DESENVOLVIMENTO ÁGIL DE APLICAÇÕES WEB COM PLATAFORMA LOW-CODE/NO-CODE

Marcelo dos Santos de Oliveira [1], Marcel Campos Inocencio [2]

Resumo: Este artigo avalia, por meio do desenvolvimento de um protótipo, como a utilização de plataformas *low code* e *no code* agilizam o processo de criação de APIs. O estudo engloba uma revisão de literatura que ressalta vantagens como agilidade, diminuição de custos, facilidade de uso e democratização do desenvolvimento. A plataforma foi desenvolvida por meio de metodologia ágil, modular, empregando práticas iterativas para geração de APIs RESTful. Optou-se por uma interface intuitiva, integrando com banco de dados e suporte a regras de negócio em JavaScript. Os resultados indicam que a plataforma simplifica a elaboração de APIs de maneira ágil e compreensível, promovendo a economia e a escalabilidade. No entanto, destaca-se restrições na customização e em projetos de maior complexidade. Conclui-se que essas tecnologias possuem um grande potencial, porém necessitam de aprimoramentos para lidar com cenários mais avançados.

Palavras-chave: no code; low code; desenvolvimento; backend.

ABSTRACT: This article evaluates, through the development of a prototype, how the use of low-code and no-code platforms accelerates the process of API creation. The study includes a literature review that highlights advantages such as agility, cost reduction, ease of use, and the democratization of development. The platform was developed using an agile and modular methodology, employing iterative practices for the generation of RESTful APIs. An intuitive interface was chosen, integrating with a database and supporting business rules written in JavaScript. The results indicate that the platform simplifies the creation of APIs in an agile and understandable way, promoting cost-efficiency and scalability. However, limitations were noted in terms of customization and applicability to more complex projects. It is concluded that these technologies hold great potential but require further improvements to handle more advanced scenarios.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), s.omarcelo93@gmail.com

[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), marcel.inocencio@unesc.net

Keywords: no code; low code; development; backend.

1 INTRODUÇÃO

A digitalização requer que as indústrias implementem estratégias tecnológicas para se adaptar às transformações organizacionais. Neste contexto, as plataformas *low code* se sobressaem ao tornar o desenvolvimento de software mais simples, diminuindo a exigência de codificação manual através de interfaces visuais e abstrações de alto nível. Esta estratégia potencializa a eficácia e diminui despesas, facilitando e agilizando a criação de aplicativos (Alves; Alcala, 2022).

Adicionalmente, as plataformas *low code* contribuem para suprir a escassez de programadores altamente qualificados, possibilitando que profissionais sem um conhecimento profundo de programação, conhecidos como "desenvolvedores cidadãos", desenvolvam soluções eficientes com o treinamento adequado. Isso torna o desenvolvimento de software mais democrático e permite uma resposta mais ágil às necessidades do mercado, promovendo a digitalização das empresas (Alves; Alcala, 2022).

A transformação digital tornou-se crucial para a competitividade das organizações, possibilitando a automação de procedimentos e a integração da infraestrutura de tecnologia da informação. Ao implementar soluções digitais, as empresas ampliam sua adaptabilidade e agilidade, assegurando maior eficácia operacional e inovação contínua em um cenário de mudanças constantes (Santos; Aganette, 2023).

O propósito principal deste estudo é simplificar o desenvolvimento de software através da elaboração de Web APIs, empregando métodos *no code* e *low code*. Já os objetivos específicos visam reconhecer as melhores práticas para a criação dessas APIs, detalhar o processo de desenvolvimento, empregar os princípios de *no code* e *low code*, além de avaliar as vantagens e obstáculos dessas tecnologias, incluindo um mecanismo que permita a execução de código dinâmico durante a execução do programa, assegurando maior adaptabilidade e funcionalidade no desenvolvimento das APIs.

As plataformas *no code* e *low code* transformam a criação de software, possibilitando que profissionais sem experiência prévia em programação desenvolvam e customizem sistemas. Tais metodologias tornam o processo de desenvolvimento mais simples por meio de interfaces intuitivas e componentes pré-definidos, tornando-o mais acessível e rápido, simplificando a manutenção e a expansão dos projetos (DeSilva; Ranathunga;

Shangavie, 2023; Kamouchi; Kissi; Beggar, 2023).

Por meio da vantajosa independência oferecida aos usuários empresariais, que podem possuir a capacidade de criar soluções sem total dependência de equipes especialistas em programação, agilizando a introdução de novas funcionalidades e assegurando que os sistemas correspondam de maneira mais eficaz às demandas específicas de cada instituição. Com isso, as plataformas *no code* e *low code* impulsionam a inovação, aumentam a produtividade e fortalecem a competitividade no setor de software (Dendena; Bisognin, 2021; Dhoke; Lokulwar, 2023; Picek, 2023).

As linguagens de programação evoluíram de linguagens de baixo nível, como Assembly, para linguagens de alto nível, como Fortran e Cobol. Isso fez com que a programação se tornasse mais simples. Atualmente, a variedade de linguagens como Java, JavaScript, Python, Rust e Kotlin reflete a evolução contínua da tecnologia, com cada nova linguagem oferecendo soluções para necessidades específicas e enfatizando a importância de entender tanto as tendências atuais quanto o passado para o futuro da tecnologia (Stival; Carlos; Almeida, 2009).

O surgimento de redes sociais, dispositivos móveis e serviços em nuvem, juntamente com plataformas como Google, Amazon, Facebook e Netflix, marcou o início da transformação digital no século XXI. Tecnologias como Big Data, Internet das Coisas (IoT) e inteligência artificial têm sido fundamentais para esse avanço desde 2010 (Alves; Alcala, 2022; Souza, 2022).

O uso estratégico de tecnologias digitais para reestruturar processos, modelos de negócios e relações com clientes, além de modificar a cultura da empresa, é um elemento essencial da transformação digital. Não basta apenas adotar novas tecnologias, as empresas precisam repensar a maneira como operam e geram valor. O termo *low code*, está associado ao desenvolvimento de software com baixa codificação manual, que emprega diagramas visuais e linguagens declarativas para desenvolver aplicações de maneira mais rápida e econômica do que os métodos convencionais (Alves; Alcala, 2022; Souza, 2022).

A Forrester Research, uma empresa americana, introduziu o termo *low code* em 2014 como um método de geração automática de código e desenvolvimento ágil de aplicativos. Uma plataforma chamada *low code* usa um alto grau de abstração de código para criar aplicações. É possível integrar rapidamente uma aplicação ou um conjunto de aplicações a códigos personalizados, implantá-los em ambientes de produção e

gerenciá-los em servidores em nuvem com um único clique (Alves; Alcala, 2022; Souza, 2022).

Na indústria de TI, o uso de plataformas de código baixo torna o desenvolvimento de aplicações mais eficiente, econômico e acessível. Os desenvolvedores podem criar APIs de backend com mais agilidade usando uma abordagem simples e compreensível, aumentando a produtividade, a colaboração e a qualidade do código. Os programadores podem prototipar, alterar e implementar APIs rapidamente com as ferramentas de arrastar e soltar e as opções de geração e personalização de código. Isso reduz o tempo de desenvolvimento e a complexidade.

Plataformas *no code* e *low code* também simplificam o desenvolvimento de aplicativos, possibilitando que especialistas de variados campos trabalhem como programadores e contribuam diretamente para a criação de soluções digitais. O uso de blocos e templates pré-definidos permite que indivíduos sem conhecimento em arquitetura de software desenvolvam aplicações úteis, principalmente para automatizar processos de trabalho e sistemas básicos, diminuindo a demanda por codificação manual (DeSilva; Ranathunga; Shangavie, 2023; Dhoke; Lokulwar, 2023).

Ao diminuir os gastos com desenvolvimento e operação, acelerando o desenvolvimento de aplicativos e automatizando processos, a colaboração entre as equipes de TI e os desenvolvedores cidadãos garante que as soluções satisfaçam as demandas empresariais, possibilitando que as organizações se ajustem de maneira rápida e flexível às alterações do mercado (Brue, 2023).

Quando se trata de sistemas de planejamento de recursos empresariais (ERP), o uso de plataformas de desenvolvimento *no code* e *low code* tem o potencial de mudar a forma como as empresas administram e personalizam esses sistemas. As empresas podem usar esse modelo de desenvolvimento para fazer ajustes e melhorias em seus ERPs de maneira mais ágil, o que é importante em um ambiente de negócios dinâmico. O *no code* e *low code* permite a personalização das funcionalidades do ERP sem precisar de habilidades de programação avançadas, facilitando a adição de novas funções, a personalização da interface e a integração de serviços de terceiros. Isso torna o ERP mais fácil de entender e o adapta às necessidades dos usuários. Além disso, esse método facilita a integração de dados entre vários sistemas, permitindo conexões com várias ferramentas (Picek, 2023).

Apesar das plataformas *no code* e *low code* possibilitarem um

desenvolvimento mais rápido de aplicativos por profissionais que não são especialistas em programação, ainda requerem desenvolvedores experientes para projetos de maior escala, pois a sua implementação pode ser complicada pela ausência de capacitação ou resistência da organização, o que torna crucial uma estratégia com o devido suporte e governança (Brue, 2023; DeSilva; Ranathunga; Shangavie, 2023).

Pode-se citar também uma dificuldade na utilização de plataformas de baixo código em sistemas ERP, onde há possibilidade de alterações substanciais, elevando a chance de ocorrência de problemas operacionais. A elaboração de soluções por usuários que não possuem conhecimento técnico pode comprometer a eficácia e a integração do sistema. Embora acessíveis, essas plataformas podem demandar competências técnicas para assegurar segurança e conformidade nesses sistemas (Picek, 2023).

Muitas empresas utilizam as plataformas *no code* e *low code* para criar rapidamente aplicações acessíveis em navegadores e dispositivos móveis, facilitando a interação com os usuários. Embora haja dúvidas sobre a segurança e precisão dos dados, essas plataformas são usadas no setor de saúde para criar soluções como sistemas de agendamento e gerenciamento de dados de pacientes. Além disso, elas também podem ser utilizadas na automação de processos financeiros, onde a segurança e a conformidade com as regras de negócio ainda são uma preocupação significativa (Kamouchi; Kissi; Beggar, 2023).

O artigo "Utilização dos Conceitos de Low Code e No Code na Geração de Web Services com Arquitetura MDA", dos autores Alairton Dendena e Gustavo Bisognin aborda a criação de uma plataforma que emprega os princípios de *no code* e *low code* para produzir o código-fonte de aplicações, visando diminuir o trabalho de codificação. A plataforma supre a demanda por implementar cadastros básicos, uma atividade habitual no ciclo de vida de sistemas, que demanda um tempo considerável na criação de software.

A plataforma cria APIs RESTful que o usuário pode configurar usando o Spring Framework, reduzindo assim a quantidade de código redundante. A metodologia *low code* possibilita o uso direto do código em ambientes de teste e produção, sem a exigência de modificações. No entanto, caso necessárias novas diretrizes de negócio, o código pode ser modificado, incorporando a versatilidade do *low code*.

O trabalho ressalta ainda a relevância da documentação da API,

produzida automaticamente por meio do uso de bibliotecas como Springfox, simplificando sua utilização pelos programadores. O estudo se fundamenta na Arquitetura Orientada a Modelos (MDA), que emprega modelos tais como o Modelo Independente de Computação (CIM) e o Modelo Independente de Plataforma (PIM) para aprimorar a interação entre especialistas e programadores (Dendena; Bisognin, 2021).

O artigo "Vantagens e Desvantagens do Desenvolvimento de Sistemas de Informação com Tecnologias Low-Code – Uma Revisão de Literatura", dos autores Fernando Araújo e João Varajão, discute a adoção cada vez maior de tecnologias *low code* na criação de software. Os escritores ressaltam que essas tecnologias possibilitam um progresso mais ágil e acessível, permitindo que indivíduos sem conhecimento técnico em programação possam desenvolver aplicativos.

A pesquisa conduz uma análise sistemática da literatura com o objetivo de identificar e debater as vantagens e desvantagens ligadas ao emprego de códigos de baixo nível. Dentre os benefícios, destacam-se a agilidade no desenvolvimento, a diminuição de despesas, a participação de perfis de negócios, a redução de requisitos voláteis e a simplicidade na utilização das aplicações. Em contrapartida, as limitações incluem a necessidade de plataformas específicas e restrições na customização.

Os autores afirmam que, mesmo com as significativas vantagens, ainda é necessário realizar mais estudos científicos sobre o assunto, pois a literatura disponível é limitada. O objetivo do artigo é ampliar a consciência acerca das potencialidades das tecnologias de baixo código, propondo que futuras pesquisas levem em conta também a literatura cinzenta para uma compreensão mais completa do tema (Araujo; Varajao, 2023).

O artigo "Desenvolvimento de Aplicações Web em Plataforma de Low-Code", do autor André Alexandre Ribeiro Matos, discute o uso de plataformas de desenvolvimento *low code*, que possibilitam o desenvolvimento de aplicativos com mínima ou nenhuma codificação, usando interfaces visuais e recursos de arrastar e soltar. A tecnologia OutSystems, amplamente utilizada em todo o mundo, permite o desenvolvimento ágil de aplicações em três paradigmas: Web Tradicional, Mobile e Web Reativa.

O documento aborda a relevância das plataformas *low code* no ambiente profissional, ressaltando como elas simplificam o processo de desenvolvimento, tornando-o mais rápido e acessível. Durante seu período de estágio na empresa Glintt, o autor buscou principalmente adquirir conhecimento sobre os diversos paradigmas da OutSystems, envolver-se ativa-

mente em projetos e implementar boas práticas de desenvolvimento, tanto no lado do backend quanto do frontend.

Ademais, o texto destaca a importância da vivência prática em contextos empresariais, a avaliação técnica de requisitos e a cooperação em grupo, componentes cruciais para o êxito na criação de soluções de software (Matos, 2022).

2 MATERIAIS E MÉTODOS

Com o aumento da procura por soluções tecnológicas ágeis e eficazes, as técnicas ágeis ganham destaque na elaboração de software. Neste cenário, a utilização de plataformas *no code* e *low code* se apresenta como uma opção viável, possibilitando a criação de aplicações web com menor quantidade de código. Para atingir o objetivo deste estudo, desenvolveu-se uma plataforma de desenvolvimento *no code* e *low code*, para desenvolver soluções de web API dinâmicas, acessíveis e escalonáveis.

2.1 METODOLOGIA

Para atingir os objetivos deste trabalho, possibilitando ao usuário a criação da estrutura de um backend para uma aplicação web, definiu-se a seguinte metodologia: levantamento bibliográfico e referencial teórico, estudos de trabalhos correlatos e materiais e métodos.

Os recursos utilizados para o desenvolvimento dessa aplicação para o Backend: Linguagem Java, Framework Spring Boot, IDE IntellJ, Postman, banco de dados MongoDB, Gerenciador de banco não relacional Compass, banco de dados relacional, PostgreSQL, Gerenciador de banco relacional DBeaver e biblioteca de interpretação de java script GraalVM. Já para o Frontend Visual Studio Code, linguagem JavaScript e Biblioteca JavaScript React.

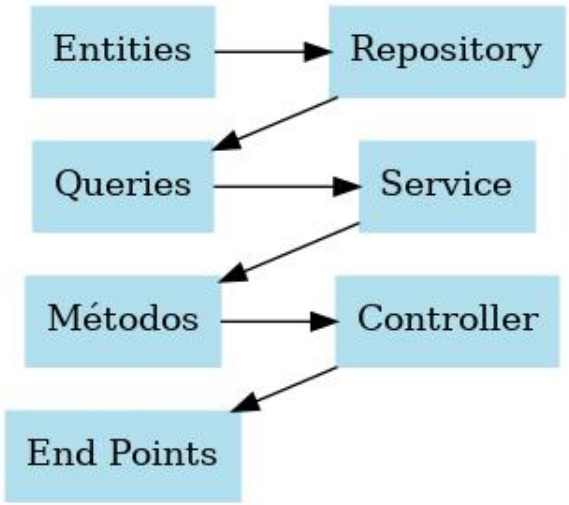
2.2 LEVANTAMENTOS DE REQUISITOS

Os requisitos funcionais e não funcionais da plataforma *no code* e *low code* foram estabelecidos, com ênfase nos seguintes pontos: Capacidade para desenvolver APIs para aplicações web, interface de usuário intuitiva que facilite o desenvolvimento e estruturas sem a exigência de codificação, possibilidade de criação de regras de negócio codificadas pelo usuário em JavaScript e interligação com bases de dados, possibilitando que o usuário configure a conexão com o banco de dados relacional.

2.3 ARQUITETURA DA PLATAFORMA

A arquitetura da plataforma foi projetada com base nos requisitos identificados, respeitando os princípios de modularidade. A arquitetura é formada pelas etapas descritas no fluxograma ilustrado na Figura 1.

Figura 1 - Desenho da Arquitetura Backend



Fonte: Elaborado pelo autor.

As *Entities* representam os modelos e entidades do sistema, estabelecendo as informações fundamentais do registro, já o *Repository* é encarregado de se comunicar com a fonte de dados, encapsulando os métodos de acesso ao banco de dados. Por outro lado, as *Queries* referem-se às consultas específicas de dados realizadas no repositório para persistir ou obter as informações requeridas.

O *Service* é a camada onde a lógica de negócio é armazenada, ele possui os dados das consultas e os manipulam para atender os casos de uso do sistema. Os Métodos representam os casos de usos específicos que a camada de serviço realiza, aplicando as regras estabelecidas. Já o *Controller* atua como intermediário entre as solicitações (como, por exemplo, de uma interface web) e os serviços responsáveis pelo processamento dos dados. Em última etapa, os *endpoints* representam os pontos finais de acesso à aplicação, disponibilizados por meio de APIs, possibilitando a interação com a mesma.

Utilizou-se o MongoDB para armazenar as configurações definidas pelos usuários, bem como os códigos criados para atender as regra de negócio de aplicações criadas na plataforma, após a criação do Backend, desenvolveu-se uma interface gráfica (frontend) web que possibilita ao usuário desenvolver a estrutura de dados e as regras de negócio.

2.4 DESENVOLVIMENTO

O desenvolvimento da plataforma foi organizado e separado em grupos de tarefas por camadas do software, utilizando práticas ágeis como, desenvolvimento iterativo e incremental onde cada tarefa concluída incrementa o projeto final, trazendo valor continuamente.

2.4.1 Desenvolvimento inicial

Para iniciar o desenvolvimento, esquematizou-se a arquitetura do projeto dividindo em funcionalidades, sendo essas configuração de banco de dados, entidades, migrações, repositórios, consultas, serviços, métodos, controlador e *endpoints*. Criou-se *CRUDs* (Create, Read, Update, Delete) para cada uma das funcionalidades citadas e após a conclusão dessa etapa desenvolveu-se um motor genérico de execução *low code*, que no primeiro passo identifica qual *endpoint* esta sendo requisitado na API e posteriormente consulta os dados de serviços previamente cadastrados, como *script*, parâmetros, tipos de dados e por fim executa o *script* e instruções SQL e retorna os dados processados para o usuário.

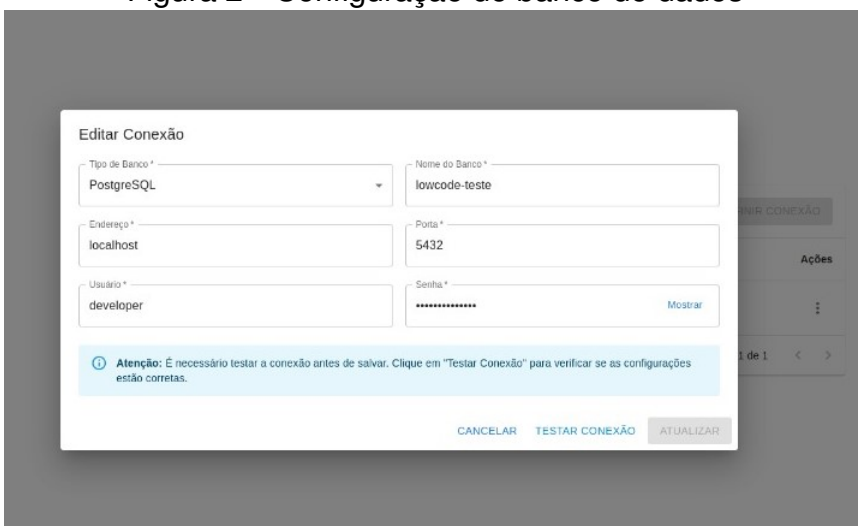
A estrutura adotada para o backend consiste na arquitetura hexagonal, separando a regra de negócio das portas de entrada e saída, favorecendo um elevado grau de independência entre a lógica principal do aplicativo e os componentes externos, tais como bancos de dados, APIs e interfaces do usuário. Esta estratégia simplifica a manutenção, a realização de testes e a troca de componentes externos sem afetar a lógica do sistema. No que diz respeito à parte visual, o frontend foi criado utilizando componentes em *React*, possibilitando a criação de interfaces dinâmicas, reutilizáveis e altamente responsivas, em consonância com os princípios atuais de desenvolvimento web.

2.4.2 Configurações de banco de dados

Para realizar a configuração do banco de dados, que para os fins do protótipo, adotou-se apenas a conexão com postgreSQL, ao acessar a opção "Definir Conexão", faz-se necessário preencher os seguintes campos: tipo do banco, nome do banco, endereço, porta, usuário e senha. Antes de salvar a configuração, faz-se necessário realizar um teste de conexão para verificar se a conexão foi bem-sucedida, para então salvar a configuração. A Figura 2 apresenta um exemplo da configuração de banco de dados preenchida.

Para a criação de uma tabela, primeiro faz-se necessário a definição de uma entidade, para isso é preciso informar o nome, o label e o tipo

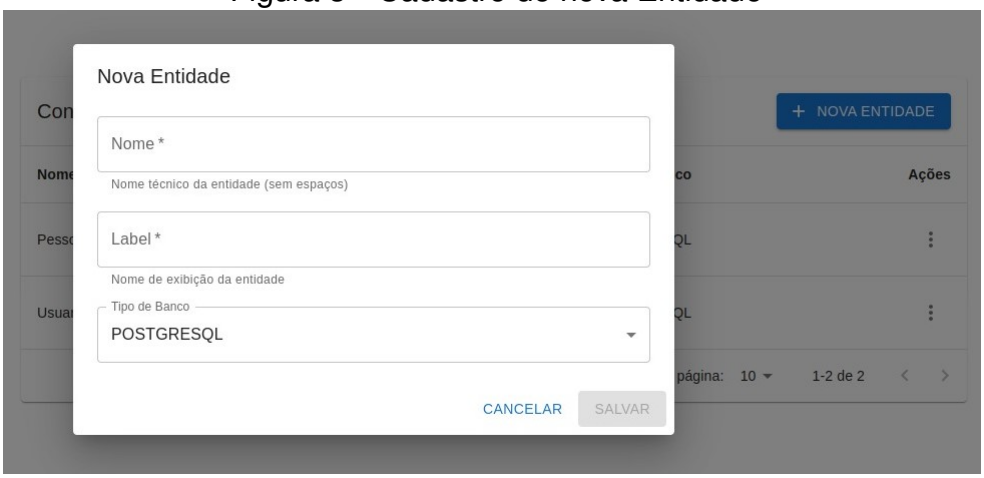
Figura 2 - Configuração do banco de dados



Fonte: Elaborado pelo autor.

de banco de dados e salvar. Os campos preenchidos estão demonstrado na Figura 3.

Figura 3 - Cadastro de nova Entidade



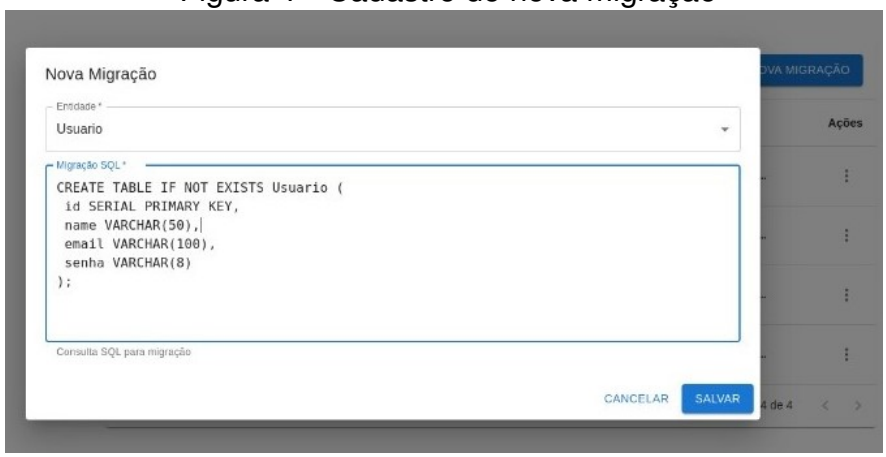
Fonte: Elaborado pelo autor.

Após a criação da entidade, o próximo passo é criar as Migrações SQL. A primeira migração a ser criada é a *Create Table*, utilizando os dados demonstrados na Figura 4. Após a criação, a migração será enviada ao banco de dados, onde é possível consultar o status.

A migração pode ser editada apenas antes da execução. Após a execução, só estará habilitada a opção visualizar. Após a execução da migração é possível verificar no banco de dados que a tabela foi criada.

Com a tabela já criada no banco de dados, um repositório (*repository*) foi criado, preenchendo os dados solicitados, conforme demonstrado

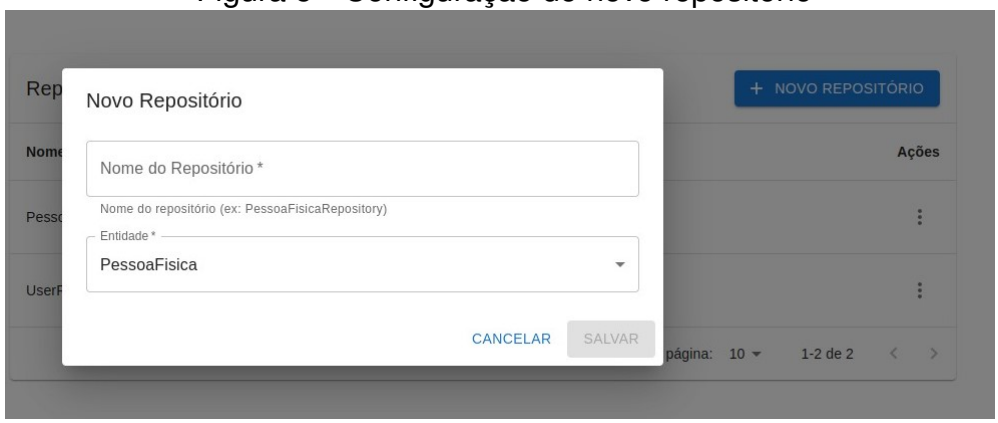
Figura 4 - Cadastro de nova migração



Fonte: Elaborado pelo autor.

na Figura 5. Este *repository* representa a camada responsável por acessar e manipular dados no banco de dados.

Figura 5 - Configuração de novo repositório



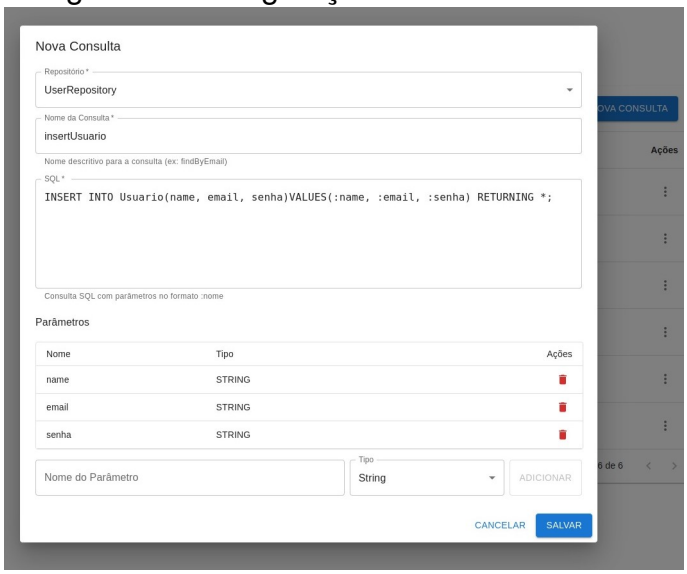
Fonte: Elaborado pelo autor.

Com o *repository* pronto, na seção de consultas (*queries*) criou-se uma nova consulta que pode ser chamada de *insertusuario* (o nome da consulta não precisa seguir o padrão utilizado). No campo SQL, inseriu-se o trecho de código responsável pela inserção dos dados e os demais dados solicitados, demonstrados na Figura 6.

Faz-se necessário adicionar os nomes dos parâmetros, que serão utilizados pelo sistema para injetar dinamicamente os valores na cláusula *WHERE* ou *VALUE* da instrução SQL, quando aplicável.

Uma vez criado o *UserService*, definiu-se o método, onde este funciona como um bloco de código que representa um caso de uso da API. Um exemplo de regra de negócio implementada nesse método é a valida-

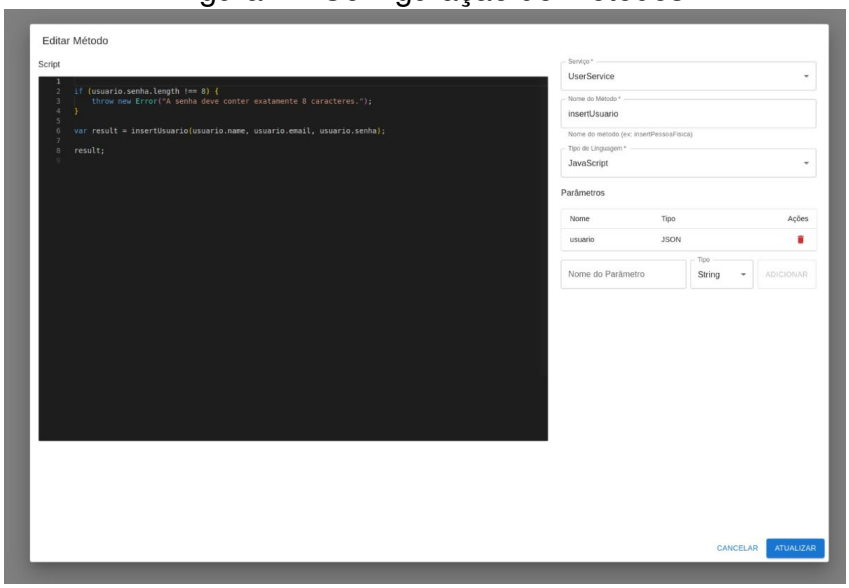
Figura 6 - Configuração de novas consultas



Fonte: Elaborado pelo autor.

ção de que a senha deve conter ao menos 8 (oito) caracteres, se a regra for atendida então os dados são armazenados no banco de dados por meio da instrução SQL criada na Figura 6. O trecho de código descrito na Figura 7 foi utilizado para criar o *UserService*.

Figura 7 - Configuração de métodos

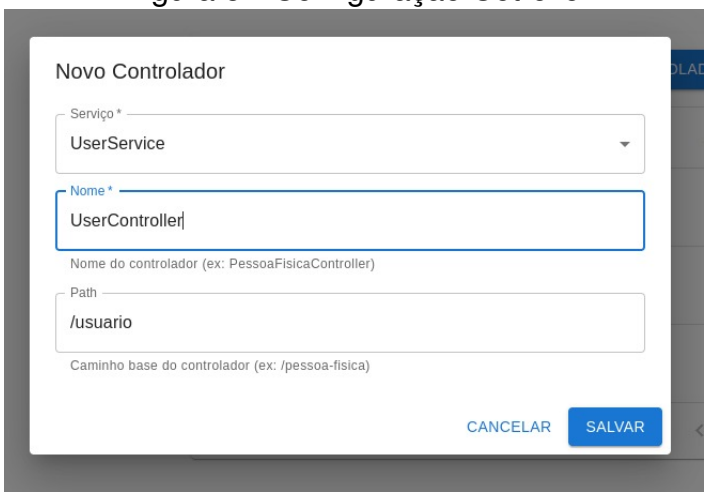


Fonte: Elaborado pelo autor.

O próximo etapa foi a criação de um controlador (*controller*), que se conectará ao *UserService*. Faz-se necessário definir um nome para o *controller*, como sugestão, definiu-se *UserController*, e o *path* (caminho do

recurso) como "/usuario"e salvar, conforme demonstrado na Figura 8.

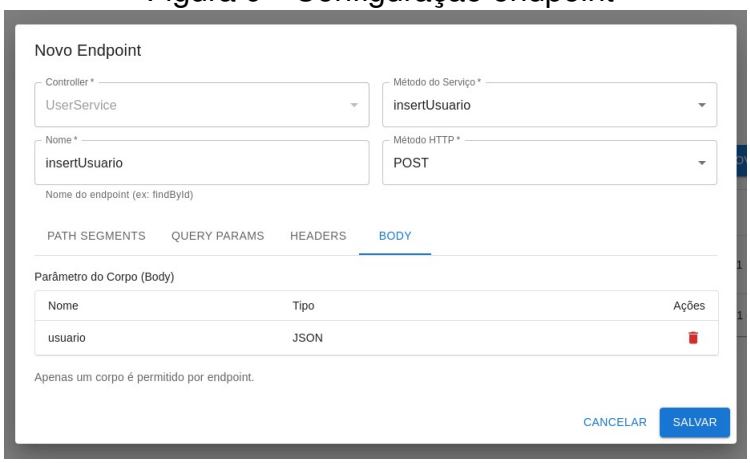
Figura 8 - Configuração Cotroller



Fonte: Elaborado pelo autor.

Para dar sequência, criou-se um *endpoint*. Nessa etapa fez-se necessário definir um nome, referenciando qual método do *service* será utilizado para processar as informações recebidas no *endpoint*, informando qual método *http* esperado pelo *endpoint* e definido um objeto do tipo *json* denominado usuario no *body* da solicitação *http*, ilustrado na Figura 9.

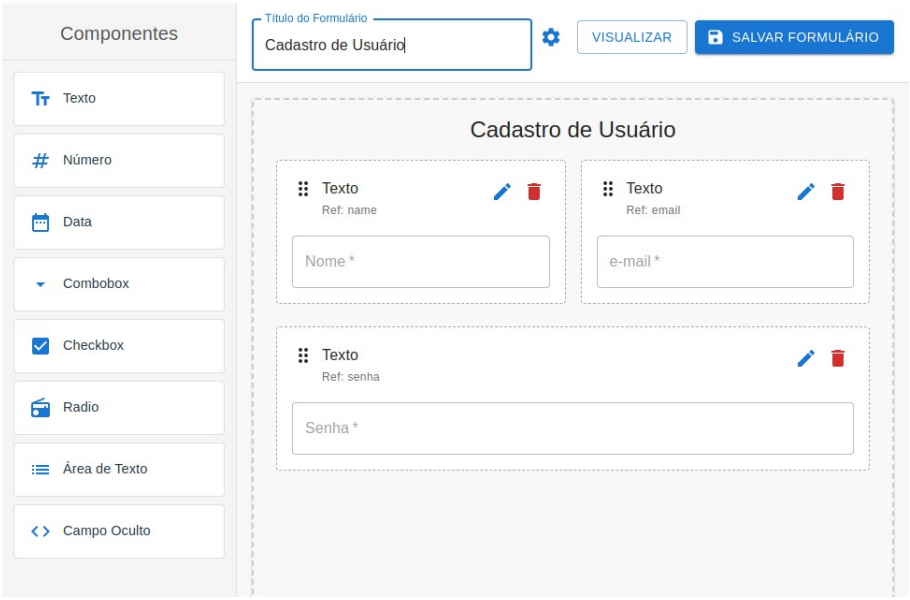
Figura 9 - Configuração endpoint



Fonte: Elaborado pelo autor.

Na Figura 10 é demonstrada a construção do formulário que utilizou o *endpoint* criado anteriormente. Nessa etapa configurou-se endereço da API, o *path* do *endpoint* e definiu-se os campos nome, email e senha que representam o objeto *json* que será enviado no corpo da requisição http.

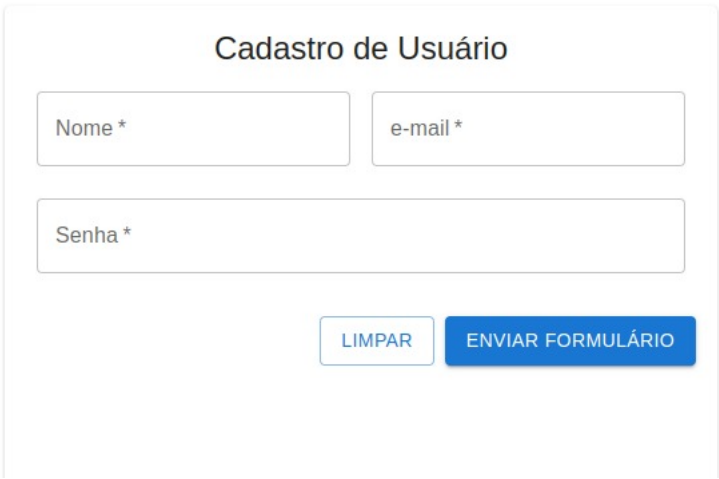
Figura 10 - Configuração de formulário



Fonte: Elaborado pelo autor.

Após a construção do formulário, é possível clicar na opção visualizar onde será exibido o fomulário pronto para uso, conforme exibido na Figura 11.

Figura 11 - Formulário funcionando

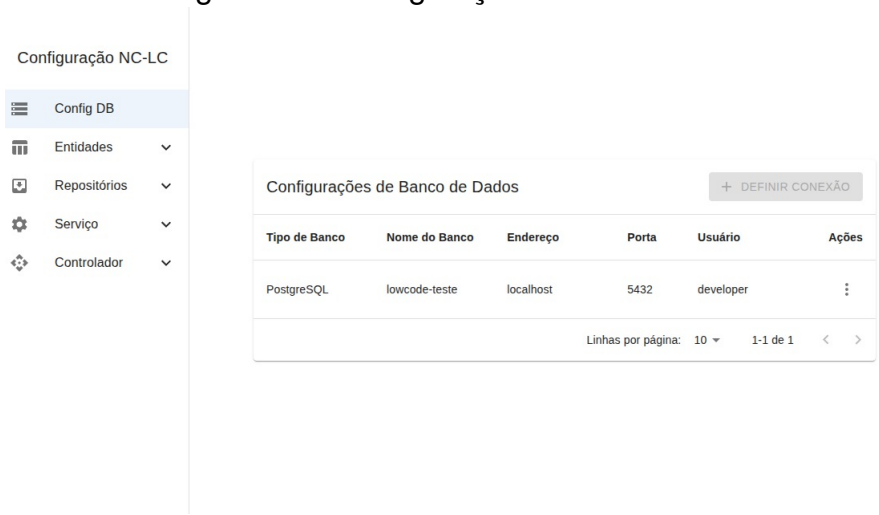


Fonte: Elaborado pelo autor.

3 DISCUSSÃO E RESULTADOS

Após o desenvolvimento da plataforma, pode-se obter os resultados desejados no estudo. A Figura 12 exibe o sistema em sua apresentação final após os cadastros realizados.

Figura 12 - Configuração inicial do sistema



Fonte: Elaborado pelo autor.

Com base nos assuntos discutidos no artigo, os resultados indicam que a plataforma desenvolvida alcançou êxito em cumprir o objetivo do estudo, simplificando o desenvolvimento de software através da elaboração de Web APIs, empregando métodos *no code* e *low code*. O estudo conseguiu estabelecer uma estrutura modular que permite a criação rápida e eficaz de APIs RESTful, diminuindo a complexidade e o tempo de desenvolvimento convencional.

Ademais, as avaliações realizadas com ferramentas como o Postman mostraram que a plataforma é capaz de produzir APIs eficazes e alinhadas com as demandas dos sistemas contemporâneos, o que corrobora sua aplicabilidade prática. A realização de consultas, registro de entidades e realização de operações básicas demonstra que a solução é expansível e proporciona uma interface compreensível para usuários com variados níveis de conhecimento técnico, em consonância com os princípios de democratização do desenvolvimento de software.

A análise e comparação com trabalhos análogos indicam que a plataforma possui benefícios notáveis em termos de rapidez, simplicidade e economia, mesmo que elementos como a personalização profunda ainda possam requerer aprimoramentos ou estratégias extras para cenários mais complexos. Portanto, a pesquisa conclui que plataformas *no code* e *low code* representam uma opção eficaz e viável para a criação de APIs Web em variados cenários organizacionais.

4 CONCLUSÃO

A partir dos resultados obtidos, conclui-se que as plataformas *no code* e *low code* constituem uma opção eficaz, rápida e economicamente viável para a criação de APIs web. Essas ferramentas proporcionam um método acessível, simplificando a elaboração de soluções digitais por especialistas com variados níveis de conhecimento técnico, fomentando dessa forma a democratização da produção de software e contribuindo de maneira relevante para a transformação digital das empresas. Adicionalmente, destacam-se benefícios como a diminuição do tempo de execução, a simplificação dos procedimentos de desenvolvimento e a capacidade de iterações e correções ágeis em contextos de negócios voláteis.

No entanto, ainda persistem restrições significativas, particularmente em situações que requerem dinamicidade e personalizações. Frequentemente, é necessário adotar estratégias adicionais ou melhorar as plataformas já existentes. Essas limitações indicam a necessidade de um desenvolvimento constante e progresso dessas tecnologias, com o objetivo de expandir sua utilização em contextos mais complexos.

Com base no exposto, sugere-se para trabalho futuros superar as limitações identificadas, além de investigar métodos que melhorem a personalização e a integração das plataformas *no code* e *low code* com sistemas já existentes. Prevê-se que, no futuro, essas plataformas possam desempenhar um papel ainda mais significativo na aceleração da criação de soluções digitais, atendendo de maneira mais abrangente às demandas específicas do mercado.

Com isso, propõe-se desenvolver uma interface visual para geração automática de instruções SQL, baseada em critérios estabelecidos pelo usuário, simplificando sua utilização. Sugere-se ainda o aprimoramento do construtor de formulários *form builder*, oferecendo suporte mais completo aos métodos *PUT*, *GET* e *DELETE*, além do progresso do serviço genérico de *low code*, tornando-o mais robusto e apto a reutilizar métodos de outros serviços. Tais aprimoramentos visam melhorar a usabilidade, a integração e a flexibilidade da plataforma.

REFERÊNCIAS

ALVES, F. R.; ALCALA, S. G. S. **Análise da abordagem LOW-CODE como facilitador da transformação digital em indústrias**. [S.l.: s.n.], 2022. v. 15. 1-19 p.

ARAUJO, F.; VARAJAO, J. **Vantagens e desvantagens do desenvolvi-**

mento de sistemas de informação com tecnologias low-code – uma revisão de literatura. 2023. 1039 p. Acesso em 27 Set. 2024. Disponível em: <https://aisel.aisnet.org/amcis2023/lacais/lacais/1/?utm_source=aisel.aisnet.org%2Famcis2023%2Flacais%2Flacais%2F1&utm_medium=PDF&utm_campaign=PDFCoverPages>.

BRUE, M. **Low-Code, No-Code Aligns Business Needs With IT Strategy For Greater Workforce Productivity**. Forbes, 2023. Disponível em: <https://www.forbes.com/sites/moorinsights/2023/01/17/low-code-no-code-aligns-business-needs-with-it\-strategy-for-greater-workforce-productivity/?sh=8ae45001b320>.

DENDENA, A.; BISOGNIN, G. **Utilização dos Conceitos de Low Code e No Code na geração de web services com arquitetura mda**. UNESC - Universidade do Extremo Sul Catarinense, 2021. Disponível em: <http://repositorio.unesc.net/handle/1/9136>.

DESILVA, D.; RANATHUNGA, R.; SHANGAVIE, R. **Quality Assurance in Low-Code/No-Code Development**. [S.l.: s.n.], 2023. 1-5 p.

DHOKE, P.; LOKULWAR, P. **Evaluating the Impact of No-Code/Low-Code Backend Services on API Development and Implementation: A Case Study Approach**. [S.l.: s.n.], 2023. 1-5 p.

KAMOUCHI, H. E.; KISSI, M.; BEGGAR, O. E. **Low-code/No-code Development : A systematic literature review**. [S.l.: s.n.], 2023. 1-8 p.

MATOS, A. A. R. **Desenvolvimento de aplicações web em plataforma de low-code**. [s.n.], 2022. Acesso em 27 Set. 2024. Disponível em: <http://hdl.handle.net/10198/26419>.

PICEK, R. **Low-code/No-code Platforms and Modern ERP Systems**. [S.l.: s.n.], 2023. 44-49 p.

SANTOS, R. F.; AGANETTE, E. **Plataformas de desenvolvimento low-code na transformação digital e no gerenciamento de processos de negócios**. 2023. Acesso em 21 Ago. 2024. Disponível em: <https://forped.eci.ufmg.br/revista/forped/article/view/100>.

SOUZA, J. L. d. **A Contribuição do Low Code no Âmbito Educacional: Um Mapeamento Sistemático da Literatura**. 2022. Acesso em 21 Ago. 2024. Disponível em: <https://www.cin.ufpe.br/~tg/2021-2/tg_SI/TG_jls3.pdf>.

STIVAL, H. P.; CARLOS, L.; ALMEIDA, T. M. A. **O processo da evolução tecnológica**. 2009. Acesso em 30 Ago. 2024. Disponível em: <https://semanaacademica.org.br/system/files/artigos/artigo_5_0.pdf>.