USO DE FLUTTER, NODE.JS E CLEAN ARCHITECTURE NO DESENVOLVIMENTO DE UM SOFTWARE MOBILE PARA MONITORIA ACADÊMICA

Souza, W. O.[1], Jukemura, A. S.[1]

[1] Pontifícia Universidade Católica de Goiás, Goiânia, GO, Brasil

RESUMO: O objetivo desse trabalho é descrever as vantagens do uso das tecnologias modernas, como *Flutter* para a interface de usuário, *Node.js* para gestão do servidor e *Clean Architecture* para estruturação de um *software mobile.* O projeto, desenvolvido dentro de um contexto de pesquisa exploratória, destaca-se pelo uso de ferramentas e práticas avançadas da engenharia de *software*. Essas tecnologias não são apenas instrumentos para alcançar um objetivo de criar uma aplicação para monitoria acadêmica de disciplinas, mas também simbolizam um avanço significativo na maneira como as soluções digitais podem ser desenvolvidas. Os resultados permitiram concluir que, enquanto o projeto avançou em seu estágio exploratório, ele se estabeleceu como uma prova do potencial transformador das tecnologias de ponta e das práticas avançadas de engenharia de *software*. Observou-se que o compromisso com a utilização de *Node.js*, *Flutter* e *Clean Architecture* vai além do desenvolvimento de uma única aplicação, sinalizando uma mudança paradigmática na criação de soluções digitais que são tecnicamente robustas, esteticamente agradáveis e altamente funcionais. As futuras expansões tecnológicas prometem não só enriquecer o campo da educação digital, mas também oferecer *insights* valiosos para inovações em diversos setores, reforçando o impacto abrangente dessas ferramentas no mundo da tecnologia.

Palavras chaves: *Flutter*, Arquitetura Limpa, Desenvolvimento de Aplicações Móveis.

ABSTRACT: *The objective of this work is to describe the advantages of using modern technologies such as Flutter for the user interface, Node.js for server management, and Clean Architecture for structuring a mobile software. The project, developed within the context of exploratory research, stands out for its use of advanced software engineering tools and practices. These technologies are not only instruments to achieve the goal of creating an application for academic monitoring of subjects but also represent a significant advance in how digital solutions can be developed. The results allowed the conclusion that, while the project advanced in its exploratory stage, it established itself as proof of the transformative potential of cutting-edge technologies and advanced software engineering practices. It was observed that the commitment to using Node.js, Flutter, and Clean Architecture goes beyond the development of a single application, signaling a paradigm shift in the creation of digital solutions that are technically robust, aesthetically pleasing, and highly functional. Future technological expansions promise not only to enrich the field of digital education but also to offer valuable insights for*

innovations in various sectors, reinforcing the comprehensive impact of these tools on the world of technology.

Keywords: *Flutter, Clean Architecture, Mobile Application Development.*

1. Introdução

A capacidade de desenvolver sistemas robustos e escaláveis se tornou uma prerrogativa no campo da tecnologia, demandando metodologias e ferramentas que facilitem tais empreendimentos. A *Clean Architecture*, proposta por Martin [1], emerge como uma solução promissora, estruturando o software em camadas isoladas para promover flexibilidade e facilidade de manutenção. Esta abordagem destaca-se pela sua modularidade, permitindo atualizações e melhorias contínuas com reduzido impacto nas demais partes do sistema [2].

Acompanhando a *Clean Architecture* na vanguarda do desenvolvimento de *software*, o *Flutter*, introduzido por Dagne [3], oferece uma plataforma agnóstica para o desenvolvimento de interfaces de usuário, permitindo a criação de aplicações *cross-platform* eficientes. Este *framework*, baseado na linguagem *Dart* [4], simplifica o desenvolvimento de aplicações *mobile* para *iOS* e *Android*, garantindo consistência visual e funcional entre plataformas.

Paralelamente, conforme informado por Bessa [5] o *Node.js* representa uma escolha estratégica para o desenvolvimento do servidor da aplicação, graças à sua eficiência em gerenciar múltiplas conexões simultâneas um recurso indispensável para suportar funcionalidades em tempo real. A documentação de *APIs REST*, facilitada por uma ferramenta de apoio, a ferramenta *Swagger* [6], juntamente com o *Express* [7], *framework* do *Node.js*, que fornece um conjunto de recursos para aplicativos *web* e *mobile*, são elementos que complementam a robustez e a flexibilidade do servidor, garantindo uma comunicação ágil e segura entre cliente e servidor.

Considerando a integração dessas tecnologias avançadas de desenvolvimento de *software*, este trabalho explora como *Flutter*, *Node.js* e *Clean Architecture* podem facilitar a criação de aplicações robustas, escaláveis e modulares. Em particular, apresenta uma solução para o desenvolvimento de um *software mobile* destinado à monitoria

acadêmica de disciplinas, visando otimizar a pesquisa por monitores, facilitar a interação entre o aluno e o monitor e proporcionar flexibilidade nos horários das sessões.

O objetivo central é destacar as vantagens do uso dessas tecnologias modernas, utilizando *Flutter* para a interface de usuário, *Node.js* para a gestão do servidor e *Clean Architecture* para a estruturação do *software*. A questão de pesquisa que orienta este estudo é como a combinação de *Flutter, Node.js e Clean Architecture* pode contribuir para o desenvolvimento de *software* que seja robusto, escalável e modular.

A utilização do *Node.js*, em conjunto com a *Clean Architecture* e o *Flutter*, não apenas viabiliza a construção de um aplicativo *mobile* eficiente e de alta usabilidade, mas também abre caminho para novas possibilidades de interação e aprendizagem [2]. No âmbito acadêmico, a prática da monitoria desempenha um papel crucial, enriquecendo o processo educacional ao promover uma interação direta e significativa entre alunos e monitores [8; 9].

Esta integração de metodologias de desenvolvimento com a prática educativa sublinha a potencialidade de criar ambientes de aprendizagem mais acessíveis e engajadores, enfatizando a importância de aliar inovação tecnológica à tradição educacional para enfrentar os desafios contemporâneos do ensino e aprendizagem [10].

Considerando o cenário supracitado, o principal objetivo deste trabalho é descrever a sinergia entre a *Clean Architecture* e o *Dart/Flutter*, aprofundando-se nas potencialidades desta combinação no contexto atual de desenvolvimento de software.

Como um estudo de caso prático desta integração, foi desenvolvido um aplicativo de monitoria de disciplinas acadêmicas. Dentre os objetivos específicos, o aplicativo permite o envio de mensagens ao monitor, além de permitir a busca por monitores com base na disciplina de interesse.

2. Referencial teórico

A evolução contínua das metodologias de desenvolvimento de *software* tem sido direcionada para soluções que promovem não apenas a eficiência e a qualidade, mas também a flexibilidade e a sustentabilidade a longo prazo dos projetos. Neste panorama, a *Clean Architecture*, apresentada por Robert C. Martin [1], surge como um marco conceitual, inspirando desenvolvedores a repensarem a estruturação de suas aplicações.

Essa abordagem é fundamentada nos princípios das arquiteturas Hexagonal e *Onion*, visando uma divisão clara das responsabilidades dentro do *software*, que se traduz em camadas dedicadas às regras de negócios e às interfaces de usuário.

A *Clean Architecture* destaca-se por promover a independência de *frameworks*, facilitar a testabilidade, e permitir flexibilidade nas mudanças de interface de usuário e banco de dados, garantindo que as aplicações sejam escaláveis, fáceis de manter e eficientes. Martin [1] argumenta que uma arquitetura bem definida é crucial para a longevidade e sucesso do *software*, estabelecendo um alicerce sólido que suporta o crescimento e adaptação às mudanças tecnológicas e de mercado.

Em consonância com os princípios de design modular e independente promovidos pela *Clean Architecture*, o *framework Flutter* (uma inovação do Google) revoluciona o desenvolvimento de aplicações multiplataforma. O *framework Flutter* permite aos desenvolvedores também construírem aplicações com interfaces de usuário ricas e responsivas para *Android*, *iOS*, *MS-Windows*, *macOS*, *GNU/Linux* e *web*, a partir de um único código-base [11].

A principal vantagem do *Flutter* reside na sua capacidade de acelerar o desenvolvimento através do *hot reload*, permitindo mudanças em tempo real no código sem necessidade de reinício da aplicação que eleva significativamente a produtividade e eficiência do desenvolvimento. Além disso, *Dart* foi projetada para otimizar a experiência do desenvolvedor, incorporando funcionalidades modernas como *null safety* e um robusto sistema de *widgets*, facilitando a criação de interfaces complexas e interativas [11].

A integração de *Flutter* e *Dart*, conforme explorado na monografia de Ferrero [12], ilustra uma abordagem inovadora no desenvolvimento de aplicações multiplataforma. Alinhada com os princípios de modularidade e separação de preocupações fundamentais para uma arquitetura limpa, esta combinação não só adere aos padrões arquitetônicos recomendados, mas também redefine as expectativas de eficiência e versatilidade no desenvolvimento de *software* moderno.

Em complemento, a comparação entre o *Flutter* e outros *frameworks* de desenvolvimento multiplataforma, destacada por Sattar et al. [13], posiciona o *Flutter* como uma escolha preferencial para desenvolvedores em busca de eficiência, rapidez e qualidade. Distinto de outras plataformas, como *React Native* e *Xamarin*, o *Flutter* se

sobressai pela sua capacidade de compilar para código nativo em múltiplas plataformas a partir de uma base de código unificada. Essa capacidade única confere ao *Flutter* vantagens notáveis em termos de desempenho e fluidez das aplicações desenvolvidas, além de permitir um design de interface de usuário mais coeso e atraente em todas as plataformas.

Avançando na discussão sobre eficiência de uso de recursos, a análise realizada por Markowski e Smołka [14] sugere que, embora não se possa declarar um vencedor definitivo em termos de eficiência de recursos, o *Flutter* mantém um consumo comparável ao do *React Native*. Este equilíbrio destaca a eficácia do *Flutter* em harmonizar agilidade no desenvolvimento com otimização de recursos, essencial para o desenvolvimento de aplicativos que não apenas oferecem performance fluida em diversas plataformas, mas também são eficientes em seu consumo de recursos.

Além disso, a capacidade do *Flutter* de entregar interfaces de usuário ricas e interativas, sem um comprometimento significativo do consumo de memória e *CPU*, reforça sua posição como uma ferramenta inovadora no cenário atual de desenvolvimento de aplicativos móveis. A habilidade de equilibrar performance, eficiência no uso de recursos e excelência visual faz do *Flutter* a escolha ideal para desenvolvedores que priorizam tanto a eficiência quanto a qualidade. Portanto, através das análises de Ferrero [12], Sattar et al. [13], e Markowski e Smołka [14], fica evidente a superioridade do *Flutter* como uma solução abrangente e preferencial para o desenvolvimento de aplicativos multiplataforma.

No que tange ao desenvolvimento do lado do servidor, segundo Daylay [15] o *Node.js* representa uma transformação significativa na aplicação do *JavaScript*, estendendo seu domínio do *front-end* para abarcar também soluções de *back-end*. Com sua arquitetura orientada a eventos e assíncrona, baseada no motor V8 do Google Chrome, o *Node.js* introduz um paradigma eficiente para o desenvolvimento de aplicações *web* escaláveis, incluindo *APIs REST* e sistemas em tempo real. Esta plataforma se beneficia de um ecossistema rico e diversificado, proporcionado pela comunidade de código aberto e pelo *Node Package Manager (NPM)*, que oferece uma ampla gama de pacotes que facilitam a implementação de funcionalidades complexas e a integração com diversas tecnologias [16].

Nesse contexto, o *Swagger* emerge como uma ferramenta essencial, otimizando o desenvolvimento e a documentação de *APIs REST*. Ao fornecer uma representação detalhada das *APIs*, incluindo operações, parâmetros e respostas esperadas, ele promove uma compreensão clara e um fácil acesso às funcionalidades da *API*, além de permitir testes diretos na interface. Esta integração do *Swagger* com o *Node.js* potencializa a criação de soluções *back-end* robustas, enfatizando a importância da documentação precisa e interativa no ciclo de desenvolvimento de *software* [17].

A sinergia entre *Node.js* e *frameworks* como *Express.js* potencializa ainda mais a capacidade de desenvolver aplicações robustas, eficientes e flexíveis, alinhadas aos princípios de modularidade, independência e escalabilidade destacados pela *Clean Architecture* e pelas capacidades multiplataforma do *Flutter* [1; 15; 16].

Incorporando a flexibilidade da *Clean Architecture* na escolha e integração de bancos de dados ao projeto, tem-se a liberdade de selecionar a tecnologia de armazenamento que melhor se adapte às necessidades específicas da aplicação. A *Clean Architecture*, conforme descrita por Martin [1], permite uma integração harmoniosa de qualquer sistema de gerenciamento de banco de dados, graças à sua abordagem de separação de dependências.

O banco de dados *SQLite* foi escolhido para o armazenamento local em dispositivos móveis. Sua portabilidade e simplicidade, aliadas à capacidade de funcionamento autônomo sem a necessidade de um servidor, o tornam uma opção primordial para aplicações móveis, garantindo um armazenamento eficiente e prático [18].

Paralelamente, o *PostgreSQL* foi selecionado para o armazenamento centralizado de dados, valorizando sua robustez, confiabilidade e aderência aos padrões *SQL*. Essas qualidades o posicionam como a solução ideal para administrar a complexidade e o volume dos dados gerenciados pelo aplicativo, demonstrando a capacidade do projeto em alinhar-se com tecnologias de armazenamento de dados altamente escaláveis e eficazes [19].

Essa dualidade de estratégias que permite a seleção de bancos de dados, sob a orientação da *Clean Architecture*, evidencia um compromisso com a escolha de soluções tecnológicas adequadas, assegurando que o aplicativo não apenas satisfaça as demandas

atuais de maneira eficiente, mas esteja igualmente preparado para futuras expansões e inovações tecnológicas.

3. Materiais e Métodos

O desenvolvimento desta aplicação é uma resposta direta à crescente necessidade de melhorar a interação entre alunos e monitores de disciplinas acadêmicas, visando otimizar o processo de aprendizado. Em um esforço para criar um canal de comunicação eficiente e direto, a aplicação desenvolvida facilita o acesso dos estudantes a suporte de monitores qualificados, abordando suas demandas específicas pela busca de reforço nas suas disciplinas.

A aplicação foi construída sobre uma fundação de práticas de desenvolvimento de *software* de ponta, adotando a *Clean Architecture* para assegurar uma arquitetura de aplicação robusta e flexível. Esta abordagem arquitetônica nos permite manter o foco nas regras de negócio, enquanto mantém a infraestrutura, como interface, banco de dados e *frameworks*, em camadas separadas e facilmente intercambiáveis. Isso não apenas facilita a manutenção e o teste do aplicativo, mas também o prepara para adaptações e melhorias futuras sem emaranhados de dependências complexas [1].

Implementando a interface do aplicativo, o *Flutter* apresentou-se como solução ideal devido à sua excepcional capacidade de compilar para plataformas móveis nativas, *web* e *desktop* a partir de um único código-fonte [3]. Esta escolha estratégica garante uma experiência de usuário unificada e de alto desempenho em todos os dispositivos, beneficiando-se do rápido ciclo de desenvolvimento que o *hot reload* do *Flutter* proporciona. A linguagem *Dart*, com sua sintaxe moderna e recursos avançados, permite construir funcionalidades complexas de maneira eficiente, enquanto mantém o código legível e conciso [11].

O *Flutter* introduz uma metodologia centrada em *widgets*, destacando-se dos demais *frameworks* por sua abordagem inovadora na construção de interfaces de usuário. Esta estratégia, detalhada por Sattar et al. [13], enfatiza a eficácia do *Flutter* em promover um desenvolvimento ágil e eficiente, permitindo a rápida composição de interfaces complexas e interativas. Analogamente aos blocos de construção, os *widgets* do *Flutter* servem como elementos fundamentais que, ao serem combinados, facilitam a criação de aplicações robustas e altamente funcionais. Esse método não apenas acelera

significativamente o processo de desenvolvimento, mas também assegura um desempenho superior em várias plataformas, estabelecendo o *Flutter* como uma ferramenta poderosa que redefine a eficiência e a flexibilidade no desenvolvimento de *software*.

Para o *back-end*, o *Node.js* entrega uma arquitetura baseada em eventos e ao modelo de *I/O* não bloqueante [16]. Isso é particularmente vantajoso para o sistema de *chat* em tempo real, cuja capacidade de lidar com múltiplas conexões simultaneamente, sem prejudicar o desempenho, é crucial. A escolha do *Node.js* reflete o compromisso com a construção de um *back-end* escalável e eficiente, capaz de suportar as interações dinâmicas exigidas pelos usuários do aplicativo.

4. Resultados e Discussão

A Figura 1 ilustra a arquitetura única da árvore de *widgets* do *Flutter*, destacando a organização e a função específica de cada *widget* dentro do contexto amplo da aplicação. A escolha do *Flutter*, respaldada pelo estudo de Thanh Tran [20], não apenas reforça a eficiência na construção de interfaces expressivas e dinâmicas, mas também sublinha o alto desempenho nativo oferecido pela tecnologia. Essa abordagem orientada a *widgets* simplifica profundamente o processo de desenvolvimento do *front-end*, permitindo uma criação rápida e intuitiva das telas do aplicativo. Graças a essas características inerentes ao *Flutter*, como detalhado por Tran [20], foi viável desenvolver uma interface detalhada e altamente responsiva em um período significativamente reduzido.

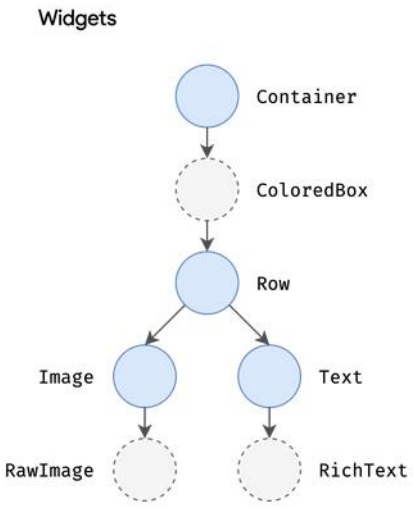

Figura *1:* Árvore de *Widgets*. [11]

A experiência dos usuários pelo aplicativo tem início com a Tela de *Login*, ilustrada na Figura 2, projetada para ser não apenas visualmente atraente, mas também um bastião de segurança e privacidade. Utilizando as capacidades avançadas do *Flutter*, destacadas no estudo de Thanh Tran [20], essa tela foi concebida para ser mais do que o primeiro ponto de interação com o aplicativo, é uma porta de entrada segura e confiável para o processo de autenticação. Através do uso dos *widgets* do *Flutter*, foi possível desenvolver uma interface que não só captura a atenção dos usuários desde o primeiro momento, mas também sublinha o compromisso inabalável com a proteção de seus dados.

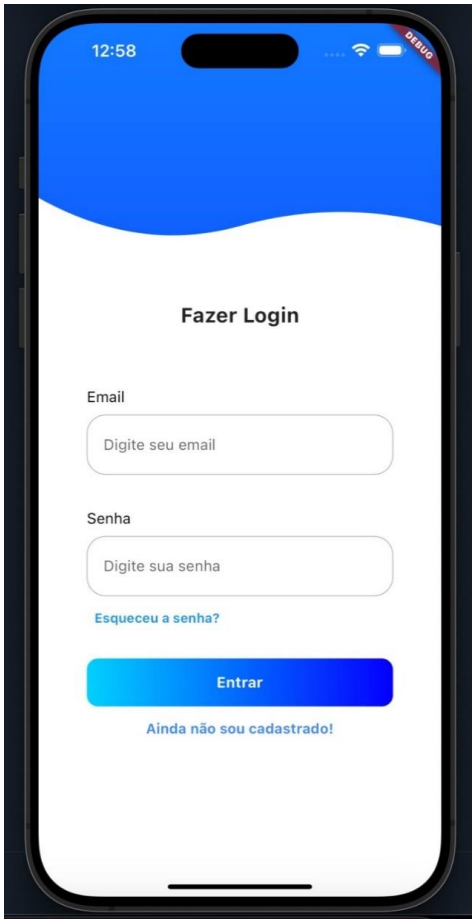

Figura 2: Tela de *Login*.

Avançando para a Tela Principal ilustrada na Figura 3, os usuários são recebidos com uma saudação personalizada, criando uma atmosfera acolhedora que valoriza a experiência individual de cada usuário. Essa tela centraliza o acesso às funcionalidades principais do aplicativo, oferecendo uma navegação fluida e intuitiva.

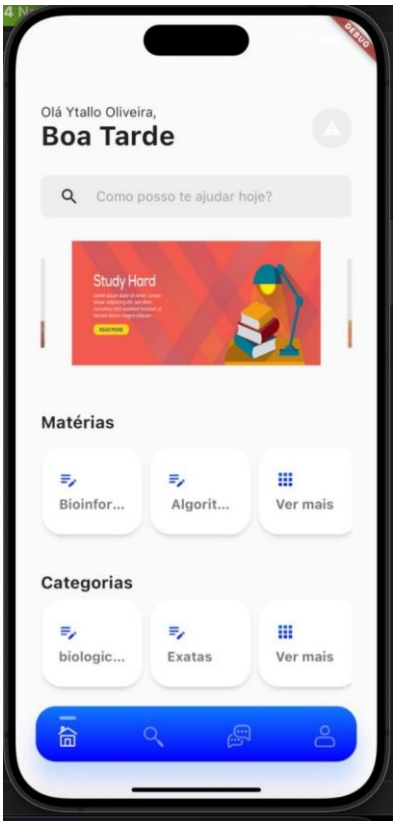

Figura 3: Tela principal.

A Tela de Busca, como ilustrado na Figura 4, demonstra a capacidade do aplicativo de proporcionar uma pesquisa eficiente e personalizada para os usuários, permitindo a localização de monitores por disciplina, nome ou avaliação. Essa funcionalidade é crucial para facilitar a conexão dos alunos com os recursos acadêmicos de maneira rápida e precisa.

A sofisticada capacidade de pesquisa avançada do aplicativo possibilita aos usuários encontrarem monitores através de requisições SQL elaboradas, processadas no servidor Node.js. Essa abordagem não só garante uma conexão direta e veloz entre os alunos e os recursos acadêmicos mais adequados, mas também otimiza a eficiência e precisão do processo de busca. Isso evidencia o potencial da integração dessas tecnologias no desenvolvimento de aplicações móveis.

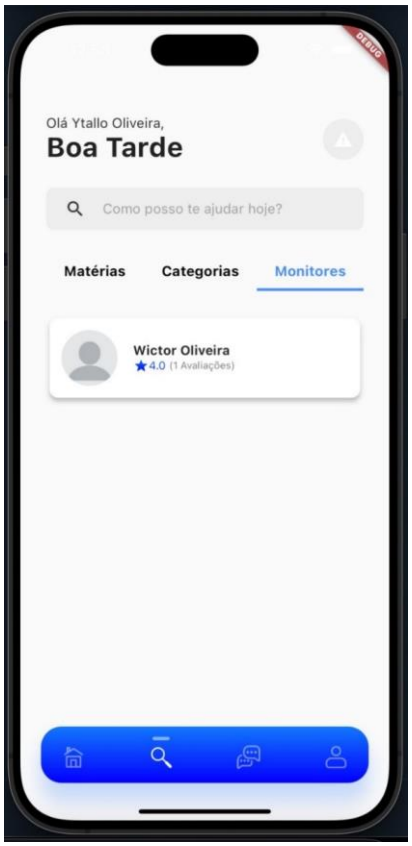

Figura 4: Tela de busca por monitores.

Para assegurar uma interação eficiente e personalizada, o monitor possui um perfil separado do usuário comum. Este perfil, ilustrado na Figura 5, permite que ele seja encontrado na página de busca pelos alunos que necessitam de monitoria. O monitor pode especificar as matérias que leciona, definir dias e horários de disponibilidade, e adicionar uma breve descrição sobre sua experiência e abordagem de ensino. Esta estrutura não só facilita a organização e a gestão das atividades de monitoria, mas também proporciona aos alunos uma visão clara e detalhada sobre cada monitor disponível, aprimorando a qualidade do suporte acadêmico oferecido.

Figura 5: Tela de perfil do monitor.

A interface de *Chat*, apresentada na Figura 6, posiciona-se como um ponto central das interações dentro do aplicativo, estabelecendo um espaço para comunicação em tempo real entre estudantes e monitores (tutores). Este recurso não apenas facilita a troca de informações, mas também reforça o compromisso do aplicativo em fornecer suporte acadêmico acessível e eficiente.

Figura 6: Tela da sala de conversa.

Conforme descrito por *SQLite* [18], ao incorporar um banco de dados que elimina a necessidade de um servidor dedicado, o banco de dados *SQLite* se destaca por sua natureza compacta, contribuindo significativamente para a redução do tamanho do aplicativo. A eficácia desse banco de dados é reconhecida e utilizada por empresas de renome como *Apple*, *Google* e *Adobe*.

5. Conclusão

O projeto, desenvolvido dentro de um contexto de pesquisa exploratória, destaca-se pelo uso inovador de ferramentas e práticas avançadas de engenharia de *software*, como *Clean Architecture*, *Flutter*, *Dart*, e *Node.js*. Estas tecnologias não são apenas meios para o fim de criar uma aplicação de monitoria acadêmica, elas representam um avanço significativo na forma como as soluções digitais podem ser desenvolvidas, prometendo transformar o panorama da engenharia de *software* através da modularidade, escalabilidade, e manutenção eficiente.

Ao colocar estas ferramentas no centro do processo de desenvolvimento, o projeto não apenas buscou solucionar desafios específicos da educação digital, mas também ilustrou como a integração de práticas e tecnologias modernas pode ser fundamental na criação de soluções sofisticadas que atendem a uma ampla gama de necessidades. A capacidade do *Flutter* de oferecer interfaces de usuário interativas e elegantes, juntamente com a robustez e escalabilidade garantidas pelo *Node.js* e a organização proporcionada pela *Clean Architecture*, exemplifica um modelo que pode ser replicado e adaptado para outros domínios além da educação.

Para futuras expansões, uma direção promissora envolve o uso de inteligência artificial no aplicativo. Essa tecnologia pode ser utilizada para desenvolver um sistema de recomendação avançado, sugerindo monitores específicos para os alunos com base em suas necessidades individuais e histórico de interações. Além disso, a inteligência artificial pode melhorar a análise de desempenho dos monitores, fornecendo *feedback* automatizado e identificando áreas de melhoria. Essa aplicação tem o potencial de personalizar ainda mais a experiência do usuário, tornando o processo de monitoria mais eficiente e eficaz.

Este foco em ferramentas e metodologias avançadas coloca o projeto em uma posição de destaque, não só como um desenvolvimento promissor no campo da monitoria acadêmica, mas como um caso de estudo valioso para a engenharia de *software*. A exploração dessas tecnologias em contextos variados e para diferentes tipos de aplicações não apenas comprovará sua versatilidade, mas também poderá inspirar abordagens inovadoras no design e desenvolvimento de *software*, abrindo novas perspectivas para a aplicação de inteligência artificial e personalização da experiência do usuário.

O estudo realizado permitiu concluir que, enquanto o projeto avançou em seu estágio exploratório, ele se estabeleceu como uma prova do potencial transformador das tecnologias de ponta e das práticas avançadas de engenharia de software. Observou-se que o compromisso com a utilização de *Node.js*, *Flutter* e *Clean Architecture* vai além do desenvolvimento de uma única aplicação, sinalizando uma mudança paradigmática na criação de soluções digitais que são tecnicamente robustas, esteticamente agradáveis, e altamente funcionais. As futuras expansões tecnológicas prometem não só enriquecer o campo da educação digital, mas também oferecer *insights* valiosos para inovações em

diversos setores, reforçando o impacto abrangente dessas ferramentas no mundo da tecnologia.

6. Referências Bibliográficas

1. Martin, R.C.: Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, Filadélfia, PA, USA, (2017).
2. Bueno, C.E. de O.: Desenvolvimento de um aplicativo utilizando o framework Flutter e arquitetura limpa. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) - Pontifícia Universidade Católica de Goiás, Goiânia - GO, (2021). Disponível em: https://repositorio.pucgoias.edu.br/jspui/handle/123456789/1861. Acesso em: 6 set. 2023.
3. Dagne, L.: Flutter for Cross-platform App and SDK Development. Bachelor's Thesis, Metropolia University of Applied Sciences, Helsinki (2019). Disponível em: https://www.theseus.fi/bitstream/handle/10024/172866/Lukas%20Dagne%20Thesis.pdf.
4. Dart: Guia de Dart. Acesso em 14 mar. 2024. Disponível em: https://dart.dev/guides.
5. Bessa, A.: O que é Node.JS? Como funciona e um Guia para iniciar. Alura. Acesso em 23 out. 2023. Disponível em: https://www.alura.com.br/artigos/node-js.
6. Swagger: Swagger Framework for APIs Website. (2016). Disponível em: http://www.swagger.io/. Acesso em: 23 out. 2023.
7. Express.js: Express: Framework de aplicativo web rápido, flexível e minimalista para Node.js. Acesso em 23 out. 2023. Disponível em: https://expressjs.com/pt-br/.
8. Bastos, M.H.C., Faria Filho, L.M. (Orgs.): A Escola Elementar no Século XIX: O Método Monitorial/Mútuo. Ediupf, Passo Fundo (1999).
9. Matoso, L.M.L.: A importância da monitoria na formação acadêmica do monitor: um relato de experiência. Rev. Científica da Escola da Saúde, Acesso em 19 mar. 2023; n. 2, Abr./Set., (2014). Disponível em: https://repositorio.unp.br/index.php/catussaba/article/view/567.
10. Silva, C.P.B. da: A Escola Elementar no Século XIX. O Método Monitorial/Mútuo. Rev. Bras. Hist. Educ.; 1(1), 211-213; (2012).
11. Flutter: Flutter Documentation. Acesso em 08 out. 2023. Disponível em: https://docs.flutter.dev/ (2018).

12. Ferrero Ligorred, A.: Development of a Large-Scale Flutter App. Politecnico di Milano, School of Industrial and Information Engineering, Master's Thesis (2022).
13. Sattar, A., Soni, P., Kr, M., Ranjan, M., Kumar, A., Sahu, C., Saxena, S., Chaudhari, P.: Open Source Developments Accelerating Cross-platform Development with Flutter Framework. JOURNAL OF OPEN SOURCE DEVELOPMENTS, 10, 1-11 (2023). DOI: 10.37591/joosd.v10i2.580.
14. Markowski, M., Smołka, J.: A comparative analysis of the Flutter and React Native frameworks. Journal of Computer Sciences Institute, 29, 346–351, (2023). https://doi.org/10.35784/jcsi.3794
15. Dayley, B.: Node.js, MongoDB, and AngularJS Web Development. Addison-Wesley Professional, Michigan, EUA, (2014).
16. Herron, D.: Node.js Web Development. 5th ed. Pact Publishing Ltd, (2020).
17. Surwase, V.B.: REST API Modeling Languages - A Developer's Perspective. International Journal For Science Technology And Engineering, 2, 634-637, (2016). Disponível em: https://api.semanticscholar.org/CorpusID:54773316. Acesso em: 23 out. 2023.
18. SQLite: Documentação do SQLite. Acesso em: 13 de março de 2024. Disponível em: https://www.sqlite.org/docs.html.
19. PostgreSQL: PostgreSQL: O banco de dados open source mais avançado do mundo. Acesso em 02 de março de 2024. Disponível em: https://www.postgresql.org/.
20. Tran, T.: Flutter Native Performance and Expressive UI/UX. Thesis, (2020). https://www.theseus.fi/handle/10024/336980

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante WICTOR OLIVEIRA DE SOUZA
do Curso de Ciência da Computação, matrícula 2019.1.00280038-0, telefone: (62) 98261-4636 e-mail wictor@dablinlabs.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado USO DE FLUTTER, NODE.JS E CLEAN ARCHITECTURE NO DESENVOLVIMENTO DE UM SOFTWARE MOBILE PARA MONITORIA ACADÊMICA, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 24 de Junho de 2024.

Assinatura do autor: Wictor Oliveira de Souza

Nome completo do autor: WICTOR OLIVEIRA DE SOUZA

Assinatura do professor–orientador: _____

Nome completo do professor-orientador: ANIBAL SANTOS JUKEMURA