira dimensão aborda os vários elementos relacionados com *developers* que utilizam a tecnologia *low-code/no-code* para o desenvolvimento das suas soluções, mais concretamente: (1) os tipos de *developers* existentes e (2) o papel do profissional de TI dentro das organizações.

A quarta dimensão apresenta os desafios associados ao desenvolvimento de soluções com recurso às LCP: (1) os problemas técnicos enfrentados no desenvolvimento das soluções, e (2) a seleção e comparação de LCP.

Por fim, a última dimensão comtempla as informações associada a cada uma das publicações selecionadas: (1) o tipo de estudo associado à publicação, (2) o *source type* e o *doument type* da publicação, e (3) o ano da publicação.

Além disso, o trabalho realizado permitiu estabelecer a ligação entre o estudo de caso e algumas dimensões do *framework* criado, tais como: a dimensão tecnologia, a dimensão projeto e a dimensão *developer*. Isto suporta a validade do *framework* e das duas dimensões.

Por outro lado, outra contribuição que surgiu, ao longo trabalho, é a identificação de novos elementos que devem ser adicionados às várias dimensões que compõem o nosso *framework*: os critérios para a avaliação do desempenho e do resultado final do projeto, a avaliação do sucesso, a gestão de projetos, a gestão do sucesso, a gestão dos *stakeholders* e a comunicação do sucesso.

Estes elementos podem ser levados em consideração em investigações futuras mais concentradas no sucesso dos projetos que envolvem a utilização de tecnologia *low-code*, ou então como complemento e expansão do *framework* desenvolvido.

6.2. Limitações

O trabalho realizado apresenta algumas limitações. Os trabalhos de natureza científica existentes sobre a tecnologia *low-code*, bem como sobre os projetos desenvolvidos com tecnologia *low-code* são ainda escassos, devido ao facto de ser um conceito relativamente recente. É necessário continuar a realizar e a aprofundar os trabalhos de natureza científica nesta área, para que o corpo de conhecimento se desenvolva.

Uma limitação identificada decorre do facto de, no estudo de caso, apenas terem sido recolhidas as opiniões e perceções da organização fornecedora. Isto impossibilitou a

confrontação de ideias entre a perspetiva do fornecedor da solução e a perspetiva do cliente. As duas perspetivas iriam complementar a visão do estudo de caso.

6.3 Investigação futura

As investigações futuras nesta área podem-se dar em três direções diferentes.

Na primeira direção, podem ser realizados estudos adicionais para reforçar o *framework* desenvolvido com base na literatura, acompanhando a evolução da investigação. Esses estudos poderão conduzir a novas dimensões e respetivos elementos.

A segunda linha de investigação futura passa por restringir o âmbito da investigação apenas ao sucesso dos projetos desenvolvidos com recurso à tecnologia *low-code*. Tendo em conta a grande quantidade de elementos adicionais identificados através da análise do estudo de caso ligados ao sucesso e à gestão do sucesso, justifica-se a realização de uma investigação nesta área.

Por fim a terceira linha de direção de investigação, destina-se ao aprofundamento do estudo de caso através da análise das opiniões e das perceções de outros *stakeholders* do como o cliente, sobre o projeto desenvolvido, o que iria permitir a confrontação de ideias e o enriquecimento do estudo de caso.

REFERÊNCIAS

Al Alamin, M. A., Malakar, S., Uddin, G., Afroz, S., Haider, T. B., & Iqbal, A. (2021). *An Empirical Study of Developer Discussions on Low-Code Software Development Challenges.* Paper presented at the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR).

Alexander, F. (2021a). Low-Code and No-Code: What's the Difference and When to Use What? Retrieved from https://www.outsystems.com/blog/posts/low-code-vs-no-code/

Alexander, F. (2021b). What Is Low-Code? Retrieved from https://www.outsystems.com/blog/posts/what-is-low-code/

Almonte, L., Cantador, I., Guerra, E., & de Lara, J. (2020). *Towards automating the construction of recommender systems for low-code development platforms.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.

Alsaadi, H. A., Radain, D. T., Alzahrani, M. M., Alshammari, W. F., Alahmadi, D., & Fakieh, B. (2021). Factors that affect the utilization of low-code development platforms: survey study. *Romanian Journal of Information Technology and Automatic Control, 31*(3), 123-140.

Arora, R., Ghosh, N., & Mondal, T. (2020). *Sagitec software studio (s3)-a low code application development platform.* Paper presented at the 2020 International Conference on Industry 4.0 Technology (I4Tech).

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The case research strategy in studies of information systems. *MIS quarterly*, 369-386.

Bhattacharyya, S. S., & Kumar, S. (2021). Study of deployment of "low code no code" applications toward improving digitization of supply chain management. *Journal of Science and Technology Policy Management*. doi:10.1108/JSTPM-06-2021-0084

Bock, A. C., & Frank, U. (2021a). *In search of the essence of low-code: an exploratory study of seven development platforms.* Paper presented at the 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).

Bock, A. C., & Frank, U. (2021b). Low-Code Platform. *Business & Information Systems Engineering, 63*(6), 733-740.

Bragança, A., Azevedo, I., Bettencourt, N., Morais, C., Teixeira, D., & Caetano, D. (2021). *Towards supporting SPL engineering in low-code platforms using a DSL approach.* Paper presented at the Proceedings of the 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences.

Butler, R. R. (2020). *Estimating Effort for Low-code Applications.* (Master's Thesis). East Carolina University,

Cabot, J. (2020). *Positioning of the low-code movement within the field of model-driven engineering.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.

da Cruz, M. A., de Paula, H. T., Caputo, B. P., Mafra, S. B., Lorenz, P., & Rodrigues, J. J. (2021). OLP—A RESTful Open Low-Code Platform. *Future Internet, 13*(10), 249. Retrieved from https://www.mdpi.com/1999-5903/13/10/249

Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 1-10. doi:10.1007/s10270-021-00970-2

Di Ruscio, D., Kolovos, D., De Lara, J., Tisi, M., & Wimmer, M. (2021). *LowCode 2021: 2 nd Workshop on Modeling in Low-Code Development Platforms.* Paper presented at the 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).

Di Sipio, C., Di Ruscio, D., & Nguyen, P. T. (2020). *Democratizing the development of recommender systems by means of low-code platforms.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.

Dubé, L., & Paré, G. (2003). Rigor in Information Systems Positivist Case Research: Current Practices, Trends, and Recommendations. *MIS quarterly, 27*(4), 597-636. doi:10.2307/30036550

Dushnitsky, G., & Stroube, B. K. (2021). Low-code entrepreneurship: Shopify and the alternative path to growth. *Journal of Business Venturing Insights, 16*, e00251. doi:10.1016/j.jbvi.2021.e00251

ElBatanony, A., & Succi, G. (2021a). *The pareto distribution of software features and no-code.* Paper presented at the Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code.

ElBatanony, A., & Succi, G. (2021b). *Towards the no-code era: a vision and plan for the future of software development.* Paper presented at the Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code.

Fryling, M. (2019). Low code app development. *Journal of Computing Sciences in Colleges, 34*(6), 119-119.

Henriques, H., Lourenço, H., Amaral, V., & Goulão, M. (2018). *Improving the Developer Experience with a Low-Code Process Modelling Language.* Paper presented at the Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Copenhagen, Denmark https://doi.org/10.1145/3239372.3239387.

Horváth, B., Horváth, Á., & Wimmer, M. (2020). *Towards the next generation of reactive model transformations on low-code platforms: three research lines.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.

Hurlburt, G. F. (2021). Low-Code, No-Code, What's Under the Hood? *IT Professional, 23*(6), 4-7.

Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., & Pierantonio, A. (2020). *Low-code engineering for internet of things: a state of research.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.

Khorram, F., Mottu, J.-M., & Sunyé, G. (2020). *Challenges & opportunities in low-code testing.* Paper presented at the Proceedings of the 23rd ACM/IEEE International

Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.
Kukesh, J., & Brandenburg, K. (2018). *Innovative low-code tool for Systems Analysis and Design.* Paper presented at the Paper presented at the MWAIS 2018 Proceedings.
Lourenço, H., Ferreira, C., & Seco, J. C. (2021). *OSTRICH-A Type-Safe Template Language for Low-Code Development.* Paper presented at the 2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS).
Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). *Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective.* Paper presented at the Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).
Metrôlho, J., Ribeiro, F., & Araújo, R. (2020). *A strategy for facing new employability trends using a low-code development platform.* Paper presented at the 14th International Technology, Education and Development Conference.
Oteyo, I. N., Pupo, A. L. S., Zaman, J., Kimani, S., De Meuter, W., & Boix, E. G. (2021). *Building Smart Agriculture Applications Using Low-Code Tools: The Case for DisCoPar.* Paper presented at the 2021 IEEE AFRICON.
Overeem, M., & Jansen, S. (2021). *Proposing a Framework for Impact Analysis for Low-Code Development Platforms.* Paper presented at the 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).
Pantelimon, S.-G., Rogojanu, T., Braileanu, A., Stanciu, V.-D., & Dobre, C. (2019). *Towards a seamless integration of iot devices with iot platforms using a low-code approach.* Paper presented at the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT).
Paré, G. (2004). Investigating information systems with positivist case research. *Communications of the association for information systems, 13*(1), 18.
Philippe, J., Coullon, H., Tisi, M., & Sunyé, G. (2020). *Towards transparent combination of model management execution strategies for low-code development platforms.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.
Pichidtienthum, S., Pugsee, P., & Cooharojananone, N. (2021). *Developing Module Generation for Odoo Using Concept of Low-Code Development Platform and Automation Systems.* Paper presented at the 2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA).
Quidgest. (2019a). Genio. Retrieved from https://quidgest.com/quidgest/plataforma-genio/#genio-4all
Quidgest. (2019b). Genio Models & Artificial Intelligence Inovação Disruptiva. Retrieved from https://quidgest.com/wp-content/uploads/2019/06/Quidnews_26_maquete-09MAI.pdf
Quidgest. (2019c). Genio vs Low-code: Function Points/Month comparison. Retrieved from https://genio.quidgest.com/wp-content/uploads/2016/10/fpa.pdf
Quidgest. (2019d). Sobre a Quidgest. Retrieved from https://www.quidgest.pt/boasvindasPT.asp?LT=PTG
Ragusa, G., & Henriques, H. (2018). *Code review tool for visual programming languages.* Paper presented at the 2018 IEEE symposium on visual languages and human-centric computing (VL/HCC).

Richardson, C., & Rymer, J. (2014). New Development Platforms Emerge For Customer-Facing Applications. *Cambridge, Forrester Research*.

Rymer, & Appian. (2017). The Forrester Wave™: Low-Code Development Platforms For AD&D Pros, Q4 2017. *Cambridge, Forrester Research*.

Rymer, & Koplowitz. (2019). The forrester wave™: Low-code development platforms for ad&d professionals, q1 2019. *Cambridge, Forrester Research*.

Sahay, A., Di Ruscio, D., & Pierantonio, A. (2020). *Understanding the role of model transformation compositions in low-code development platforms.* Paper presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). *Supporting the understanding and comparison of low-code development platforms.* Paper presented at the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).

Sahinaslan, E., Sahinaslan, O., & Sabancıoglu, M. (2021). *Low-code application platform in meeting increasing software demands quickly: SetXRM.* Paper presented at the AIP Conference Proceedings.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-Code as Enabler of Digital Transformation in Manufacturing Industry. *Applied Sciences, 10*(1), 1-17. doi:10.3390/app10010012

Silva, C., Vieira, J., Campos, J. C., Couto, R., & Ribeiro, A. N. (2021). Development and validation of a descriptive cognitive model for predicting usability issues in a low-code development platform. *Human Factors, 63*(6), 1012-1032.

Strømsted, R. I., Marquard, M., & Heuck, E. (2018). *Towards Low-Code Adaptive Case Management Solutions with Dynamic Condition Response Graphs, Subprocesses and Data.* Paper presented at the 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW).

Unqork. No-Code vs. Low-Code. Retrieved from https://finxact.com/wp-content/uploads/2020/10/ebook-no-code-vs-low-code.pdf

Varajão, J. (2021). Software development in disruptive times. *Communications of the ACM, 64*(10), 32-35.

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine, 52*(10), 376-381.

Woo, M. (2020). The Rise of No/Low Code Software Development—No Experience Needed? *Engineering, 6*(9), 960. https://doi.org/10.1016/j.eng.2020.07.007

Yin, R. K. (2003). Case study research: Design and methods. Thousand Oaks. *Sage. Young, LC and Wilkinson, IR (1989). The role of trust and co-operation in marketing channels: a preliminary study. European Journal of Marketing, 23*(2), 109-122.

APÊNDICE I - RELAÇÃO DAS PUBLICAÇÕES SELECIONADAS COM OS CONCEITOS RELEVANTES

Artigos	Conceitos									
	Definição	Vantagens e benefícios	Barreiras	Fatores de sucesso	Impactos	Desvantagens e limitações	Competências requeridas	Perspetivas futuras	Termos equivalentes ou complementares	Produtividade da tecnologia *low-code*
(Al Alamin et al., 2021)	X	X			X				X	
(Almonte et al., 2020)	X	X			X		X		X	
(Alsaadi et al., 2021)	X	X			X	X	X		X	
(Arora et al., 2020)	X	X			X		X		X	
(Bhattacharyya & Kumar, 2021)	X	X	X		X		X			
(Bock & Frank, 2021a)	X	X		X	X		X			X
(Bock & Frank, 2021b)	X	X							X	X
(Bragança et al., 2021)	X	X			X	X	X		X	X
(Butler, 2020)	X	X		X	X	X	X			
(Cabot, 2020)	X	X								
(da Cruz et al., 2021)	X	X			X	X	X			
(Di Ruscio et al., 2022)	X	X		X			X	X	X	
(Di Ruscio et al., 2021)	X	X		X			X			
(Di Sipio et al., 2020)	X	X				X	X		X	
(Dushnitsky & Stroube, 2021)	X	X			X		X	X		
(ElBatanony & Succi, 2021a)	X	X			X		X			

Artigos	Conceitos									
	Definição	Vantagens e benefícios	Barreiras	Fatores de sucesso	Impactos	Desvantagens e limitações	Competências requeridas	Perspetivas futuras	Termos equivalentes ou complementares	Produtividade da tecnologia *low-code*
(ElBatanony & Succi, 2021b)	X	X					X	X		
(Henriques et al., 2018)	X	X			X					
(Horváth et al., 2020)	X	X			X					
(Hurlburt, 2021)	X	X			X		X	X		
(Ihirwe et al., 2020)	X	X					X	X		
(Khorram et al., 2020)	X	X		X	X		X		X	
(Kukesh & Brandenburg, 2018)	X	X			X		X			
(Lourenço et al., 2021)	X	X			X		X			X
(Luo et al., 2021)	X	X			X	X	X		X	
(Metrôlho et al., 2020)	X	X					X			
(Oteyo et al., 2021)	X	X					X	X	X	
(Overeem & Jansen, 2021)	X	X			X		X	X	X	
(Philippe et al., 2020)	X	X			X				X	
(Pichidtienthum et al., 2021)	X	X					X			
(Ragusa & Henriques, 2018)									X	
(Sahay, Di Ruscio, et al., 2020)	X	X			X		X			
(Sahay, Indamutsa, et al., 2020)	X	X			X		X		X	

Artigos	Conceitos									
	Definição	Vantagens e benefícios	Barreiras	Fatores de sucesso	Impactos	Desvantagens e limitações	Competências requeridas	Perspetivas futuras	Termos equivalentes ou complementares	Produtividade da tecnologia *low-code*
(Sahinaslan et al., 2021)	X	X					X	X	X	
(Sanchis et al., 2020)	X	X				X	X	X		
(Silva et al., 2021)	X	X			X		X		X	X
(Strømsted et al., 2018)	X	X			X					
(Varajão, 2021)		X		X	X			X		
(Waszkowski, 2019)	X	X			X		X			
(Woo, 2020)		X			X	X	X	X		

APÊNDICE II - QUESTÕES DE INVESTIGAÇÃO, OBJETIVOS E, PROBLEMAS/OPORTUNIDADES DAS PUBLICAÇÕES SELECIONADAS

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
(Al Alamin et al., 2021)	"RQ1. What types of topics are discussed about Low-Code Software Development in Stack Overflow? RQ2. How are the topics distributed across the Low-Code Software Development life cycle phases? RQ3. What Low-Code Software Development topics are the most difficult to answer?"	Secondary-data studies
(Almonte et al., 2020)	"RQ1. Can a recommender system help in class modelling tasks? RQ2. Which recommendation method of relevant attributes, methods and superclasses has the best performance? RQ3. Can hybrid approaches be beneficial for the recommendation of attributes, methods and superclasses? RQ4. Which method performs better when considering user and item coverage in the recommendation of relevant attributes, methods and superclasses?"	Laboratory experiment
(Alsaadi et al., 2021)	"This paper highlights the concept of LCDP (Low-Code Development Platforms) in detail. It aims to explore the factors that make programmers and developers use LCDP. Moreover, it seeks to figure out the reasons that make them prefer to use the traditional programming approach over LCDP."	Survey
(Arora et al., 2020)	"Due to massive digitization in every aspect of life in past several years demand for software competency has increased rapidly by all kinds of industry. As a result, pressure has gone up tremendously on application developer to develop software much more rapidly and more efficiently. A Low Code Application Development Platform (LCADP) has ability to build enterprise level application in rapid space without using very expensive software professionals."	Conceptual
(Bhattacharyya & Kumar, 2021)	"The purpose of this study is to understand the concept of "Low-Code and No-Code" applications and study its scope of application for web designing, rapid	In-depth interviews

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	application development (RAD) and supply chain digitization (SCD).”	
(Bock & Frank, 2021a)	“This paper has two goals. The first is to present the results of an exploratory study of the features of seven low-code development platforms available on the current market. The aim of this study is provide an overview and comparison of the technical capacities and functionalities of these products now sold under the heading of ‘low-code.’ The investigation is governed by an analysis framework covering several major segments of software development. The second goal is to assess the revealed scope of low-code development platforms in relation to research on software development productivity and end-user computing. Are LCDPs designed on the basis of existing research on software design and engineering? Are their components innovative in comparison with available methods, tools, and instruments?”	Conceptual
(Bock & Frank, 2021b)	“1. What are the characteristic features of low-code platforms? 2. How do low-code platforms compare with the current status of research, and what, if any, technological innovations are realized by these platforms? 3. What opportunities for future research arise from the present attention to low-code development?”	Survey Conceptual
(Bragança et al., 2021)	“In this paper, we argue that the major limitation is that these platforms seldom provide access to their metamodel, the access to applications’ models and code is also limited and, therefore, makes it harder to analyze commonality and variability and construct models based on it.”	Case Study
(Butler, 2020)	“Can the current methods be used to estimate effort for applications built with low-code platforms?”	Demonstration, exemplification test
(Cabot, 2020)	“Is there something fundamentally new behind the low-code movement? How does it relate to other concepts like Model-Driven Engineering or Model-Driven development? And what are the implications for researchers in the modeling community?”	Conceptual

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
(da Cruz et al., 2021)	"Despite low-code and no-code platforms rising in popularity, to the best of the authors' knowledge, the literature discussing how to build them is nonexistent, and the advantages and disadvantages are not well documented. Thus, the main contributions of this paper can be summarized as follows: •An in-depth review of the state of the art on low-code and no-code platforms; •Presentation of low-code functional and nonfunctional requirements; •Proposal of an architecture for low-code development platforms; •Proposal, demonstration, and validation of OLP (Open Low-Code Platform), a lowcode solution that enables regular users to create applications; •Invaluable lessons learned throughout the project."	Demonstration, exemplification test
(Di Ruscio et al., 2022)	"In this expert-voice paper, the authors compare and contrast low-code and model-driven approaches, identifying their differences and commonalities, analysing their strong and weak points, and proposing directions for crosspollination."	Focus Group
(Di Ruscio et al., 2021)	"The objectives of the LowCode workshop are to: •Identify the open challenges that vendors and users of low-code development platforms face; •Identify solutions from the MDE community that may be ported/adapted in the context of low-code development."	Workshop
(Di Sipio et al., 2020)	"C.1 How is it possible to simplify the development of recommender systems in software engineering? Developing recommender systems can be a daunting task. We believe that an LCDP can simplify and speed up the overall process. C.2 What are the modeling constructs needed to support the specification of a new RSSE? It is necessary to investigate in detail a possible set of features and create a model capable of representing them. Taking into account existing work, we define a feature model representing all the relevant RSSE features. C.3 Can such kind of systems support modeling activities? Supporting modeling activities (e.g., metamodeling, and development of model-to-model or model-to-code transformations) could be a daunting task, as it is an activity extremely bounded	Demonstration, exemplification test

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	with the personal developer's experience. Investigating possible applications of the LCPD to develop modeling assistants which can disclose interesting and research directions."	
(Dushnitsky & Stroube, 2021)	"How do low-code tools affect growth trajectory and entrepreneurial success? How do they change the resources required to scale-up and grow?"	Secondary-data studies
(ElBatanony & Succi, 2021a)	"The first problem is analyzing the distribution of the features in popular modern software applications. The second is to figure out how no-code tools help software developers in creating software faster, especially at the starting phase of a project. The underlying goal of this research is to optimize the software development process."	Secondary-data studies
(ElBatanony & Succi, 2021b)	"This paper provides a highly opinionated and biased vision and a two-stage plan with guidelines to reach a new era of software development, where anyone can create software without bothering to write code. Moreover, this paper explores in depth the first of these stages, which consists of creating a no-code tool based on six principles: configuration driven development, APIs, open-source, cross-platform, cloud computing, and design systems."	Conceptual
(Henriques et al., 2018)	"• RQ1. Is the original Business Process Technology concrete syntax semantically opaque? • RQ2. Can participants unfamiliar with Business Process Technology design more semantically transparent symbols for Business Process Technology than the original? • RQ3. Which concrete syntax (original, stereotype, prototype, proposed) is more semantically transparent? • RQ4. Which concrete syntax (original, proposed) leads to a better understandability of Business Process Technology models in the context of interpretation tasks? • RQ5. Which concrete syntax (original, proposed) leads to a better understandability of Business Process Technology models, as perceived by practitioners after performing model interpretation tasks? • RQ6. Which concrete syntax (original, proposed) leads to a lower cognitive effort, as perceived by	Survey Demonstration, exemplification test

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	practitioners while performing Business Process Technology model interpretation tasks?"	
(Horváth et al., 2020)	"(RO1.1.) Isolation of tenants while maintaining fair resource distribution between them. (RO1.2.) Tenant-aware model management services with optimized memory and model access. (RO1.3.) High-throughput lock-free and lock-based concurrent model access mechanisms. Parallel reactive model transformations. (RO2.1.) Find the independent model transformation rules that can be executed in a task-parallel execution mode to achieve high degree of parallelism. (RO2.2.) Transactional model management with model synchronization primitives to allow parallel model editing and guarantee model consistency. (RO2.3.) Alternatively, adopt different lock-free mechanisms for reactive transformations. (RO2.4.) If the task-parallel execution does not provide satisfactory results, then the data-parallel approach has to be studied for reactive transformations. (RO2.5.) Exploit declarative languages and static analysis to derive efficient imperative transformation code. Multi-tenant, reactive model transformation benchmark. (RO3.1.) Empirically identify the Key Performance Indicators of reactive model transformations on multi-tenant platforms. (RO3.2.) Derive evaluation criteria that can be used to assess the model transformation engines. (RO3.3.) Design transformation rules and models that are similar in complexity to real-world scenarios. (RO3.4.) Implement a benchmark workflow that runs the transformation rules on the engines using the source models along the reactive scenario descriptions".	Conceptual
(Hurlburt, 2021)	"As low-code assembly becomes more straightforward and begin to scale or Artificial intelligence (AI), machine learning (ML), and robotic process automation (RPA) stitch workable environments together, will the IT professional be able retain such value within the organization?"	Conceptual
(Ihirwe et al., 2020)	"This paper shows the current state of research on model driven engineering approaches for IoT by taking	Literature Review

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	into account low-code development platforms in particular.”	
(Khorram et al., 2020)	“Low-code introduces new concepts and characteristics. However, it is not investigated yet in academic research to point out the existing challenges and opportunities when testing low-code software. This shortage of resources motivates this research to provide an explicit definition to this area that we call it Low-code testing.”	Conceptual
(Kukesh & Brandenburg, 2018)	“The background of the students coming into Systems Analysis and Design classes is diverse and broad. During the curriculum they are introduced to Unified Modeling Language (UML), Business Process Modeling Notation (BPMN) and other various components to build systems. The tools used to teach these concepts are diverse and sparingly used in the industry. In addition, they do not offer students a tangible and working system solution that resolves business challenges. This disconnect is most noticeable upon graduation when many students are unfamiliar with common integrations used in industry.”	Workshop
(Lourenço et al., 2021)	“The challenge we tackle with this paper is on how to reuse pre-assembled applications with strong safety guarantees. We aim at having a template instantiation mechanism that produces well-formed code upon the validation of the arguments used to ground its parameters. Creating a screen from scratch, based on a sophisticated design, is cumbersome and can take a long time to get right. Cloning screen templates and ad-hoc adaptation of code does solve this problem. However, adaptation requires deep knowledge of each template internals, thus not suitable for all levels of expertise. Our goal is to remove the need for ad-hoc adaptation in the use of templates and replace it by the smart shaping of components to a set of contextualized arguments. Our goals include having fully functional applications right after instantiation, with all arguments and encoded adaptations in place.”	Demonstration, exemplification test
(Luo et al., 2021)	“RQ1: What is the understanding of Low-Code Development? RQ2: What platforms are used in Low-Code Development?	Secondary-data studies

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	RQ3: What programming languages are used in Low-Code Development? RQ4: What are the major implementation units in Low-Code Development? RQ5: What are the supporting technologies used in Low-Code Development? RQ6: What types of applications are developed by Low-Code Development? RQ7: What are the domains that use Low-Code Development? RQ8: What are the benefits of Low-Code Development? RQ9: What are the limitations and challenges of Low-Code Development?"	
(Metrôlho et al., 2020)	"The business innovations mean that IT teams have to grow in numbers and in new tools, putting unheard of pressure on the level of skilled human resources as well as on the technologies they choose to scale their operations while having highly controlled budgets. While reskilling strategies has already been a validated option, these cannot be isolated in qualifying and building multidisciplinary teams for the development of the intended digital transformation. Teams lack, in addition to different profiles with different technical skills, a mix of proficiency and experience in various technical subjects that a person who has attended a reskill program typically does not have due to the nature of their academic background, but mostly due to the technological inexperience in the first 2 to 3 years. Thus, the hypothesis we also intend to explore is the addition of new digital skills (upskill), to a population that has a professional background related to programming. Their knowledge is of technologies that are out of date or that will disappear from the market in the coming years (dreaded languages as in Stack Overflow annual report)."	Case Study
(Oteyo et al., 2021)	"However, to the farmer, it can be difficult to know which kind of LCDTs to choose and what category of these tools are best suited for the task. In this paper, we contrast different LCDTs and show how to use DisCoPar to develop SAAs by non-expert programmers. As a contribution, this paper presents properties for LCDTs that can be beneficial to farmers	Demonstration, exemplification test

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	and demonstrates how DisCoPar can be used in developing SAAs.”	
(Overeem & Jansen, 2021)	“This paper proposes the Impact Analysis for Low-Code Development Platforms framework, a conceptual framework that supports the discussion, research, and implementation of impact analysis. The proposed framework describes the different subsystems and artifacts in a low-code development platform, the different types of professionals involved, and how these professionals can use impact analysis to support their engineering decisions.”	Case Study
(Philippe et al., 2020)	“Low-code development platforms are taking an important place in the model-driven engineering ecosystem, raising new challenges, among which transparent efficiency or scalability. Indeed, the increasing size of models leads to difficulties in interacting with them efficiently. To tackle this scalability issue, some tools are built upon specific computational strategies exploiting reactivity, or parallelism. However, their performances may vary depending on the specific nature of their usage. Choosing the most suitable computational strategy for a given usage is a difficult task which should be automated. Besides, the most efficient solutions may be obtained by the use of several strategies at the same time.”	Conceptual
(Pichidtienthum et al., 2021)	“In this study, a module generator for Odoo using a low-code development concept was proposed.”	Laboratory experiment
(Ragusa & Henriques, 2018)	“Code review is a common practice in the software industry, in contexts spanning from open to close source, and from free to proprietary software. Modern code reviews are essentially conducted using cloud-based dedicated tools. Existing review tools focus in textual code. In contrast, support of low-code software languages, namely Visual Programming Languages (VPLs), is not readily available. This presents a challenge for the effectiveness of the review process with a VPL.”	Demonstration, exemplification test
(Sahay, Di Ruscio, et al., 2020)	“The problem that we want to address in the medium term is about the specification of complex workflows in LCDPs. “	Conceptual

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
(Sahay, Indamutsa, et al., 2020)	"The final aim is facilitating the understanding and the comparison of the low-code platforms that can best accommodate given user requirements. The contributions of the paper are summarized as follows: • Identification and organization of relevant features characterizing different low-code development platforms; • Comparison of relevant low-code development platforms based on the identified features; • Presentation of a short experience report related to the adoption of LCDPs for developing a simple benchmark application."	Technical survey
(Sahinaslan et al., 2021)	"The dependence on technology, digital transformation, and the need to work remotely are increasing day by day. It is predicted to increase further after this COVID-19 pandemic. The desire to digitize every object also leads to the need to develop or update many applications software. It is very difficult with traditional software development methods to produce flexible solutions to such dynamic and changing demands on time. At the same time, there are problems such as finding qualified human resources and high cost in writing and updating corporate program codes that can be considered complex on a large scale."	Case Study
(Sanchis et al., 2020)	"RQ1. What is the status about existing automation software development tools? RQ2. What are the potential challenges in the context of automation software development tools for further research?"	Case Study
(Silva et al., 2021)	"The aim of the study was the development and evaluation of a Descriptive Cognitive Model (DCM) for the identification of three types of usability issues in a low-code development platform (LCDP)."	Laboratory experiment
(Strømsted et al., 2018)	"Concretely, we show how advanced DCR features such as data, guarded relations, event types, DCR Forms and sub-processes allow for programming advanced adaptive case management solutions with forms and data without a single line of traditional code."	Case Study
(Varajão, 2021)	"It is in this context that the software development project described in this article is worth reporting on	Case Study

Artigos	Estudos realizados sobre o tema	
	Questões de investigação, objetivo, ou problema/oportunidade	Tipo de estudo
	since it involves several disruptive aspects that are fundamental in a world that requires solutions "thought today" to be "made available yesterday." From the point of market-opportunity awareness to the availability of a fully functional software product, this project took three weeks to complete and involved several state-of-the-art practices and tools: fast decision making; agile project management; and extreme low-code softwaredevelopment technology."	
(Waszkowski, 2019)	"Unfortunately for many factory managers automation is not the foreground issue. For many companies, it requires them to step into an entirely new world of development and data mastery and tackle completely new challenges like: • Acquiring and paying for expert development skills, • Understanding user experience to create intuitive applications, • Manually maintaining applications and automated process to ensure minimal downtime. Those technical and development challenges were the reason why companies did not decide to automate processes"	Conceptual
(Woo, 2020)	"The use of such no-code platforms—and their low-code cousins, which require some minimal coding knowledge—is on the rise, a trend that represents a step toward a decades-long goal in computer science to automate coding. "	Case Study Conceptual

APÊNDICE III - PRINCIPAIS CONCLUSÕES DOS ESTUDOS IDENTIFICADOS NA LITERATURA

Artigos	Estudos realizados sobre o tema Principais conclusões
(Al Alamin et al., 2021)	"We present an empirical study that provides insights into the types of topics low-code developers discuss in Stack Overflow (SO). We find 13 low-code topics in our dataset of 4.6K Stack Overflow posts (question + accepted answers). The posts are collected based on 19 Stack Overflow tags belonging to the popular nine Low-Code Software Development platforms during our analysis. We categorize them into four high-level groups, namely Customization, Platform Adoption, Database, and Integration. Our findings reveal that developers find the external API Integration topic category the most challenging and the Database category least difficult. Dynamic Event Handling is the most popular, as well as the most challenging topic. We find a severe lack of good tutorial-based documentation that deters the smooth adaptation of Low-Code Software Development."
(Almonte et al., 2020)	"In this paper, we have introduced a generic framework to automate the construction of RSs that assist "citizen developers" in creating software applications via LCDPs. The framework provides a domain specific language to customize the different aspects of the recommender system. These include the language the recommender system is built for, the recommendation algorithm and its parameters, and the evaluation method. We have illustrated the approach by creating a recommender of attributes, methods and super classes, which we have evaluated on three datasets using different recommendation methods. Some of these methods performed quite well, with results similar to those reported in the literature. Overall, the main advantage of our proposal is the flexibility to define recommender systems according to the recommender system designer needs. We are currently developing the concrete syntax of our domain specific language to configure the recommender system."
(Alsaadi et al., 2021)	"This study investigated the factors that lead to utilize LCDP and the challenges of utilizing it. Also, it figured out the factors that prevent some developers or affect their decision about using LCDP. The research targeted professional developers from IT departments in different businesses and students of computing-related departments in Saudi universities. Many programmers and developers preferred LCDP over the traditional programming approach. Some of the factors that attracted them to LCDP were a reduction in app development time, the ease of use, the automatic code generation, the providing of suggestions for the next step, and lower error rates. Yet, developers and programmers mentioned some problems that were faced during the use of LCDP. The highest-rate

Artigos	Estudos realizados sobre o tema Principais conclusões
	problem was difficulties in customizing some functions that were built into the platforms in advance. As with any technology, although LCDP provides a set of benefits for programmers and developers, they face some issues and challenges while using it. They are major concerns that affect the developer's decisions toward adopting LCDP for application development. In some cases, developers avoid using LCDP and prefer traditional programming. Limited scalability, security risks, and problems integrating the apps developed by LCDP with other systems are some of these challenges. In addition, more studies and assessments must be done by researchers to find out and suggest appropriate solutions to the issues and challenges of LCDP. Addressing these issues may help to improve the developers' and programmers' experiences with LCDP. Also, this could attract those who do not use LCDP to start using it. The most significant challenges that should be considered are the limited level of scalability for the developed apps and the security issues generated while using LCDP. Furthermore, the LCDP providers are encouraged to take steps toward improving the developer's experience while using their platforms. Developers should have more flexibility for customizing the built-in functions in these platforms. Moreover, increasing the level of security of the apps developed with LCDP must be taken into consideration by the providers. Also, the automatically generated code should be more readable, written with evident comments, and use meaningful variables names. This will make it easy to maintain and change the code."
(Arora et al., 2020)	"Though S3 and NeoAnalytics are specialized with Pension domain and fraud detection but Sagitec framework and Application Management Services are domain neutral. One can use these components to build application in another domain like banking, healthcare, communication etc. One can also add component like data analytics, machine learning etc. as separate engine which will work with Sagitec framework and provide necessary functionality to the end users."
(Bhattacharyya & Kumar, 2021)	"Given the increased digitization penetrating into supply chain, application development became an essential part to track and monitor process. Organizations often relied on and consulted third party for creation of these applications. Thus, the organizations needed to invest a lot of money, effort and time in the process of finalizing tech giants, specifying them their software requirements and so on. "Low-Code and No-Code" applications have been able to mitigate this pressure point. "Low-Code and No-Code" applications and tools allowed citizen developers that is users with no formal knowledge of coding to create in-house customized applications. This would potentially reduce lead times of software development life cycle (SDLC). The data stored in these applications could improve supply chain visibility across functions. The real time information could be used for analytics and data driven

Artigos	Estudos realizados sobre o tema Principais conclusões
	decision-making. Additionally, it could also increase the speed to market of applications with subsequent order and releases as when required. “Low-Code and No-Code” applications also provided the flexibility to make continuous changes and iterations in apps as per customer requirements. In this study, the authors explored the concept of “Low-Code and No-Code” applications and its scope of application for supply chain digitization, web designing and rapid application development. A cumulative of extant literature review studies regarding “Low-Code and No-Code” applications and supply chain digitization was analyzed. Semi structured open-ended in-depth interviews were also conducted. As revealed from the research, it was concluded that “Low-Code and No-Code” applications could be developed and used across end-to-end supply chain. This could reduce organizations dependency on outsourcing information technology (IT) developers for applications development. In case of procurement, “Low-Code and No-Code” applications could improve vendor and supplier management by streamlining processes. It was noted that “Low-Code and No-Code” applications could streamline logistics management with the unified interfaces and data-driven processes needed for successful shipment lifecycles. It was, thus concluded that the introduction of “Low-Code and No-Code” applications in supply chain could make processes more responsive, agile and transparent. Themes emerging from the study also highlighted barriers to implementation of “Low-Code and No-Code” applications. Lack of digital ecosystem and low perceived usefulness were found as the most frequent factors. The study revealed that managers lacked the proper framework for adoption of “Low-Code and No-Code” applications technology. The authors have prepared a comparative analysis of Diffusion of Innovations and Technological Organizational Environment frameworks for adoption of “Low-Code and No-Code” applications technology. The rationale to justify the publication of this research study resided in the fact that in the context of “Low-Code and No-Code” applications a comparative analysis between “Diffusion Innovation theory,” (Rogers, 2003) versus “Technology-Organization Environment” (Tornatzky and Fleischer, 1990) was missing. Given the advent of “Low-Code and No-Code” applications technology a comparative narrative explaining the value added by “Low-Code and No-Code” applications were of new knowledge exclusively provided by this study.”
(Bock & Frank, 2021a)	“Our investigation of seven low-code development platforms does not lend support to the hypothesis that the systems sold under that heading substantially advance the state of the art. In many ways, they lag behind the frontiers of research on software design, implementation, and maintenance. What distinguishes these platforms is that they integrate, in one environment, multiple well-known and traditional system design

Artigos	Estudos realizados sobre o tema Principais conclusões
	components so as to reduce the efforts of routine tasks in implementing business applications within the confines of certain, more or less restrictive frameworks. Nevertheless, LCDPs deserve the attention of researchers. These platforms are intended to address goals of great social and economic importance in a world permeated more and more by software. Also, because existing LCDPs in large measure build on modeling components, they may help increase the public awareness of conceptual modeling, and (re-)spark research on advanced modeling languages and architectures, among other themes in software design and implementation. But at the same time, the low-code trend, having its source primarily in marketing, gives occasion for critical reflection on the terminology in professional software development. As our study has demonstrated, platforms sold under the label 'lowcode' do not make up a well-defined class of technological environments with a uniform set of features. While we, as academics, are not in a position to prescribe to any practitioner what terms to use, we believe that maintaining a consistent terminology is among our core responsibilities. At present, it is questionable to consider 'low-code' a proper scientific term."
(Bock & Frank, 2021b)	"Our analysis of low-code platforms did not produce evidence that the individual components of low-code solutions are radical innovations. In many ways, they lag behind the frontiers of research on software design, implementation, and maintenance. What distinguishes these platforms is that they integrate, in one environment, multiple well-known and traditional system design components so as to reduce the efforts of routine tasks in implementing business applications within the confines of certain, more or less restrictive frameworks. As such, LCPs can promote software development productivity if all the requirements of a given project can be satisfied within the predefined, immutable framework of a certain LCP and all other pertinent technical and economic conditions are fulfilled, too. Developing methods for the careful assessment of these conditions is an important subject for future research. Moreover, the momentum generated by the low-code trend gives rise to various other inspiring research opportunities lying not only at the core of our discipline, but also at the cross-sections with other disciplines. A most notable opportunity lies in the fact that the attention directed at low-code platforms may contribute to the revival of conceptual modeling. As we have shown, conceptual modeling is one of, if not the single most important ingredient of present-day low-code platforms – even though vendors rarely declare themselves as offering modeling environments. Business information systems research is well positioned to seize this opportunity, since it is the only discipline that takes the design and analysis of domain-specific conceptual models as one of its fundamental tasks. The term 'low-code', however, is problematic. It is currently used in an inconsistent manner, being

Artigos	Estudos realizados sobre o tema Principais conclusões
	deployed to sell vastly heterogeneous development environments. While we do not think it is advisable to prescribe practitioners what terminology to use, we believe that it is our responsibility to critically reflect on whether terms are suitable for incorporation into a proper technical terminology. We do not think the term 'low-code' at present satisfies the criteria required of a scientific concept. After all, language is our most important tool, and we should beware of compromising it."
(Bragança et al., 2021)	"This paper presents an approach that provides support for software product lines engineering in low-code application platforms. The approach does not impose any change in the low-code application platform and only requires that the low-code application platform has some mechanism to import and export application models. The solution is based on a set of domain-specific languages and respective plugins to be used in, what we call, a software product lines integrated development environmental. One domain-specific language is required to model the low-code platform. This domain-specific language is produced semi-automatically, deducing part of its metamodel by inspecting models exported from the low-code application platform. This domain-specific language is also manually customized to incorporate metadata that is not automatically identified and to integrate with a second domain-specific language that deals with variability. The variability domain-specific language is used to express the variability of the software product lines as well as express configuration models specific to particular applications, or products, of the software product lines. There may be also supporting domain-specific languages, such as domain-specific languages to manipulate the data that is exported and imported from the low-code application platform (e.g., JSON files). Once all these tools are available, the low-code application platform users may model the domain and applications of the software product lines using either the low-code application platform or the software product lines integrated development environmental. Variability modeling and application configuration are available only in the software product lines integrated development environmental. A case study is presented and discussed. It addresses the application of the approach to OMNIA, a low-code application platform of a startup company, and its results are very promising. The duration of the case study was about 2 months, including the time to develop all the supporting tools. The users were able to implement a software product line for a scenario that before was only roughly addressed by elementary reuse mechanisms."
(Butler, 2020)	"In conclusion, three of the existing methodologies produced a successful estimation for the low-code development project: COCOMO II, CORADMO and Work Breakdown Structure. The estimates ranged from 2.25 person months to a pessimistic estimate of 10.4 person months with a standard deviation of 2.93 person months. The other approaches were

Artigos	Estudos realizados sobre o tema Principais conclusões
	unsuccessful due to inaccessibility of some parameters to complete the calculations, lack of historical projects for comparison and the dearth of experts to provide estimates.”
(Cabot, 2020)	“As pointed out before, I do not believe there is any fundamental technical contribution in low-code trend. It is more a synonym (or a subset) of the techniques we are already working on. In fact, we could take almost any of the open challenges in model-driven engineering [1] and just change “model-driven” by “low-code” to get, for free, a research roadmap for low-code development (e.g., we need better ways to integrate AI in low-code tools or we should strive as a community to build a shared repository of low-code examples for future research). But I do not see this as being negative. More the opposite. Clearly, low-code is attracting lots of attention, including from people that were never part of the modeling world. In this sense, low-code is lowering the barrier to enter the modeling technical space. As such, to me, low-code is a huge opportunity to bring modeling (and our modeling expertise) to new domains and communities. If we can get more funding/exposure/users/feedback by rebranding ourselves as low-code experts, I am all for it. This is exactly the approach that many well-known so-called low-code companies have taken. Let’s also take this opportunity to better understand the factors that make modeling-like techniques resonate in the broad software community and learn from it. The lack of a strong open source community around the low-code movement could also be a opportunity for us as we have plenty of experience in building opens source modeling tools and discussing the trade-offs of doing so. And while we do that, let’s keep an eye on the market trends to come. Some low-code vendors are shifting (yet again) their marketing efforts. It may not be long before we all start chanting: Low-code is dead, long live multi-experience development.”
(da Cruz et al., 2021)	“This work described a low-code concept which consists of transforming visual representations that are easy to understand into fully fledged applications. The functional and nonfunctional requirements of low-code, the similarities with the no-code concept, and how both concepts are different from traditional programming were also explored. The paper also recognized the difficulty of developing a low-code platform due to the lack of available literature and proposed the Open Low-Code Platform (OLP), a new low-code platform. OLP is a low-code platform built to understand the practical difficulties of creating a low-code platform from scratch. The details of how the visual representations can be transformed into an application are also presented and explained through a pipeline.”
(Di Ruscio et al., 2022)	“This paper compared and positioned the relatively new low-code movement against the established model-driven engineering discipline. We summarized the history of low-code so far, provided an overview of

Artigos	Estudos realizados sobre o tema Principais conclusões
	typical low-code development processes and the tools that LCDPs offer to support them, and contrasted and compared the principles and practices of low-code and model-driven engineering. While low-code and model-driven engineering both aspire to improve software development by raising abstraction and hiding implementation-level details, we argue that the two practices are not identical. Indeed, not all model-driven approaches aim at reducing the amount of code needed to implement software solutions, and not all low-code approaches are model-driven. However, being close conceptually creates substantial potential for applying existing knowledge and cross-pollination between the two disciplines."
(Di Ruscio et al., 2021)	"The growing need for secure, trustworthy and cost-efficient software as well as the availability of mature cloud computing infrastructures, and the high demand for professional software developers, have given rise to a new generation of low-code software development platforms, such as Google AppSheet and Microsoft PowerApps. Low-code development platforms promise the development and deployment of complete applications using graphical interfaces, and thus, only require small amounts of procedural code. This makes them accessible to an increasingly digital-native and tech-savvy workforce who can directly and effectively participate in the software development process, even if they lack a programming background. The adoption of low-code development platforms is growing at a fast pace. As evidence, Gartner expects that "by 2024, low-code application development will be responsible for more than 65% of [business] application development activity". While low-code development platforms are in essence model-driven, there is little evidence that they make use of technologies developed in the Model-Driven Engineering (MDE) community. This highlights a need for increased cross-pollination between low-code developers and model-driven engineering researchers."
(Di Sipio et al., 2020)	"We proposed a novel approach that leads up a new opportunity: allowing software developers to build personalized and well-defined recommender systems using an LCDP. The key points of this kind of platform are flexibility, easy development, and maintainability. The proposed idea is built on well-known concepts, including modeling, and domain specific languages. By exploiting the proposed feature model, we could represent not only the existing recommender systems but also new approaches belonging to this class of systems. The overall architecture will manage to face the main challenges arising from this first study, especially the integration of heterogeneous components and the user's experience. To allow a concrete usage of this platform, a formal specification is needed to handle the complexity of the task, e.g., specifying constraints among the components and take care of the input and output interfaces. The performed domain analysis is reassuring,

Artigos	Estudos realizados sobre o tema Principais conclusões
	recommender systems can be developed in a disciplined manner as they consist of components implementing recurring functionalities (i.e., data preprocessing, capturing context, producing and presenting recommendations). The effort done so far leverages on such commonalities with goal of asking recommender systems developers to focus only on core functionalities, which cannot be abstracted and generalized, e.g., tailored similarity functions or specific encodings of metamodel entities. Thus, we expect hierarchies of DSLs each focusing on specific aspects or phase of the recommendation process."
(Dushnitsky & Stroube, 2021)	"Our study suggests that there may be an alternative route to growth. We find that startups based on a popular low-code tool, Shopify, feature a different growth trajectory. They begin life with fewer financial and human resources, and, despite the more "lean" beginning, they seem to record similar levels of success as measured in term of successful exits. Interestingly, the profile at exit may seem mediocre at a first glance (e.g., lower valuation, fewer employees). But a closer inspection reveals that this masks a unique "lean success" profile; the cash-on-cash return for investors and the value created per employee are well on par with those of their peers. In short, we find that Shopify-based startups require fewer resources to launch and grow but deliver similar value for investors and employees."
(ElBatanony & Succi, 2021a)	"In this work, the Pareto distribution of features in popular mobile applications was analyzed to identify the benefits of using No-Code tools to increase the efficiency of software developers and decrease the development time. The results indicate that developing a no-code tool to help developers in the initial stages of a project could decrease the number of features needed to be developed by almost 50%, allowing the developers to focus on the crucial features that deliver a competitive value"
(ElBatanony & Succi, 2021b)	"In this paper, a two-stage vision of a new software development era where everyone can create software was presented. The first stage of the plan involves the combination of six core principles. Each of these concepts was reviewed and a case was made for their need and benefits. The second stage involves artificial intelligence and natural language processing. Possible enquiries were addressed and lastly, a guideline was provided for creating such a future."
(Henriques et al., 2018)	"While running into maintenance costs, and having identified the need for improving the usability of the commercial business process modelling language (BPT) at OutSystems, we have designed and put forward a systematic process to identify and fix usability issues. We identified issues on syntactic and semantic constructs, and proposed a new notation. After evolving BPT within OutSystem's development environment, we applied the evaluation process to the proposed BPT and observed a significant increase in usability. The comparison analysis between the

Artigos	Estudos realizados sobre o tema Principais conclusões
	original and the proposed version of the BPT confirmed that the process is effective and that the new notation has a higher usability rating. We could also conclude that: Semantic transparency has a large impact on usability; users create more semantic transparent symbols than language engineers; it is extremely important to have a one-to-one relationship between the concrete syntax and the semantic constructs."
(Horváth et al., 2020)	"In this paper, we outlined three research lines to address scalability and productivity challenges in Low-Code Development Platforms. We used a model validation and verification workflow to highlight the challenges in a real-world scenario. Besides, we provided a mapping of the challenges to research lines: •Adoption of multi-tenant architecture patterns for model transformations on low-code platforms, to improve the scalability in terms of number of users, •Parallel reactive model transformations, to improve the scalability of model transformations in terms of model size, •Multi-tenant, reactive model transformation benchmark, to increase productivity by providing different selection criteria that can be used to assess the performance of reactive model transformation engines on multi-tenant platforms."
(Hurlburt, 2021)	"This edition of IT Professional takes a deep dive under the hood to reinforce the importance of the IT professional in a world of growing LCAP and no-code. Each article could well serve as a reference to bolster the key points emphasizing why the IT professional will remain essential to the corporate world."
(Ihirwe et al., 2020)	"Model-driven engineering aim is to tackle different challenges faced in design, development, deployment, and run-time phases of software systems through abstraction and automation. In this study, we discussed state of the art on existing approaches supporting the development of IoT systems. In particular, we focused on languages and tools available in the MDE field and the emergent low-code development platforms covering the IoT domain. The analysis has been performed by conceiving a taxonomy, which has been formalized as a feature diagram presenting all the features of a typical modeling platform supporting the development of IoT systems. Different limitations have been identified in the analyzed platforms."
(Khorram et al., 2020)	"Due to the high trend toward low-code domain and limited academic resources for low-code testing, we performed several analyses in this article. We initially discovered the testing facilities embedded in five well-known commercial LCDPs. Afterward, we proposed a set of 16 features for low-code testing which can be used as criteria for comparing several low-code testing components, and as a guideline for LCDP developers in building new ones. Accordingly, we organized the result of our analysis on the testing component of selected LCDPs based on the proposed

Artigos	Estudos realizados sobre o tema Principais conclusões
	feature list. Our investigations lead us to the identification of existing challenges in low-code testing. We redefined them from a research point of view by providing the state-of-the-art in three main categories including, the role of citizen developer and her low-level technical knowledge in the testing activities, the importance and consequently the challenges in offering high-level test automation, and leveraging the cloud for executing tests alongside supporting testing of cloud-based applications. For each category, we also propose opportunities for future research in low-code testing, such as DSL extension with testing elements, customization of MBT for low-code testing, and supporting the cloud in MBT approaches"
(Kukesh & Brandenburg, 2018)	"Mendix platform has the Agile process built in. Most importantly, it is a full stack platform designed to rapidly develop applications. The platform gives developers control of the various application development layers from front end to back end. It abstracts the main app development components, such as data structures, business logic and user interface. Thus, students are enabled to think more critically about the design and solutions versus focused on learning a programming language with limited capabilities."
(Lourenço et al., 2021)	"In this paper we present a first approach to a rich template language for the OutSystems platform, called OSTRICH. This model conservatively extends the existing model with template annotations, to denote parameters, node properties, special iteration and conditional behavior to model nodes, and template expressions. Our language is a two-stage language that evaluates template nodes and the corresponding annotations, to ground model nodes, with concrete property values, that correspond to a standard OutSystems model. Ground model nodes contain runtime expressions that ultimately define the runtime behavior of an application. We equip our template language and the corresponding template expression language with a verification mechanism that guarantees that all staged expressions are well formed and will produce results of the right type. We evaluated our approach by porting existing screen templates available in the OutSystems platform and produced parameterized versions of the same professionally produced components to be used in a safer, easier and faster way."
(Luo et al., 2021)	"We used LCD related terms to search and collect data from SO and used LCD related subreddits from Reddit to collect data for data extraction and analysis with 301 posts in total. The main findings of this study are the following: • Although there is no clear definition of LCD, practitioners tend to use low-code and drag and drop to describe LCD according to their understanding, showing that LCD may provide a graphical user interface for users to drag and drop with little or even no code.

Artigos	Estudos realizados sobre o tema Principais conclusões
	• The equipment of out-of-the-box units in LCD platforms makes them easy to learn and use as well as speeds up the development. • Different LCD platforms support the development of different types (e.g., mobile and web applications) of applications and different parts (e.g., frontend, workflow, and integration) of applications. • LCD is particularly favored in the domains that have the need for automated processes and workflows. • While LCD platforms can speed up software development with minimal human involvement, they also suffer from no access of source code and vendor lock-in for commercial LCD platforms. Moreover, practitioners have conflicting views on the advantages and disadvantages of LCD, implying that certain features of LCD are beneficial to development if used appropriately, otherwise may become limitations or challenges in LCD. Therefore, developers should consider whether LCD is appropriate for their projects."
(Metrôlho et al., 2020)	"The success of the described case studies shows that low-code platforms can be used to reskill unemployed people or enhance new skills of other workers who may wish to upgrade or redirect their professional activities. The characteristics and potential make them suitable for technicians, or other professionals, with higher qualifications in other areas to acquire new skills through reskilling or supplementing / recycling of pre-acquired basic knowledge that may become productive labor in the IT world. These platforms may be part of a strategy for facing new employability trends."
(Oteyo et al., 2021)	"LCDTs can offer intuitive capabilities for developing smart agriculture applications by both experienced and non-experienced developers. In this paper, we categorize LCDTs and show their properties (such as reconfigurability, visual modelling etc.) that are important to farmers as non-expert programmers. LCDTs that combine these properties can be more beneficial to farmers when expressing and representing smart agriculture applications. As a demonstration, we show how to use DisCoPar to develop smart agriculture applications. From our prototype implementation, we believe that LCDTs can open programming to farmers."
(Overeem & Jansen, 2021)	"Low-code development platforms (LCDPs) accelerate the development of software through new abstractions that remove as much of the technical details as possible. However, challenges such as software evolution remain. Citizen developers, LCDP developers, and operations engineers need tools and processes to solve these challenges. Together these professionals, regardless if they are from the same company or not, are responsible for the success of the application of the LCDP. Evolution plays an important role in that success and this requires that these professionals collect feedback that informs and supports there engineering decisions. We believe that impact analysis helps and

Artigos	Estudos realizados sobre o tema Principais conclusões
	supports these teams in reaching their goals. An overall framework to structure the implementation of impact analysis for LCDPs is missing. In Section III we propose the Impact Analysis for Low-Code Development Platforms framework that conceptualizes the process of impact analysis for LCDPs. It describes the different subsystems and artifacts, together with the process of impact analysis. Using the taxonomy of Lehnert [11] we discussed the variability in the framework and how providers can implement this. Through a case study of an industry LCDP we explore the framework in more depth. Although case studies at other LCDP providers are necessary to evaluate the framework and correct it from any biases, we believe that impact analysis within LCDPs can improve both the applications developed on top of the LCDP as well as the platform itself. Impact analysis should be at the foundations of the LCDP development process."
(Philippe et al., 2020)	"In this paper, we made an overview of what, and how, execution strategies can be used for model driven engineering. In the context of developing low-code platforms for managing models, these strategies might be used for optimizing performances. However, a wrong use of a computational model can have a bad impact on calculation efficiency. The motivating example presented and the implementations illustrate that by using different strategies and different combinations of paradigms for a given input model, different advantages could be observed, such as complexity, and parallelism level. Different paradigms may be chosen, according to different properties: the type of input model, its size, its topology, the type of computation to perform, and the available infrastructure."
(Pichidtienthum et al., 2021)	"From the testing, it can be seen that the time used to create a module was 20% reduced when using he Module Generator compared with the manual coding method. With the obvious difference being usage of a model with fewer data fields, allowing the Module Generator to quickly create the model. Manual coding is even with a few fields, but creating its view still requires a lot of coding, if copying is used to help. This shows that using Module Generator can allow developers to focus on building modules using only data structures. On the contrary, with manual coding, developers have to focus more on writing code correctly- a main reason for taking more time in development. In the testing, a general user was invited to collaborate on the test. This shows that the tool can also be used by a beginner. Odoo user to start building a module for general propose. This should help making Odoo a software that everyone can use o develop software by themselves. Thus, in certain projects, a requires number of experienced developers can be reduced, resulting in cost reduction. Nonetheless, the limitations of this tool are that the generated codes can deploy only data models with views and the user interfaces are not quite user-friendly. "

Artigos	Estudos realizados sobre o tema Principais conclusões
(Ragusa & Henriques, 2018)	"All in all, our main objective with this tool is to support, but not restrict the code review process, providing a flexible and lightweight tool. Our tool minimizes the effort devoted to administrative aspects such as scheduling meetings, encouraging attendance, and recording review comments. Using VPLreviewer makes it effortless to invite reviewers, distribute artifacts and gather feedback. Instead of recording issues on separate log forms, the tool lets reviewers insert their comments in context, right next to the visual code element in question, facilitating discussions among the reviewers on issues that are brought up."
(Sahay, Di Ruscio, et al., 2020)	"The paper gives preliminary study on the usage of model transformation compositions in the domain of low-code development platforms. The problem that we want to address in the medium term is about the specification of complex workflows in LCDPs. The goal is to support citizen developers by providing them with modeling constructs that permit to specify the goal of the desired workflows at a high-level of abstraction. By relying on the techniques and tools developed for composing model transformations, the idea is to generate possible workflows that satisfy the initial goal."
(Sahay, Indamutsa, et al., 2020)	"In this paper, we analyzed eight low-code platforms that are considered as leaders in the related market, to identify their commonalities and variability points. An organized set of distinguishing features has been defined and used to compare the considered platforms. A short experience report has been also presented to discuss some essentials features of each platform, limitations and challenges we identified during the course of development of our benchmark application."
(Sahinaslan et al., 2021)	"The SetXRM low code development tool we examined in this study almost meets all the features expected from such applications. The information received from the manufacturer is that SetXRM is now accepted on a global scale, especially due to its fast and easy development features, and it is also preferred by large-scale companies such as Mercedes. Although this study is based on a product, it has been observed during our research that the application logic of other low code application development platforms is similar and differ in the development interface and preprepared tools. In this respect, SetXRM implementation features will be useful in understanding all other low code platforms"
(Sanchis et al., 2020)	"So, to provide some certainty in this domain, and after analyzing the literature, it can be concluded that the low-code technology as an automation software development tools is under-researched. As the results from the literature are scant, the main features that a low-code development platforms should have and offer (open source, developers' hub, messaging and Pub/Su) have been identified. Then, benchmarking among existing low-code development platforms has been performed against vf-OS. This has resulted in a set of findings that highlight the lack

Artigos	Estudos realizados sobre o tema Principais conclusões
	of transversal platforms developed on open standards to create ecosystems for building and deploying, in an automated manner, apps. All this indicates that the issues addressed in the research question RQ1 have been answered. Based on these findings, vf-OS platform emerges to facilitate the app-building process and to not be dependent on off-the-shelf and third-party software. Based on the RQ2, one of the key challenge is to address the issues of real-time data processing these applications require [6]. In light of this, the fundamental aspect for further study is the linkage between low-code development platforms and IoT devices to consolidate Industry 4.0. Doing so requires new further attempts focused on low-code research for the apps development that facilitates the communication and optimization of intelligent, interconnected equipment and products along the entire value chain. To this end, vf-OS was born. vf-OS is an open multi-sided framework able to manage the overall network of a collaborative manufacturing and logistics environment that enables humans, applications, and devices (IoT) to seamlessly communicate and interoperate in the interconnected environment, promoting digital transformation. vf-OS may be considered as an example from where future research efforts should be directed, taking into account its multi-sided and transversal characteristics. In light of this, one of the main challenges of the current approach is the integration of the different development environments in a single IDE. To provide a satisfactory user experience for developers, it is necessary to deliver a high-level of integration between the different development tools, not only enabling access to the different tools in an integrated environment, but also implementing the same look and feel design and usability considerations across different tools to improve user experience. Also, the development of wizards that guide developers through the development steps across the different development tools can improve significantly the user experience when developing new applications. Another important challenge is the integration of secure code analysis into the development workflows. Currently, the development tools provide static security code analysis reports when building new applications, based on the source code provided. Finally, experts predict that low-code will be possibly crucial for working efficiently and being competitive. In the literature, there are examples of many companies that saw their business continuity in danger because they were not resilient enough and they did not adapt quickly to the new wave of digital technology promoted by Industry 4.0. Enterprises need to develop new digital solutions much faster than their competitors and automation software development tools such as the low-code development platforms may be the response to these new companies' digital requirements."

Artigos	Estudos realizados sobre o tema Principais conclusões
(Silva et al., 2021)	"LCDPs have the potential to dramatically change how software is developed, making it possible, at least for particular domains, for someone without a formal education in computer science to develop quality software, and for experienced developers to significantly speed up the development process. Understanding how programmers and nonprogrammers approach this type of platform is key to support their design and evolution. By developing and applying PRECOG, a new descriptive cognitive model aimed at identifying interaction issues with the learning of low-code platforms, we were able to gain insights into potential problems with a specific low-code platform's use. The proposed descriptive cognitive model was validated, using empirical techniques. Twenty participants were observed interacting with the LCDP, of which 10 were expert programmers and the other 10 were novice programmers. All performed the same tasks, and all interactions were analyzed according to the proposed model. Although a high number of FPs were identified after a first mapping between the user's mental model and the system's requirements, it is relevant to notice that all issues but one (button link), which occurred during users' interaction with the LCDP, were predicted by this mapping. Expert programmers had a higher number of observed issues, although each occurred less frequently. This was due to expert programmer performing the tasks more quickly and with a more explorative behavior, giving room for more issues to occur. On the other hand, novice programmers faced fewer issues, although each occurred more frequently. These results allowed us to successfully identify high-criticality use errors through the analysis of the users' mental model and, importantly, the results allowed us to identify the root-causes of each issue. One of the future goals of the current research is to validate PRECOG as a predictive model without recurring to user studies. PRECOG revealed itself quite valuable in the search for more usable LCDP and effective EUD platforms. As Maeda (2006) points out, "observing what fails to make sense to the non-expert, and then following that trail successively to the very end of the knowledge chain is the critical path to success [that is, in developing simple and easy to learn systems]." Our proposed method allows the systematic and effective exploration of the conflict between users' knowledge and system requirements/challenges, thus providing important insights for system developers that aim at creating a broadly accessible development platform. Moreover, this method can be applied in other contexts where learnability might be an issue for it allows one to identify possible sources of faulty interaction and subtasks where the users' background knowledge will be insufficient to guarantee a successful performance of the task at hand"
(Strømsted et al., 2018)	"Throughout the development process, DCR graph was used as means of developing the process model. The DCR graph was created and improved

Artigos	Estudos realizados sobre o tema Principais conclusões
	upon during meetings with foundation employees. This part involved no code. When the DCR graph was implemented to the ECM system, very little code was used. The DCR graph also entailed that most further developments can be carried out by making edits to the process model with no coding required."
(Varajão, 2021)	"In this project, the challenge was to "deploy software faster than the coronavirus spread." In a project with such peculiar characteristics, several factors can influence success, but some clearly stand out: top management support, agility (in decision and management), understanding and commitment of the project team, and the technology used. Conventional development approaches and technologies would simply not be able to meet the requirements promptly. The project described here reflects the demands currently placed on companies in terms of decision and action capacity. It combines market vision and rapid decision-making capacity with action. The company identified an opportunity, defined a project, and decided to move forward, structuring and organizing a team by adopting a different approach to project management — a streamlined agile approach — and adopting a proactive marketing posture. Without technology that supported the rapid development and deployment of software, however, the project could not have been achieved in such a short time—in a context of high instability and rapid evolution of requirements."
(Waszkowski, 2019)	"The low-code platform Aurea BPM developed as the result of my research and development work is in line with current market demand. Experts see great potential for the development of such solutions. The basic reason for this is the lack of programmers and increasing requirements as to the scope and frequency of changes introduced in IT systems. Manual coding is time-consuming and labor-intensive. In the case of low-code platforms, these obstacles disappear because programming is not necessary to build applications. Using low-code platforms for automating business processes in manufacturing is a new and innovate approach. It can significantly reduce cost and time of implementing, developing and maintaining processes. Furthermore, due to a new possibility to utilize company's internal human resources for analytical and development work, the results of the work would better meet the real needs of business."
(Woo, 2020)	"The use of such no-code platforms—and their low-code cousins, which require some minimal coding knowledge—is on the rise, a trend that represents a step toward a decades-long goal in computer science to automate coding. Without enough software engineers to meet the demand, companies are now turning to these platforms, which are becoming increasingly powerful. Ultimately, AI might be able to automatically produce code, an ambition referred to as ''program synthesis." In principle, a computer can generate a program by going

Artigos	Estudos realizados sobre o tema
	Principais conclusões
	through all the possible code and then identifying the combination that would accomplish the desired task. That is not practical, however, so computer scientists have been developing smarter ways to narrow down the search. Unlike the drag-and-drop platforms, these tools are for coders, and while they do not synthesize code, they aim to help programmers write code faster and with fewer errors. But whether with program synthesis or drag-and-drop platforms, increasingly more people will be able to create programs, freeing up highly-trained coders to focus on the hardest problems. The demand for coders may lessen, but experts will still be needed.”

APÊNDICE IV - DETALHES DAS PUBLICAÇÕES SELECIONADAS

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
1	Sanchis, R., García-Perales, Ó., Fraile, F., Poler, R.	2020	Applied Sciences (Switzerland) 10(1),12	Article	Journal	Scopus
2	Dushnitsky, G., Stroube, B.K.	2021	Journal of Business Venturing Insights 16, e00251	Article	Journal	Scopus
3	da Cruz, M.A.A., de Paula, H.T.L., Caputo, B.P.G., (...), Lorenz, P., Rodrigues, J.J.P.C.	2021	Future Internet 13(10),249	Article	Journal	Scopus
4	Apurvanand Sahay; Arsene Indamutsa; Davide Di Ruscio; Alfonso Pierantonio	2020	Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020 9226356, pp. 171-178	Conference Paper	Conference Proceeding	IEEE
5	Al Alamin, M.A., Malakar, S., Uddin, G., (...), Haider, T.B., Iqbal, A.	2021	Proceedings - 2021 IEEE/ACM 18th International Conference on Mining Software	Conference Paper	Conference Proceeding	IEEE

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
			Repositories, MSR 2021 9463132, pp. 46-57			
6	Sahinaslan, E., Sahinaslan, O., Sabancioglu, M.	2021	AIP Conference Proceedings 2334,070007	Conference Paper	Conference Proceeding	Scopus
7	Butler, Rhonda R.	2020	-	Thesis	-	ProQuest
8	Waszkowski, R.	2019	IFAC-PapersOnLine 52(10), pp. 376-381	Conference Paper	Conference Proceeding	Scopus
9	Kukesh, Julie, and Kat Brandenburg.	2018	MWAIS 2018 Proceedings. 55.	Conference Paper	Conference Proceeding	AIS
10	Arora, R., Ghosh, N., Mondal, T.	2020	Proceedings - 2020 International Conference on Industry 4.0 Technology, I4Tech 2020 9102703, pp. 13-17	Conference Paper	Conference Proceeding	IEEE
11	Pichidtienthum, S., Pugsee, P., Cooharojananone, N.	2021	2021 IEEE 8th International Conference on Industrial Engineering and Applications, ICIEA 2021 9436754, pp. 529-533	Conference Paper	Conference Proceeding	IEEE
12	Bock, A.C., Frank, U.	2021	Proceedings - 24th International Conference on	Conference Paper	Conference Proceeding	IEEE

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
			Model-Driven Engineering Languages and Systems, MODELS-C 2021 pp. 57-66			
13	Lourenco, H., Ferreira, C., Seco, J.C.	2021	Proceedings - 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS 2021, pp. 216-226	Conference Paper	Conference Proceeding	IEEE
14	Overeem, M., Jansen, S.	2021	Proceedings - 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS-C 2021 pp. 88-97	Conference Paper	Conference Proceeding	IEEE
15	Davide Di Ruscio; Dimitris Kolovos; Juan De Lara; Massimo Tisi; Manuel Wimmer	2021	2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS-C)	Conference Paper	Conference Proceeding	IEEE
16	Elbatanony, A., Succi, G.	2021	BCNC 2021 - Proceedings of the 1st ACM SIGPLAN International	Conference Paper	Conference Proceeding	ACM

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
			Workshop on Beyond Code: No Code, co-located with SPLASH 2021, pp. 18-22			
17	Elbatanony, A., Succi, G.	2021	BCNC 2021 - Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code, co-located with SPLASH 2021, pp. 29-35	Conference Paper	Conference Proceeding	ACM
18	Khorram, F., Mottu, J.-M., Sunyé, G.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 490-499	Conference Paper	Conference Proceeding	ACM
19	Cabot, J.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 535-537	Conference Paper	Conference Proceeding	ACM
20	Luo, Y., Liang, P., Wang, C.,	2021	International Symposium on Empirical	Conference Paper	Conference Proceeding	ACM

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
	Shahin, M., Zhan, J.		Software Engineering and Measurement			
21	Di Sipio, C., Di Ruscio, D., Nguyen, P.T.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 471-479	Conference Paper	Conference Proceeding	ACM
22	Bragança, A., Azevedo, I., Bettencourt, N., (...), Teixeira, D., Caetano, D.	2021	GPCE 2021 - Proceedings of the 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, co-located with SPLASH 2021, pp. 16-28	Conference Paper	Conference Proceeding	ACM
23	Horváth, B., Horváth, Á., Wimmer, M.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 441-450	Conference Paper	Conference Proceeding	ACM
24	Almonte, L., Cantador, I.,	2020	Proceedings - 23rd ACM/IEEE	Conference Paper	Conference Proceeding	ACM

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
	Guerra, E., De Lara, J.		International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 451-460			
25	Sahay, A., Di Ruscio, D., Pierantonio, A.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 431-435	Conference Paper	Conference Proceeding	ACM
26	Philippe, J., Coullon, H., Tisi, M., Sunyé, G.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 507-516	Conference Paper	Conference Proceeding	ACM
27	Davide Di Ruscio, Dimitris Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi & Manuel Wimmer	2022	Software and Systems Modeling, 1-10.	Article	Journal	Scopus

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
28	Bock, A.C., Frank, U.	2021	Business and Information Systems Engineering 63(6), pp. 733-740	Article	Journal	Scopus
29	Bhattacharyya & Kumar	2021	Journal of Science and Technology Policy Management	Article	Journal	Web of Science
30	Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., Pierantonio, A.	2020	Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020, pp. 522-529	Conference Paper	Conference Proceeding	Scopus
31	Alsaadi, HA (Alsaadi, Hana A.), Radain, DT (Radain, Dhefaf T.), Alzahrani, MM (Alzahrani, Maysoon M.), Alshammari, WF (Alshammari, Wahj F.), Alahmadi, D (Alahmadi, Dimah), Fakieh, B,	2021	Romanian Journal of Information Technology and Automatic Control, 31(3), 123-140	Article	Journal	Web of Science

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
	(Fakieh, Bahjat)					
32	Metrolho, JC (Metrolho, J. C.), Ribeiro, F (Ribeiro, F.), Araujo, R (Araujo, R.)	2020	14th international technology, education and development conference (inted2020)	Conference Paper	Conference Proceeding	Web of Science
33	Silva, C., Vieira, J., Campos, J.C., Couto, R., Ribeiro, A.N.	2021	Human Factors 63(6), pp. 1012-1032	Article	Journal	Web of Science
34	Henriques, H., Lourenço, H., Amaral, V., Goulão, M.	2018	Proceedings - 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, pp. 200-210	Conference Paper	Conference Proceeding	Web of Science
35	Ragusa, G., Henriques, H.	2018	Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2018-October, pp. 287-288	Conference Paper	Conference Proceeding	Scopus
36	George F. Hurlburt	2021	IT Professional, vol. 23, no. 6, pp. 4-7	Article	Journal	IEEE
37	Oteyo, I.N., Pupo, A.L.S., Zaman, J., (...), De	2021	IEEE AFRICON Conference 2021	Conference Paper	Conference Proceeding	IEEE

ID ref.	Autores	Ano	Source	Document type	Source Type	Base Conhecimento
	Meuter, W., Boix, E.G.					
38	Marcus Woo	2020	*Engineering (Beijing, China)*, *6*(9), 960–961.	Article	Journal	Web of Science
39	Strømsted, R.I., Marquard, M., Heuck, E.	2018	Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW 2018-October, pp. 12-16	Conference Paper	Conference Proceeding	Scopus
40	Varajão, J.	2021	Communications of the ACM 64(10), pp. 32-35	Article	Journal	ACM

APÊNDICE V - DIMENSÃO TECNOLOGIA -TIPO

ID ref.	Autores	*Dimensão Tecnologia -Tipo*		
		Low-code	*No-code*	*Low-code* e *No-code*
1	Sanchis, R., García-Perales, Ó., Fraile, F., Poler, R.	X		
2	Dushnitsky, G., Stroube, B.K.	X		
3	da Cruz, M.A.A., de Paula, H.T.L., Caputo, B.P.G., (...), Lorenz, P., Rodrigues, J.J.P.C.	X		
4	Apurvanand Sahay; Arsene Indamutsa; Davide Di Ruscio; Alfonso Pierantonio	X		
5	Al Alamin, M.A., Malakar, S., Uddin, G., (...), Haider, T.B., Iqbal, A.	X		
6	Sahinaslan, E., Sahinaslan, O., Sabancioglu, M.	X		
7	Butler, Rhonda R.	X		
8	Waszkowski, R.	X		
9	Kukesh, Julie, and Kat Brandenburg.	X		
10	Arora, R., Ghosh, N., Mondal, T.	X		
11	Pichidtienthum, S., Pugsee, P., Cooharojananone, N.	X		
12	Bock, A.C., Frank, U.	X		
13	Lourenco, H., Ferreira, C., Seco, J.C.	X		
14	Overeem, M., Jansen, S.	X		
15	Davide Di Ruscio; Dimitris Kolovos; Juan De Lara; Massimo Tisi; Manuel Wimmer	X		
16	Elbatanony, A., Succi, G.		X	
17	Elbatanony, A., Succi, G.		X	
18	Khorram, F., Mottu, J.-M., Sunyé, G.	X		
19	Cabot, J.			X

ID ref.	Autores	*Dimensão Tecnologia -Tipo*		
		Low-code	*No-code*	*Low-code* e *No-code*
20	Luo, Y., Liang, P., Wang, C., Shahin, M., Zhan, J.	X		
21	Di Sipio, C., Di Ruscio, D., Nguyen, P.T.	X		
22	Bragança, A., Azevedo, I., Bettencourt, N., (...), Teixeira, D., Caetano, D.	X		
23	Horváth, B., Horváth, Á., Wimmer, M.	X		
24	Almonte, L., Cantador, I., Guerra, E., De Lara, J.	X		
25	Sahay, A., Di Ruscio, D., Pierantonio, A.	X		
26	Philippe, J., Coullon, H., Tisi, M., Sunyé, G.	X		
27	Davide Di Ruscio, Dimitris Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi & Manuel Wimmer			X
28	Bock, A.C., Frank, U.	X		
29	Bhattacharyya, S.S., Kumar, S.			X
30	Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., Pierantonio, A.	X		
31	Alsaadi, HA (Alsaadi, Hana A.), Radain, DT (Radain, Dhefaf T.), Alzahrani, MM (Alzahrani, Maysoon M.), Alshammari, WF (Alshammari, Wahj F.), Alahmadi, D (Alahmadi, Dimah), Fakieh, B, (Fakieh, Bahjat)			X
32	Metrolho, JC (Metrolho, J. C.), Ribeiro, F (Ribeiro, F.), Araujo, R (Araujo, R.)	X		
33	Silva, C., Vieira, J., Campos, J.C., Couto, R., Ribeiro, A.N.	X		
34	Henriques, H., Lourenço, H., Amaral, V., Goulão, M.	X		
35	Ragusa, G., Henriques, H.	X		

ID ref.	Autores	*Dimensão Tecnologia -Tipo*		
		Low-code	*No-code*	*Low-code* e *No-code*
36	George F. Hurlburt			X
37	Oteyo, I.N., Pupo, A.L.S., Zaman, J., (...), De Meuter, W., Boix, E.G.	X		
38	Marcus Woo			X
39	Strømsted, R.I., Marquard, M., Heuck, E.	X		
40	Varajão, J.	X		

APÊNDICE VI - DIMENSÃO TECNOLOGIA - TERMOS EQUIVALENTES OU COMPLEMENTARES

ID ref.	Dimensão Tecnologia - Termos equivalentes ou complementares																
	LCSD	LCD	LCDT	LCP	LCDP	LCADP	LCAP	NCDP	VPL	WYSIWYG	Arrastar e soltar	Programação gráfica	Engenharia de software do	Programação do utilizador final	Meta-design	Linguagens de modelação	*Quasi low-code*
1																	
2																	
3																	
4																	
5	X																
6					X												
7																	
8																	
9																	
10						X											
11																	
12																	
13																	
14					X												
15																	
16																	
17																	
18					X												

ID ref.	Dimensão Tecnologia - Termos equivalentes ou complementares																
	LCSD	LCD	LCDT	LCP	LCDP	LCADP	LCAP	NCDP	VPL	WYSIWYG	Arrastar e soltar	Programação gráfica	Engenharia de software do	Programação do utilizador final	Meta-design	Linguagens de modelação	*Quasi low-code*
19																	
20		X								X	X	X					
21					X												
22							X										
23																	
24					X												
25					X												
26					X												
27					X			X									
28				X	X			X									
29																	
30																	
31					X								X	X	X		
32																	
33					X								X	X	X		
34																X	
35									X								
36																	
37			X														
38																	
39																	
40																	X

APÊNDICE VII - DIMENSÃO TECNOLOGIA - VANTAGENS

ID ref.	Dimensão Tecnologia - Vantagens											
	Privacidade	Rapidez de desenvolvimento	Redução de custos	Envolvimento de perfis de negócio	Minimização de requisitos instáveis e inconsistentes	Simplificação da construção de software complexo	Curva de aprendizagem mais rápida	Qualquer pessoa pode assumir a tarefa do responsável pelo desenvolvimento de soluções	Construção mais rápida de protótipos	Flexibilidade e agilidade	Redução de erros das aplicações	Manutenção facilitada
1	X	X	X	X	X			X				
2			X									
3			X			X	X	X	X			
4								X				
5		X								X	X	X
6		X										
7		X										
8		X	X									
9		X										
10		X	X									
11		X						X				
12			X					X				
13								X				
14												
15								X				
16		X	X					X				

ID ref.	Dimensão Tecnologia - Vantagens											
	Privacidade	Rapidez de desenvolvimento	Redução de custos	Envolvimento de perfis de negócio	Minimização de requisitos instáveis e inconsistentes	Simplificação da construção de software complexo	Curva de aprendizagem mais rápida	Qualquer pessoa pode assumir a tarefa do responsável pelo desenvolvimento de soluções	Construção mais rápida de protótipos	Flexibilidade e agilidade	Redução de erros das aplicações	Manutenção facilitada
17								X				
18		X	X					X				
19		X										
20		X	X					X		X		
21		X	X					X				
22								X				
23		X						X				
24								X				
25								X				
26												
27		X	X					X				
28			X									X
29			X					X				
30								X				
31		X						X				
32		X						X				X
33								X				
34		X										

ID ref.	Dimensão Tecnologia - Vantagens											
	Privacidade	Rapidez de desenvolvimento	Redução de custos	Envolvimento de perfis de negócio	Minimização de requisitos instáveis e inconsistentes	Simplificação da construção de software complexo	Curva de aprendizagem mais rápida	Qualquer pessoa pode assumir a tarefa do responsável pelo desenvolvimento de soluções	Construção mais rápida de protótipos	Flexibilidade e agilidade	Redução de erros das aplicações	Manutenção facilitada
35												
36								X				
37		X				X		X				
38								X				
39		X		X				X				
40		X			X							

ID ref.	Dimensão Tecnologia - Vantagens											
	Incorporação do feedback dos clientes	Transparência	Experiência do utilizador	Minimização do esforço de codificação manual	Minimização do esforço de instalação, configuração,	Melhor rendimento do investimento	Foco na solução e não nos aspetos técnicos	Reutilização	Tamanho mais reduzido das equipas	Facilidade de Uso	Aumento da velocidade de lançamento das soluções no mercado	Produtividade da tecnologia *low-code*
1												
2				X								
3	X		X								X	
4							X					
5	X			X								
6		X	X	X								
7				X							X	
8	X			X	X		X					
9							X					
10				X		X						
11				X			X				X	
12			X	X				X				X
13								X				X
14												
15				X								
16									X		X	

ID ref.	Dimensão Tecnologia - Vantagens											
	Incorporação do feedback dos clientes	Transparência	Experiência do utilizador	Minimização do esforço de codificação manual	Minimização do esforço de instalação, configuração,	Melhor rendimento do investimento	Foco na solução e não nos aspetos técnicos	Reutilização	Tamanho mais reduzido das equipas	Facilidade de Uso	Aumento da velocidade de lançamento das soluções no mercado	Produtividade da tecnologia *low-code*
17												
18												
19				X								
20			X		X					X		
21												
22							X	X		X		X
23											X	
24												
25												
26				X								
27			X	X	X		X					
28		X						X				X
29											X	
30												
31				X					X			
32												
33				X								

ID ref.	Dimensão Tecnologia - Vantagens											
	Incorporação do feedback dos clientes	Transparência	Experiência do utilizador	Minimização do esforço de codificação manual	Minimização do esforço de instalação, configuração,	Melhor rendimento do investimento	Foco na solução e não nos aspetos técnicos	Reutilização	Tamanho mais reduzido das equipas	Facilidade de Uso	Aumento da velocidade de lançamento das soluções no mercado	Produtividade da tecnologia *low-code*
34			X				X					
35												
36												
37							X					
38												
39	X						X					
40	X			X								

APÊNDICE VIII - DIMENSÃO TECNOLOGIA - DESVANTAGENS

ID ref.	Dimensão Tecnologia - Desvantagens						
	Limitações de escalabilidade	Fragmentação associada aos vários fornecedores	Segurança	Limitações de personalização da interface gráfica	Menor complexidade de desenvolvimento de aplicações	Ausência de código-fonte	Ausência de acesso ao meta-modelo
1	X	X	X				
2							
3			X	X	X	X	
4							
5							
6							
7						X	
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							

ID ref.	Dimensão Tecnologia - Desvantagens						
	Limitações de escalabilidade	Fragmentação associada aos vários fornecedores	Segurança	Limitações de personalização da interface gráfica	Menor complexidade de desenvolvimento de aplicações	Ausência de código-fonte	Ausência de acesso ao meta-modelo
18							
19							
20				X			
21							
22						X	X
23							
24							
25							
26							
27							
28							
29							
30							
31	X		X				
32							
33							
34							
35							
36							

ID ref.	Dimensão Tecnologia - Desvantagens						
	Limitações de escalabilidade	Fragmentação associada aos vários fornecedores	Segurança	Limitações de personalização da interface gráfica	Menor complexidade de desenvolvimento de aplicações	Ausência de código-fonte	Ausência de acesso ao meta-modelo
37							
38			X				
39							
40							

ID ref.	Dimensão Tecnologia - Desvantagens						
	Custo elevado da utilização das plataformas	Dependência face a terceiros	Vulnerabilidade das soluções	Necessidade de codificação em questões complexas e críticas	Capacidade de atualização das soluções	Tempo de execução das soluções	Compreensão do código gerado
1							
2							
3	X	X	X				
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19				X			

ID ref.	Dimensão Tecnologia - Desvantagens						
	Custo elevado da utilização das plataformas	Dependência face a terceiros	Vulnerabilidade das soluções	Necessidade de codificação em questões complexas e críticas	Capacidade de atualização das soluções	Tempo de execução das soluções	Compreensão do código gerado
20	X	X		X			
21			X				
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							X
32							
33							
34							
35							
36							
37							
38		X			X	X	X

ID ref.	Dimensão Tecnologia - Desvantagens						
	Custo elevado da utilização das plataformas	Dependência face a terceiros	Vulnerabilidade das soluções	Necessidade de codificação em questões complexas e críticas	Capacidade de atualização das soluções	Tempo de execução das soluções	Compreensão do código gerado
39							
40							

APÊNDICE IX - DIMENSÃO TECNOLOGIA - OUTRAS ABORDAGENS E FERRAMENTAS

ID ref.	Dimensão Tecnologia – Outras abordagens			
	MDSD	MBSE	MDE	IDE
1				
2				
3	X			X
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23		X		
24			X	X
25				

ID ref.	Dimensão Tecnologia – Outras abordagens			
	MDSD	MBSE	MDE	IDE
26				
27			X	
28				
29				
30			X	
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				

APÊNDICE X - DIMENSÃO TECNOLOGIA - BARREIRAS

ID ref.	Dimensão Tecnologia - Barreiras	
	Falta de estratégias adequadas de adoção disponíveis para as organizações	Pobre e fraca perceção cultural e falta de consciencialização sobre as aplicações.
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		

ID ref.	Dimensão Tecnologia - Barreiras	
	Falta de estratégias adequadas de adoção disponíveis para as organizações	Pobre e fraca perceção cultural e falta de consciencialização sobre as aplicações.
25		
26		
27		
28		
29	X	X
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		

APÊNDICE XI - DIMENSÃO TECNOLOGIA - PERSPETIVAS FUTURAS

ID ref.	Dimensão Tecnologia - Perspetivas futuras						
	IoT	Indústria 4.0	Inteligência artificial	*Chat bots*	*Natural Processing Language*	*Machine learning*	*Robotic process automation*
1	X	X					
2	X						
3							
4							
5							
6	X	X					
7							
8							
9							
10							
11							
12							
13							
14	X						
15							
16							
17			X	X	X		
18							
19							
20							
21							
22							

ID ref.	Dimensão Tecnologia - Perspetivas futuras						
	IoT	Indústria 4.0	Inteligência artificial	*Chat bots*	*Natural Processing Language*	*Machine learning*	*Robotic process automation*
23							
24							
25							
26							
27	X			X			
28							
29							
30	X					X	
31							
32							
33							
34							
35							
36			X			X	X
37							
38			X				
39							
40			X				

APÊNDICE XII - DIMENSÃO PROJETO - METODOLOGIAS DE DESENVOLVIMENTO

ID ref.	Dimensão Projeto - Metodologias de desenvolvimento	
	Agile	*Scrum*
1		
2		
3		
4	X	X
5	X	
6		
7		
8		
9	X	
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		

ID ref.	Dimensão Projeto - Metodologias de desenvolvimento	
	Agile	*Scrum*
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40	X	

APÊNDICE XIII - DIMENSÃO PROJETO - FASES DE DESENVOLVIMENTO DE APLICAÇÕES RECORRENDO A PLATAFORMAS LOW-CODE

ID ref.	Dimensão Projeto - Fases de desenvolvimento de aplicações recorrendo a plataformas low-code					
	Modelação de dados	Definição da interface do utilizador	Especificação das regras de negócio e fluxos de trabalho	Integração de serviços externos	Implementação da solução	Manutenção da solução desenvolvida
1						
2						
3						
4	X	X	X	X	X	
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						

ID ref.	Dimensão Projeto - Fases de desenvolvimento de aplicações recorrendo a plataformas low-code					
	Modelação de dados	Definição da interface do utilizador	Especificação das regras de negócio e fluxos de trabalho	Integração de serviços externos	Implementação da solução	Manutenção da solução desenvolvida
18						
19						
20						
21						
22						
23						
24						
25						
26						
27	X	X	X	X	X	X
28						
29						
30						
31						
32						
33						
34						
35						
36						
37						
38						

ID ref.	Dimensão Projeto - Fases de desenvolvimento de aplicações recorrendo a plataformas low-code					
	Modelação de dados	Definição da interface do utilizador	Especificação das regras de negócio e fluxos de trabalho	Integração de serviços externos	Implementação da solução	Manutenção da solução desenvolvida
39						
40						

APÊNDICE XIV - DIMENSÃO PROJETO – FATORES DE SUCESSO

ID ref.	Dimensão Projeto – Fatores de sucesso				
	Processo de desenvolvimento iterativo	Disponibilidade de infraestruturas de *cloud computing*	Suporte da alta direção	Agilidade (na decisão e gestão)	Compreensão e comprometimento da equipe do projeto
1					
2					
3					
4					
5					
6					
7	X				
8					
9					
10					
11					
12					
13					
14					
15		X			
16					
17					
18					

ID ref.	Dimensão Projeto – Fatores de sucesso				
	Processo de desenvolvimento iterativo	Disponibilidade de infraestruturas de *cloud computing*	Suporte da alta direção	Agilidade (na decisão e gestão)	Compreensão e comprometimento da equipe do projeto
19					
20					
21					
22					
23					
24					
25					
26					
27		X			
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					
38					

ID ref.	Dimensão Projeto – Fatores de sucesso				
	Processo de desenvolvimento iterativo	Disponibilidade de infraestruturas de *cloud computing*	Suporte da alta direção	Agilidade (na decisão e gestão)	Compreensão e comprometimento da equipe do projeto
39					
40			X	X	X

APÊNDICE XV - DIMENSÃO DEVELOPER - TIPO

ID ref.	Dimensão *Developer* - Tipo	
	Não Profissional e Profissional	Não Profissional
1		X
2		X
3	X	
4	X	
5	X	
6	X	
7		X
8	X	
9	X	
10	X	
11	X	
12	X	
13	X	
14	X	
15		X
16		X
17		X
18		X
19	X	
20	X	
21	X	
22	X	
23	X	
24		X
25		X

ID ref.	Dimensão *Developer* - Tipo	
	Não Profissional e Profissional	Não Profissional
26	X	
27	X	
28	X	
29	X	
30	X	
31	X	
32		X
33	X	
34	X	
35	X	
36	X	
37		X
38	X	
39		X
40	X	

APÊNDICE XVI - DEVELOPER - PAPEL DO PROFISSIONAL DE TI

ID ref.	Dimensão *Developer* - Papel do profissional de TI nas organizações
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	X
17	
18	
19	
20	
21	
22	
23	
24	

ID ref.	Dimensão *Developer* - Papel do profissional de TI nas organizações
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	X
37	
38	X
39	
40	

APÊNDICE XVII - DIMENSÃO DESAFIOS - PROBLEMAS TÉCNICOS

ID ref.		Dimensão Desafios - Problemas técnicos				
	Personalização	Adoção de plataformas	Gestão de base de dados	Integração de terceiros	Testes automatizados	Revisão do código
1						
2						
3	X					
4						
5	X	X	X	X	X	
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20	X					
21						
22						

ID ref.		Dimensão Desafios - Problemas técnicos				
	Personalização	Adoção de plataformas	Gestão de base de dados	Integração de terceiros	Testes automatizados	Revisão do código
23						
24						
25						
26						
27						
28						
29						
30						
31	X					
32						
33						
34						X
35						
36						
37						
38						
39						
40						

APÊNDICE XVII - DIMENSÃO DESAFIOS - SELEÇÃO E COMPARAÇÃO DE PLATAFORMAS

ID ref.	Dimensão Desafios - Seleção e comparação de plataformas				
	Interface gráfica do utilizador	Técnicas de desenvolvimento de software	Suporte de interoperabilidade com serviços externos e fontes de dados	Suporte de segurança das aplicações desenvolvidas	Suporte ao desenvolvimento colaborativo
1					
2					
3					
4	X		X	X	X
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

ID ref.	Dimensão Desafios - Seleção e comparação de plataformas				
	Interface gráfica do utilizador	Técnicas de desenvolvimento de software	Suporte de interoperabilidade com serviços externos e fontes de dados	Suporte de segurança das aplicações desenvolvidas	Suporte ao desenvolvimento colaborativo
21					
22					
23					
24					
25					
26					
27					X
28					
29					
30					
31					
32					
33					
34					
35					
36					
37		X	X		
38					
39					
40					

ID ref.	Dimensão Desafios - Seleção e comparação de plataformas				
	Suporte à reutilização	Suporte à escalabilidade	Mecanismos de especificação de regras de negócio	Mecanismos de construção das soluções	Suporte à implantação das soluções
1					
2					
3					
4	X	X	X	X	X
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					

ID ref.	Dimensão Desafios - Seleção e comparação de plataformas				
	Suporte à reutilização	Suporte à escalabilidade	Mecanismos de especificação de regras de negócio	Mecanismos de construção das soluções	Suporte à implantação das soluções
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37				X	X
38					
39					
40					