UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ
INSTITUTO DE GEOCIÊNCIAS E ENGENHARIAS
Faculdade de Engenharia da Computação
Bacharelado em Engenharia da Computação

Projeto Final de Curso II

DESENVOLVIMENTO DE UM PROTÓTIPO DE SOFTWARE PARA GESTÃO DE TURMAS E ACOMPANHAMENTO ACADÊMICO

Isac Galvão Santos

Marabá - PA
2025

Isac Galvão Santos

DESENVOLVIMENTO DE UM PROTÓTIPO DE SOFTWARE PARA GESTÃO DE TURMAS E ACOMPANHAMENTO ACADÊMICO

Projeto Final de Curso II, apresentado à Universidade Federal do Sul e Sudeste do Pará, como parte dos requisitos necessários para obtenção do Título de Bacharel em Engenharia da Computação.

Orientador:
Prof. Dr. João Victor Costa Carmona

Marabá - PA
2025

Isac Galvão Santos

DESENVOLVIMENTO DE UM PROTÓTIPO DE SOFTWARE PARA GESTÃO DE TURMAS E ACOMPANHAMENTO ACADÊMICO

Projeto Final de Curso II, apresentado à Universidade Federal do Sul e Sudeste do Pará, como parte dos requisitos necessários para obtenção do Título de Bacharel em Engenharia da Computação.

Marabá: 03 de Fevereiro de 2025

BANCA QUALIFICADORA:

Prof. Dr. João Victor Costa Carmona
(Orientador - Unifesspa)

Profa. Dra. Marcela Alves de Souza
(Membro da Banca - Unifesspa)

Prof. Dr. Jose Carlos da Silva
(Membro da Banca - Unifesspa)

Marabá - PA
2025

AGRADECIMENTOS

Primeiramente, expresso minha mais profunda gratidão à minha família: minha mãe, Elane Pontes Galvão Santos; meu pai, Raimundo de Sousa Santos; e minha irmã, Isabela Galvão Santos. Desde o início da minha jornada acadêmica, vocês foram minha base e meu alicerce. Sem o amor, o apoio e os sacrifícios de vocês, eu não teria conseguido chegar até aqui.

Agradeço, também, ao meu tio, Alan Pontes Galvão, que teve um papel fundamental nos momentos mais difíceis do início da minha graduação. Sua ajuda e seu incentivo foram essenciais para que eu pudesse seguir em frente, e sou imensamente grato por sua presença e apoio incondicional.

Ao meu orientador, João Victor Costa Carmona, agradeço pela paciência, pela dedicação e pelo compromisso em me guiar durante a elaboração deste trabalho. Seu esforço e suas orientações foram fundamentais para que eu pudesse concluir esta etapa tão importante da minha vida.

Não poderia deixar de mencionar meus amigos da faculdade, que tornaram essa caminhada menos árdua e mais significativa. Com vocês, aprendi muito além do conteúdo das aulas; aprendi a enfrentar os desafios da distância de casa, a superar dificuldades e a valorizar cada aprendizado que a vida universitária trouxe. Em especial, meu sincero agradecimento a Naiara Taiane, Samuel Patrick, Thiago Eleuterio, Sidcley Souza, Mateus Luz e Henrique Viana.

Ao longo dessa jornada acadêmica, enfrentei desafios que foram muito além das salas de aula e dos livros. Foram anos de aprendizado, mas também de provações, onde precisei lidar com dificuldades que me afetaram profundamente, testando não apenas minha capacidade intelectual, mas também minha resiliência emocional. Houve momentos de incerteza, de exaustão e de solidão, nos quais duvidei se conseguiria seguir em frente.

Sendo assim, agradeço a mim mesmo, pois, em meio a tudo isso, encontrei forças para continuar. Esta conquista não representa apenas um diploma, mas a superação de inúmeras batalhas internas e externas que tornaram esse caminho desafiador e, ao mesmo tempo, transformador.

A todos que, de alguma forma, contribuíram para a realização deste sonho, meu mais sincero muito obrigado.

"No meio do inverno, aprendi enfim que havia em mim um verão invencível."
— Albert Camus

RESUMO

Com o avanço das tecnologias móveis e a crescente demanda por soluções que otimizem a rotina de professores, este trabalho apresenta o desenvolvimento de um protótipo de aplicativo voltado para a organização e gerenciamento de turmas e acompanhamento acadêmico. A ideia do aplicativo visa atender tanto professores independentes quanto aqueles que atuam em instituições de ensino, oferecendo uma ferramenta prática para o cadastro e gerenciamento de turmas e alunos, bem como para o lançamento de frequências e notas. Uma das principais inovações sugeridas é a disponibilização de uma área exclusiva para que alunos ou seus responsáveis possam acompanhar seu histórico acadêmico. Por meio dessa interface, seria possível consultar notas, verificar a frequência e acessar indicadores de desempenho. Além disso, a proposta inclui um mecanismo de registro de presença, no qual o professor poderia gerar um QR Code para que o aluno marcasse sua participação na aula. O projeto conceitual baseia-se no uso de tecnologias modernas e de código aberto, proporcionando uma base flexível e de fácil manutenção para futuras evoluções e integrações. Ademais, a proposta sugere que o aplicativo seja um projeto *open-source*, permitindo que desenvolvedores e interessados contribuam com melhorias, personalizações e adaptações, tornando-o mais aberto à colaboração e ao aperfeiçoamento contínuo. Essa abordagem *open-source* busca fomentar a transparência, a cooperação e a democratização do acesso à educação, ampliando seu impacto de forma sustentável e inclusiva.

Palavras-chave: Protótipo de Aplicativo. Gestão de Turmas. Educação. Open source. Transparência.

ABSTRACT

With the advancement of mobile technologies and the growing demand for solutions that optimize teachers' routines, this work presents the development of a prototype application aimed at organizing and managing classes and academic monitoring. The idea of the application is to serve both independent teachers and those working in educational institutions, offering a practical tool for registering and managing classes and students, as well as for recording attendance and grades. One of the main innovations suggested is the provision of an exclusive area for students or their guardians to track their academic history. Through this interface, it would be possible to check grades, verify attendance, and access performance indicators. Additionally, the proposal includes an attendance registration mechanism, in which the teacher could generate a QR Code for students to mark their participation in class. The conceptual project is based on the use of modern and open-source technologies, providing a flexible and easily maintainable foundation for future developments and integrations. Furthermore, the proposal suggests that the application be an *open-source* project, allowing developers and interested parties to contribute with improvements, customizations, and adaptations, making it more open to collaboration and continuous enhancement. This *open-source* approach aims to foster transparency, cooperation, and the democratization of access to education, expanding its impact in a sustainable and inclusive manner.

Keywords: Application Prototype. Class Management. Education. Open source. Transparency.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AOT	Ahead-of-Time
API	Application Programming Interface
COVID-19	Coronavirus Disease 2019
DIP	Dependency Inversion Principle
DRY	Don't Repeat Yourself
ISP	Interface Segregation Principle
JIT	Just-in-Time
JSON	JavaScript Object Notation
LMS	Learning Management System
LSP	Liskov Substitution Principle
MVC	Model-View-Controller
NoSQL	Not Only SQL
OCP	Open/Closed Principle
SIGAA	Sistema Integrado de Gestão de Atividades Acadêmicas
SOLID	Single Responsibility Principle; Open-Closed Principle; Liskov Substitution Principle; Interface Segregation Principle; Dependency Inversion Principle
SRP	Single Responsibility Principle
UI	User Interface
UX	User Experience

SUMÁRIO

1 INTRODUÇÃO

No contexto atual, a crescente demanda por soluções tecnológicas voltadas para a educação se torna evidente, especialmente após a pandemia de *Coronavirus Disease 2019* (COVID-19), que impulsionou o uso de ferramentas digitais em diversos segmentos, incluindo o ensino (AMANKWAH-AMOAH et al., 2021). Com isso, muitos professores, tanto independentes quanto vinculados a instituições de ensino, passaram a buscar alternativas para otimizar a organização de suas turmas e melhorar a comunicação de informações acadêmicas com seus alunos.

O presente trabalho tem como objetivo apresentar uma proposta de aplicativo móvel voltado para a gestão e organização de turmas, auxiliando professores no gerenciamento de alunos, lançamento de notas e frequência, além de permitir que os alunos e responsáveis acompanhem o desempenho acadêmico por meio de um acesso próprio. A proposta visa conceber um aplicativo que possa atuar como uma ferramenta de apoio, permitindo que professores tenham maior controle sobre suas turmas e que os alunos tenham mais transparência em relação ao seu progresso acadêmico.

Um dos principais benefícios do aplicativo é possibilitar o acompanhamento contínuo do desempenho acadêmico por parte dos alunos e seus responsáveis. Através de uma interface intuitiva, os usuários poderão visualizar notas, frequência e tendências de progresso ao longo do período letivo. Esse acompanhamento facilita a identificação de dificuldades, permitindo que medidas corretivas sejam tomadas com maior agilidade, seja por meio de reforço escolar ou ajustes na metodologia de ensino, promovendo, assim, um aprendizado mais eficiente.

A comunicação eficaz entre professores, alunos e responsáveis é essencial para um melhor desempenho acadêmico e organização escolar. O aplicativo busca facilitar esse processo por meio de notificações push, permitindo que professores enviem comunicados sobre prazos de trabalhos, datas de provas, eventos e outros avisos relevantes. Dessa forma, evita-se a dependência de recados verbais ou anotações informais, garantindo que as informações importantes cheguem diretamente aos alunos e responsáveis, promovendo maior transparência e engajamento.

Entre os diferenciais da proposta, destaca-se a possibilidade do aluno realizar o registro de presença através da leitura de um *QR Code* gerado pelo professor, além do acesso a indicadores acadêmicos, como notas mais altas e mais baixas ao longo do período letivo. A interface seria projetada para ser intuitiva e acessível, de forma a atender professores com diferentes níveis de familiaridade com tecnologia.

Ao longo deste trabalho, serão detalhados os principais aspectos da proposta do aplicativo, desde sua concepção até a estruturação de um protótipo funcional, servindo como base para possíveis desenvolvimentos futuros.

1.1 Justificativa

A proposição de um aplicativo móvel voltado para a gestão de turmas se mostra relevante diante da necessidade de professores independentes — ou seja, aqueles que ministram aulas particulares, cursos livres ou atuam de forma autônoma fora de grandes instituições de ensino — e docentes de escolas e universidades manterem um melhor controle sobre suas aulas e oferecerem maior transparência aos alunos e responsáveis. Essa transparência se justifica pelo fato de que, na maioria dos casos, os responsáveis têm acesso apenas às notas finais dos alunos, sem a devida visibilidade sobre sua frequência e evolução ao longo do período letivo. Dessa forma, o aplicativo visa proporcionar um acompanhamento mais amplo e detalhado do desempenho acadêmico, garantindo que informações essenciais estejam acessíveis de maneira centralizada e contínua.

Embora existam diversas plataformas acadêmicas disponíveis, muitas delas são voltadas para instituições de grande porte e não oferecem flexibilidade para professores que desejam gerenciar suas próprias turmas. O aplicativo proposto busca preencher essa lacuna, proporcionando uma solução mais prática e acessível para aqueles que necessitam de uma ferramenta complementar para organizar seu fluxo de trabalho.

É importante enfatizar que a proposta não tem o intuito de substituir sistemas institucionais, mas sim atuar como uma ferramenta de apoio para professores que buscam otimizar sua organização acadêmica. Dessa forma, o aplicativo se propõe a preencher uma lacuna existente no mercado, proporcionando maior autonomia na gestão de turmas e oferecendo recursos que facilitem o dia a dia de docentes e alunos.

1.2 Objetivo Geral

Propor um aplicativo *mobile*, baseado em arquitetura limpa, desenvolvido em *Flutter*, voltado para a gestão de turmas, auxiliando professores no gerenciamento de alunos, lançamento de notas e frequência, além de permitir que alunos e responsáveis acompanhem o desempenho acadêmico por meio de um acesso próprio.

1.3 Objetivos Específicos

- Definir os requisitos necessários para a concepção do aplicativo, considerando usabilidade, acessibilidade e necessidades dos professores;
- Desenvolver o projeto seguindo uma arquitetura baseada em princípios de *Clean Code*, garantindo organização, legibilidade e boas práticas no código-fonte;
- Construir um protótipo funcional da aplicação, contemplando sua interface e principais funcionalidades;

- Implementar funcionalidades essenciais, como o gerenciamento de turmas e alunos, lançamento de notas e frequência, e acesso do aluno/responsável às informações acadêmicas.

2 TRABALHOS CORRELATOS

Neste capítulo, serão referenciados alguns trabalhos que compartilham objetivos similares ou apresentam relação com o presente trabalho.

2.1 O Moodle como ferramenta didática

No artigo proposto (ALENCAR et al., 2011) é falado sobre o uso do *Moodle* como ferramenta didática em cursos presenciais e a distância. O Moodle é um *Learning Management System* (LMS) que permite ao professor gerenciar um curso a distância, provendo o planejamento, implementação e gestão do aprendizado a distância, permitindo inclusive o uso em cursos semipresenciais ou para a publicação de materiais que complementem os cursos presenciais. O artigo aborda as vantagens e desvantagens do uso do *Moodle* em ambas as situações, e os resultados que estão sendo obtidos com seu uso cada vez mais abrangente.

Figura 1 – Estatísticas do Moodle

Sites registrados	67,990
Países	219
Cursos	5,552,044
Usuários	54,242,700
Professores	1,288,006
Inscrições	24,278,218
Postagens no fórum	88,426,984
Recursos	49,205,096
Questões do quiz	100,642,803

Fonte: (ALENCAR et al., 2011)

Com o uso de um LMS, o aluno passa a ser responsável pela aquisição de seu conhecimento, desenvolvendo autonomia, perseverança, domínio de leitura e interpretação, ou seja, formando-se autodidata. Na era da informação, esta característica se torna imprescindível e potencializa a capacidade dos estudantes de lidar com a sociedade globalizada. Além disso, a utilização do *Moodle* permite a flexibilidade dos materiais didáticos concebidos e estruturados no formato digital, que permite a atualização constante de dados e informações.

2.2 Gestão Educacional e Inovação: o uso das plataformas digitais na escola

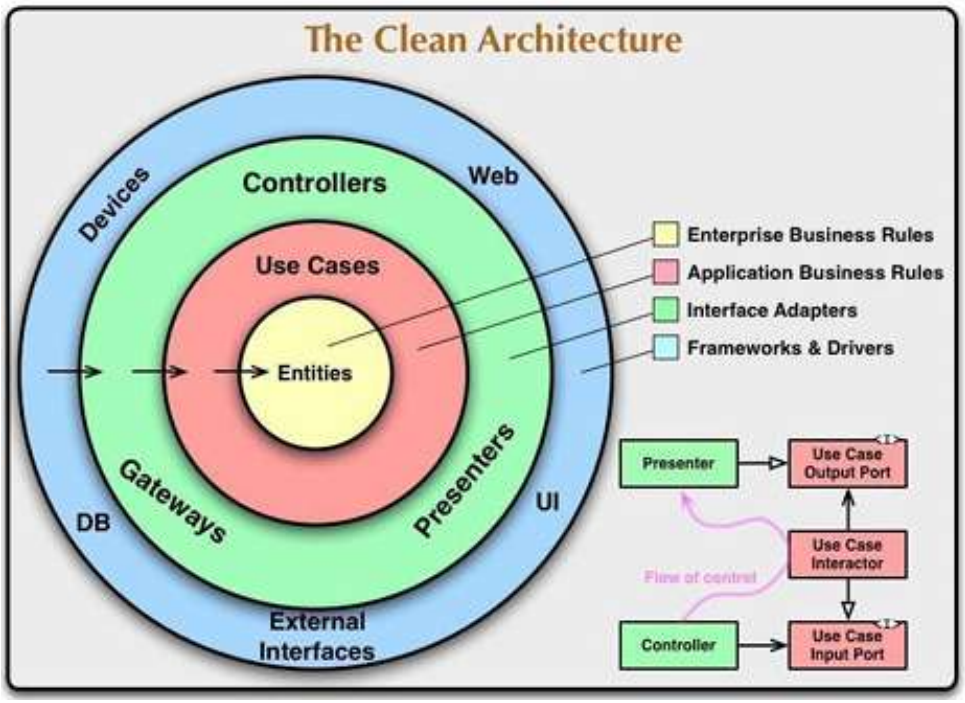
Na dissertação proposta (SOUSA, 2020) é apresentado uma análise sobre como as plataformas digitais podem ser utilizadas para melhorar a gestão educacional nas escolas e universidades. O trabalho aborda questões como as possibilidades que as plataformas digitais proporcionam para uma efetiva análise de desempenho do professor, além das questões relacionadas ao uso intenso das plataformas digitais nas dinâmicas da aula e como a gestão é capaz de mensurar e avaliar. A dissertação também apresenta um estudo de caso descritivo de uma escola particular da cidade de São Luís, estado do Maranhão, Brasil, que é considerada nacional e internacionalmente como referência e pioneira na implantação de tecnologias educacionais no estado do Maranhão.

De acordo com a autora, a utilização das plataformas digitais pode melhorar a qualidade do ensino nas escolas e universidades de diversas maneiras. Uma delas é a possibilidade de apresentar relatórios de desempenho dos professores e dos alunos, o que permite uma análise mais precisa e individualizada do processo de ensino e aprendizagem. Outra vantagem é a comunicação em tempo real, que possibilita a comunicação imediata entre estudantes, professores e membros da equipe administrativa. Facilitando um esclarecimento ágil de dúvidas acadêmicas e administrativas, a comunicação em tempo real serviria como canal direto para orientação acadêmica, promovendo a troca de informações críticas. Além disso, viabiliza a colaboração eficiente em projetos acadêmicos, proporcionando um ambiente dinâmico e interativo.

2.3 Desenvolvimento de um Aplicativo Utilizando o Framework Flutter e Arquitetura Limpa

No trabalho de Bueno (BUENO, 2021), é apresentada a criação de um aplicativo mobile utilizando o framework *Flutter* e princípios da *Arquitetura Limpa*. O objetivo do projeto foi desenvolver uma aplicação com autenticação de usuários via e-mail e senha, demonstrando a importância da organização em camadas bem definidas para facilitar a evolução e manutenção do sistema.

Figura 2 – Camadas da Arquitetura Limpa no Aplicativo



Fonte: (BUENO, 2021)

O trabalho enfatiza a utilização da *Arquitetura Limpa* como forma de criar sistemas mais organizados e desacoplados, separando responsabilidades em diferentes camadas, como Domínio, Dados, Infraestrutura e Apresentação. Além disso, o estudo aborda o uso do *Flutter* como um *framework* híbrido para o desenvolvimento da interface do usuário, destacando suas vantagens como rapidez no desenvolvimento e reutilização de código para múltiplas plataformas.

A pesquisa reforça a importância do uso de boas práticas arquiteturais no desenvolvimento de aplicativos, principalmente no contexto acadêmico e educacional, onde sistemas bem estruturados podem facilitar a gestão e o acompanhamento de informações.

2.4 Avaliação de Usabilidade de Aplicativos Móveis: Um Estudo de Caso sobre um Aplicativo de Ensino

O trabalho de Bonifácio et al. (BONIFáCIO et al., 2014) apresenta um estudo sobre usabilidade em aplicativos móveis para educação, utilizando como caso de estudo o aplicativo *Education Hub* (EH). O aplicativo foi desenvolvido com o objetivo de facilitar a comunicação entre professores, alunos e instituições de ensino, permitindo o compartilhamento de informações acadêmicas e recursos pedagógicos.

Figura 3 – Arquitetura do aplicativo Education Hub

Fonte: (BONIFáCIO et al., 2014)

O estudo destaca a importância da usabilidade no desenvolvimento de aplicativos educacionais, considerando que muitos usuários encontram dificuldades na adoção de tecnologias móveis devido a problemas na interface e na interação com os sistemas. Para avaliar a qualidade do *Education Hub*, foi realizada uma inspeção de usabilidade baseada na técnica *Usability-Based Inspection Customizable Approach* (UBICUA), adaptada para aplicações móveis. Durante a avaliação, foram identificados 52 problemas de usabilidade, os quais foram categorizados e resultaram em melhorias na interface do aplicativo.

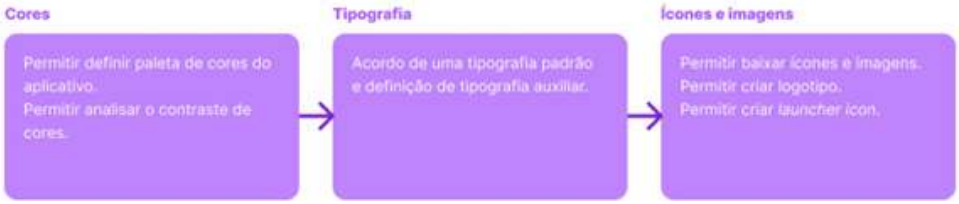
Além da análise de usabilidade, o artigo também discute a integração do *Education Hub* com sistemas de gestão acadêmica (*SGA*) e sistemas de gerenciamento de aprendizado (*LMS*), como o *Moodle* e o *Techne*, permitindo uma melhor adaptação a diferentes contextos educacionais. A pesquisa contribui para o desenvolvimento de ferramentas tecnológicas mais eficientes para o ensino, fornecendo diretrizes para o aprimoramento da experiência do usuário em aplicativos móveis educacionais.

2.5 Desenvolvimento de uma Ferramenta Web para Suporte ao Design Visual de Interface de Usuário de Aplicativos Móveis

O trabalho de Oliveira (OLIVEIRA, 2024) propõe o desenvolvimento de uma ferramenta web para auxiliar no design visual de interfaces de usuário de aplicativos móveis criados com o App Inventor. A pesquisa destaca a necessidade de um suporte mais estruturado para a criação de interfaces visuais dentro do contexto do ensino de

computação na Educação Básica.

Figura 4 – Fluxo de Design na Ferramenta Proposta



Fonte: (OLIVEIRA, 2024)

O estudo justifica a criação da ferramenta pela limitação do App Inventor em oferecer suporte adequado para o design visual de interfaces, o que resulta em aplicativos com baixa atratividade estética. Para resolver essa deficiência, a ferramenta proposta fornece suporte para a escolha de paletas de cores, tipografia, alinhamento de elementos e hierarquia visual, baseando-se em diretrizes de usabilidade e acessibilidade, como o Material Design e as WCAG (Web Content Accessibility Guidelines).

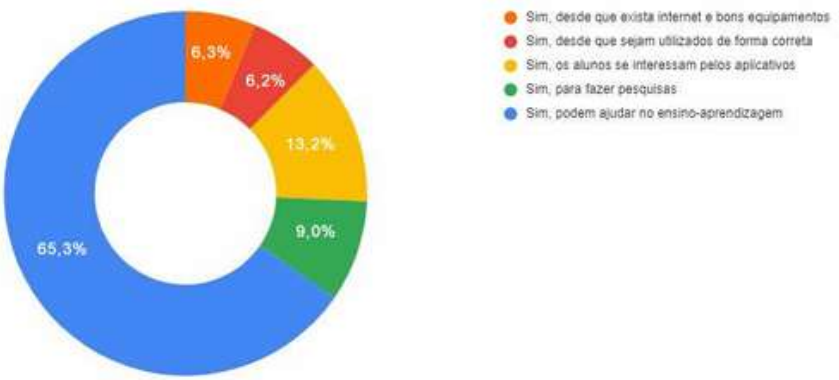
O desenvolvimento da ferramenta seguiu um processo estruturado de engenharia de software, incluindo análise de requisitos, prototipação e implementação. O sistema foi desenvolvido como uma aplicação web acessível, permitindo que alunos e professores utilizassem a ferramenta para criar designs mais harmoniosos e visualmente coerentes para seus aplicativos.

Os testes realizados demonstraram que a ferramenta melhora a usabilidade e a estética dos aplicativos criados com App Inventor, contribuindo para um aprendizado mais intuitivo sobre design visual e interação humano-computador. A pesquisa reforça a importância da inclusão do design visual no ensino de computação e apresenta uma solução inovadora para auxiliar nesse processo.

2.6 Uso de Aplicativos Educacionais em Escolas Públicas de Ensino Fundamental e Médio

O trabalho de Vieira e Silva (VIEIRA; SILVA, 2020) analisa o uso de aplicativos educacionais em escolas públicas de Ensino Fundamental e Médio na cidade de São Bento, Paraíba. O estudo investiga como os professores da Educação Básica utilizam esses aplicativos em sala de aula e os desafios enfrentados na implementação dessas tecnologias.

Figura 5 – Possibilidade de aplicativos auxiliarem o professor no ensino.



Fonte: (VIEIRA; SILVA, 2020)

A pesquisa utilizou abordagem quanti-qualitativa, baseada em questionários aplicados a 144 professores. Os resultados indicam que, apesar da presença de infraestrutura tecnológica em muitas escolas, a conectividade limitada e a falta de capacitação dos professores dificultam a adoção efetiva de aplicativos educacionais no ensino. Entre os aplicativos mais utilizados destacam-se o *Google Sala de Aula*, *YouTube*, *Kahoot* e *WhatsApp* como ferramentas auxiliares no ensino.

Além disso, o estudo revela que a falta de planejamento pedagógico e a resistência de alguns docentes ainda são obstáculos na integração dessas tecnologias ao ambiente escolar. No entanto, os professores que utilizam aplicativos educacionais relatam benefícios no engajamento dos alunos e na personalização do aprendizado. A pesquisa reforça a importância de investimentos em infraestrutura digital e capacitação docente para melhorar o uso dessas ferramentas na educação pública.

3 REFERENCIAL TEÓRICO

Neste capítulo será apresentado a revisão de conteúdos, conceitos e métodos que serão utilizados no desenvolvimento deste trabalho.

3.1 Clean Architecture

A *Clean Architecture* (MARTIN, 2017) é um padrão de arquitetura de software criado por *Robert C. Martin* que visa promover a criação de sistemas bem estruturados, flexíveis e testáveis. Essa abordagem se concentra na separação de preocupações e na independência de detalhes técnicos para facilitar a manutenção e a evolução do software ao longo do tempo.

A ideia central por trás da *Clean Architecture* é que o código fonte de um aplicativo deve ser organizado em camadas hierárquicas e cada camada deve ter responsabilidades bem definidas, com as camadas internas sendo independentes das camadas externas. Dessa forma, as mudanças em uma camada específica não devem afetar as outras camadas, tornando o sistema mais resiliente a alterações e mais fácil de entender e manter.

As principais camadas da *Clean Architecture* são as seguintes:

1. **Entidades (Entities):** Esta camada representa as entidades de negócios do sistema, como objetos que encapsulam dados e regras de negócios. Essas entidades devem ser independentes de qualquer detalhe técnico ou *framework*.

2. **Casos de uso (Use Cases):** Aqui ficam as regras de negócio e as funcionalidades principais do sistema. Os casos de uso orquestram as interações entre as entidades e coordenam as ações necessárias para realizar as operações do sistema.

3. **Interface do usuário (Interface Adapters):** Essa camada lida com a comunicação entre o mundo externo e as camadas internas do sistema. Ela converte as entradas e saídas do sistema (por exemplo, interações com a interface do usuário) em estruturas que as camadas internas entendam e vice-versa.

4. **Frameworks e drivers (Frameworks & Drivers):** Essa camada contém os detalhes técnicos e tecnológicos, como *frameworks* de *User Interface* (UI), banco de dados, web e outros dispositivos de entrada/saída. Esses detalhes são mantidos o mais afastados possível do núcleo do sistema para facilitar a substituição e a atualização de componentes técnicos.

5. **Núcleo (Core):** Essa é a camada central do sistema, contendo a lógica de negócios e as regras principais. Ela não depende de nenhuma camada externa e representa o coração da aplicação.

A direção do fluxo de dependências é importante na *Clean Architecture*. As camadas internas (Entidades e Casos de Uso) não dependem das camadas externas (Interface do usuário e *Frameworks/Drivers*), e sim o contrário. Isso significa que as camadas internas podem ser facilmente reutilizadas e testadas, pois não possuem dependências externas.

Essa arquitetura promove testabilidade, facilita a substituição de componentes, permite que o código seja mais independente de tecnologias específicas e ajuda a evitar que a lógica de negócios seja corrompida por detalhes de implementação. Além disso, a *Clean Architecture* favorece a manutenção e evolução do software, uma vez que mudanças podem ser feitas em camadas específicas sem afetar as outras partes do sistema.

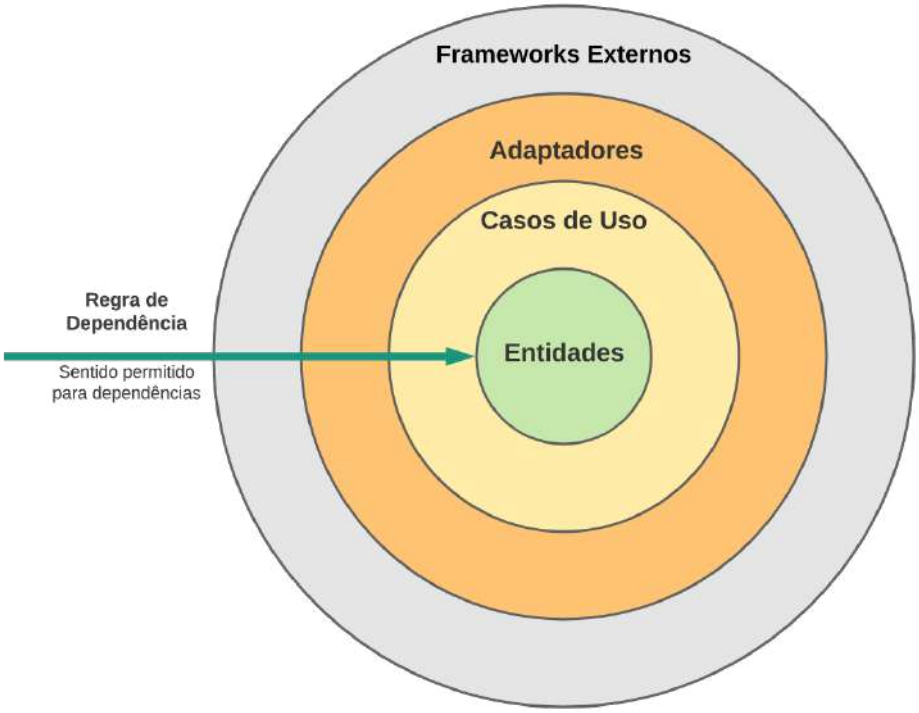
É importante ressaltar que a implementação exata da *Clean Architecture* pode variar de acordo com as necessidades e o contexto de cada projeto, mas a essência do padrão se mantém.

Se corretamente aplicada, a arquitetura limpa, segundo (BUCZYńSKI, 2020) possui os seguintes benefícios:

- **Frameworks independentes:** Atualizar ou substituir o framework deve ser significativamente menos complicado quando comparado a soluções que não utilizam o padrão;
- **Testabilidade:** Todas as regras de negócio podem ser testadas usando testes de unidade, sem a necessidade de inserir dados em um banco de dados;
- **Independência da UI / API:** O mecanismo de entrega não deve moldar a lógica do sistema;
- **Independência do banco de dados:** A forma de armazenar dados não deve limitar o desenvolvedor;
- **Independência de terceiros:** As regras de negócio não precisam estar vinculadas à plataforma de pagamentos utilizada;
- **Flexibilidade:** Certo aspecto das decisões arquiteturais pode ser adiado sem interromper o desenvolvimento;
- **Extensibilidade:** Os projetos podem ser facilmente estendidos com técnicas mais sofisticadas, como Command Query Responsibility Segregation (CQRS), Event Sourcing ou Domain-Driven Design, caso necessário.

A Figura 6 representa a arquitetura limpa. A seta ”Regra de Dependência”indica que as camadas internas podem acessar as camadas externas, mas o contrário não é permitido.

Figura 6 – Clean Architecture.



Fonte: (VALENTE, 2020)

3.2 Clean Code

Clean Code é um conceito fundamental no desenvolvimento de software que enfatiza a importância de escrever código claro, legível, organizado e de fácil manutenção. Foi popularizado pelo livro *Clean Code: A Handbook of Agile Software Craftsmanship*, escrito por *Robert C. Martin*. O objetivo do *Clean Code* é tornar o código fonte compreensível tanto para os desenvolvedores que o criam quanto para desenvolvedores que serão responsáveis pela manutenção futura.

Os Princípios do *Clean Code* são:

1. **Clareza:** O código deve ser expressivo e claro, evitando ambiguidades e abstrações desnecessárias. Isso inclui a escolha de nomes significativos para variáveis, funções e classes, de forma que o propósito de cada elemento seja facilmente identificável.

2. **Simplicidade:** O código deve ser o mais simples possível, mas não mais simples do que necessário. Evite complexidade desnecessária e abstrações complicadas. A simplicidade torna o código mais fácil de entender, depurar e modificar.

3. **Pequenas funções:** Funções devem ser curtas e fazer apenas uma coisa. Isso facilita a compreensão do que cada função faz e torna mais fácil a sua manutenção. Funções muito longas podem se tornar confusas e difíceis de entender.

4. **Testabilidade:** O código deve ser projetado de forma a facilitar a escrita de testes automatizados. A testabilidade é essencial para garantir a qualidade do software e permitir a sua evolução contínua.

5. **Composição sobre herança:** Em vez de usar herança extensivamente, o *Clean Code* favorece a composição, que permite uma maior flexibilidade e menor acoplamento entre as classes.

6. **Evitar repetições (DRY - Don't Repeat Yourself):** Código duplicado é difícil de manter, pois requer alterações em múltiplos lugares. O *Clean Code* procura eliminar duplicações e promove a reutilização de código por meio de abstrações apropriadas.

7. **Princípio da responsabilidade única (SRP - Single Responsibility Principle):** Cada classe ou módulo deve ter apenas uma única responsabilidade. Isso facilita a manutenção e evita que mudanças em uma parte do código afetem outras partes não relacionadas.

8. **Acoplamento e coesão:** O *Clean Code* busca reduzir o acoplamento entre as diferentes partes do sistema, tornando-as menos dependentes umas das outras. Além disso, procura maximizar a coesão de cada módulo, ou seja, garantir que ele contenha apenas elementos relacionados à sua funcionalidade.

Segundo (MARTIN, 2008), os benefícios de seguir os princípios *Clean Code* são:

- Facilita a compreensão do código para desenvolvedores, facilitando a colaboração em equipe.
- Melhora a manutenção do software, tornando mais fácil fazer alterações e corrigir problemas.
- Reduz o tempo gasto em depuração e correção de erros.
- Favorece a evolução contínua do software, permitindo que ele seja adaptado a novos requisitos.
- Contribui para um código mais seguro, uma vez que problemas de lógica e erros são mais fáceis de serem identificados e corrigidos.
- Aumenta a satisfação do cliente, pois resulta em um produto mais confiável e com menor probabilidade de bugs.

3.3 SOLID

Os princípios *SOLID* são um conjunto de cinco princípios de design de software que visam facilitar a criação de código limpo, flexível, extensível e de fácil manutenção. Eles foram formulados por *Robert C. Martin* e são amplamente adotados na comunidade de desenvolvimento de software.

Figura 7 – SOLID

Fonte: (PIRES, 2015)

Cada letra do acrônimo S.O.L.I.D representa um princípio específico, como demostrado na figura 7 e conforme explicado abaixo:

1. **Princípio da Responsabilidade Única (SRP - Single Responsibility Principle):** Este princípio afirma que uma classe deve ter uma única responsabilidade, ou seja, ela deve ter apenas uma razão para mudar. Isso implica que uma classe deve representar apenas uma parte específica do comportamento do sistema. Se uma classe tiver mais de uma responsabilidade, mudanças em uma área podem causar efeitos colaterais em outra, tornando o código mais difícil de manter e entender. Separar responsabilidades em classes distintas reduz o acoplamento e torna o código mais coeso.

2. **Princípio do Aberto/Fechado (OCP - Open/Closed Principle):** O Princípio do Aberto/Fechado defende que uma entidade de software, como uma classe, deve estar aberta para extensão, mas fechada para modificação. Isso significa que você pode estender o comportamento da entidade adicionando novas funcionalidades sem alterar o código existente. Isso é geralmente alcançado por meio do uso de abstrações, como interfaces ou classes abstratas, e implementações concretas que seguem essas abstrações.

3. **Princípio da Substituição de Liskov (LSP - Liskov Substitution Principle):** Este princípio enfatiza que um objeto de uma classe derivada deve ser capaz de ser substituído por um objeto da classe base sem afetar a corretude do programa. Em outras palavras, se uma classe B herda de uma classe A, ela deve respeitar o contrato da classe A e não alterar o comportamento esperado pelas classes que utilizam a classe A. Isso garante que a herança seja usada corretamente, evitando surpresas indesejadas em tempo de execução.

4. **Princípio da Segregação de Interface (ISP - Interface Segregation Principle):** O Princípio da Segregação de Interface sugere que uma classe não deve ser forçada a depender de interfaces que ela não usa. Em vez disso, é melhor ter interfaces mais especializadas, contendo apenas os métodos necessários para um conjunto específico de funcionalidades. Isso evita a criação de interfaces monolíticas que levam a uma dependência excessiva e não utilizada, promovendo um design mais modular e desacoplado.

5. **Princípio da Inversão de Dependência (DIP - Dependency Inversion Principle):** O Princípio da Inversão de Dependência aborda a direção da dependência entre módulos de um software. Ele sugere que módulos de alto nível não devem depender diretamente de módulos de baixo nível, mas sim de abstrações. Além disso, abstrações não devem depender de detalhes concretos. Isso possibilita a flexibilidade no design, tornando-o mais adaptável a mudanças e permitindo que diferentes implementações sejam facilmente substituídas sem afetar o funcionamento do código que as utiliza.

Segundo (OLORUNTOBA, 2020) o uso dos princípios S.O.L.I.D em um software traz diversos benefícios significativos para o processo de desenvolvimento e para a qualidade do código produzido. Abaixo estão alguns dos principais benefícios citados:

- **Flexibilidade e Extensibilidade:** Ao seguir o Princípio do Aberto/Fechado (OCP) e o Princípio da Inversão de Dependência (DIP), o código se torna mais flexível e facilmente extensível. Novas funcionalidades podem ser adicionadas ao sistema sem alterar o código existente, o que reduz o risco de introduzir bugs e torna a evolução do software mais rápida e segura.

- **Facilidade de Manutenção:** Os princípios S.O.L.I.D, como o Princípio da Responsabilidade Única (SRP), ajudam a manter o código organizado e coeso. Isso torna a manutenção do software mais simples, pois cada componente tem uma única responsabilidade bem definida e mudanças em uma parte do sistema têm um impacto controlado sobre outras partes.

- **Baixo Acoplamento:** O Princípio da Inversão de Dependência (DIP) reduz o acoplamento entre os módulos do sistema, tornando-os menos dependentes uns dos outros. Isso facilita a modificação de partes específicas do código sem afetar o restante do sistema, permitindo uma maior escalabilidade e flexibilidade na arquitetura do software.

- **Alta Coesão:** Os princípios S.O.L.I.D, como o Princípio da Responsabilidade Única (SRP) e o Princípio da Segregação de Interface (ISP), promovem a alta coesão no código, ou seja, cada componente ou classe tem um propósito claro e bem definido. Isso resulta em classes mais enxutas e focadas, facilitando sua compreensão e manutenção.

- **Reutilização de Código:** Através do Princípio do Aberto/Fechado (OCP) e da Inversão de Dependência (DIP), a criação de abstrações e interfaces bem definidas permite a reutilização de código em diferentes contextos. Isso aumenta a eficiência do desenvolvimento, pois menos código precisa ser escrito, e promove um design modular, onde partes do software podem ser facilmente intercambiadas ou substituídas.

- **Testabilidade:** O design orientado a interfaces e a baixa dependência entre componentes, resultado do Princípio da Inversão de Dependência (DIP), torna o código mais fácil de ser testado. A criação de mocks e simulações para testes de unidade é facilitada, permitindo uma cobertura de testes mais completa.

- **Maior Qualidade do Software:** Ao adotar os princípios S.O.L.I.D, o código se torna mais confiável, menos suscetível a erros e mais resistente a mudanças. Isso leva a um software de maior qualidade e mais robusto, com menos falhas e problemas em produção.

- **Facilidade de Colaboração:** O uso dos princípios S.O.L.I.D, como o Princípio da Responsabilidade Única (SRP) e o Princípio da Segregação de Interface (ISP), torna o código mais legível e compreensível. Isso facilita a colaboração entre desenvolvedores, pois todos podem entender mais facilmente o que cada componente faz e como eles interagem.

4 METODOLOGIA

Neste capítulo, serão abordadas as fases e os processos utilizados para a estruturação da proposta do trabalho, bem como serão descritas as ferramentas, materiais e metodologias que foram empregadas ao longo deste projeto.

4.1 Ferramentas

Nesta seção, serão descritas as ferramentas e *frameworks* que foram utilizadas na concepção da aplicação. Serão apresentadas as tecnologias e estruturas analisadas para construir a aplicação, abrangendo as ferramentas de design, recursos auxiliares e os *frameworks* que forneceram a base para a estrutura da aplicação.

4.1.1 Figma

O Figma é uma plataforma colaborativa para design de interfaces e prototipação de aplicativos. Lançado em 2016, o Figma se destacou no mercado por ser uma ferramenta baseada em nuvem, permitindo o trabalho simultâneo entre vários usuários sem a necessidade de instalações locais. Seu uso é amplamente adotado na indústria de desenvolvimento de software devido à sua acessibilidade, interface intuitiva e suporte a design responsivo.

No desenvolvimento deste projeto, o Figma foi utilizado na etapa de prototipação da aplicação. Inicialmente, foram criados os esboços das telas, permitindo a visualização da disposição dos elementos e a organização do fluxo de navegação dentro do aplicativo. Com base nesses protótipos, foi possível definir a identidade visual da aplicação, incluindo a escolha da paleta de cores, tipografia e estilos dos componentes gráficos.

Além da prototipação, o Figma também serviu como um guia para o desenvolvimento da interface, auxiliando na implementação das telas no *Flutter*. A possibilidade de exportação de ativos gráficos e a facilidade de compartilhamento do projeto entre os envolvidos foram fatores essenciais para garantir a consistência do design ao longo do desenvolvimento.

Dessa forma, o Figma desempenhou um papel fundamental na construção da interface do aplicativo, proporcionando uma abordagem organizada e visualmente estruturada para a definição dos elementos da aplicação.

4.1.2 Flutter

Flutter é um *framework* de código aberto desenvolvido pelo Google que permite a criação de aplicativos multiplataforma de alta qualidade para dispositivos móveis, web e desktop. Ele foi lançado pela primeira vez em maio de 2017 e, desde então, tem ganhado muita popularidade entre desenvolvedores devido à sua eficiência, performance e facilidade

de desenvolvimento. O Flutter utiliza a linguagem de programação *Dart* (mencionada na seção 4.1.3), também desenvolvida pelo Google, como a principal linguagem para criar aplicativos.

Motivos pelos quais foi escolhido para este projeto:

1. **Arquitetura reativa:** O Flutter utiliza uma arquitetura reativa baseada em *widgets*, o que significa que a interface do aplicativo é construída a partir de *widgets*, que são elementos de interface de usuário (UI) como botões, campos de texto, imagens, etc. Os *widgets* são organizados em uma árvore de *widgets*, e qualquer alteração em um *widget* específico irá refletir automaticamente nas partes relacionadas do aplicativo.

2. **Hot Reload:** Uma das características mais notáveis do Flutter é o recurso "*Hot Reload*". Ele permite que os desenvolvedores visualizem as mudanças feitas no código imediatamente no aplicativo em execução, sem a necessidade de reiniciar o aplicativo por completo. Isso agiliza significativamente o processo de desenvolvimento, tornando-o mais ágil e interativo.

3. **Multiplataforma:** Com Flutter, um único código base pode ser utilizado para criar aplicativos para diferentes plataformas, como Android, iOS, Web e desktop. Isso é possível porque os *widgets* do Flutter são renderizados nativamente em cada plataforma, proporcionando uma experiência de usuário consistente.

4. **Performance nativa:** Ao contrário de outros *frameworks* multiplataforma, o Flutter não usa um mecanismo de renderização baseado em *webview*. Em vez disso, ele utiliza o mecanismo de renderização *Skia*, que é o mesmo utilizado pelo *Google Chrome*. Isso resulta em uma performance nativa, com alta velocidade e fluidez nas animações.

4.1.3 Dart

Dart é uma linguagem de programação desenvolvida pelo Google com o objetivo de fornecer uma linguagem moderna e de alto desempenho para o desenvolvimento de aplicativos web, mobile e de servidor. Ela oferece tipagem opcional, o que permite escolher entre tipagem estática ou dinâmica, proporcionando flexibilidade no estilo de codificação.

Dart suporta duas formas de compilação: *Just-in-Time* (JIT) e Ahead-of-Time (AOT). A compilação JIT permite um rápido processo de desenvolvimento e depuração, enquanto a compilação AOT gera código nativo, otimizado para a plataforma de destino, melhorando o desempenho em tempo de execução.

A linguagem oferece recursos para programação assíncrona, facilitando o trabalho com tarefas que envolvem operações demoradas, como chamadas de Application Programming Interface (API) e requisições de rede, sem bloquear a execução do programa.

Dart foi projetada para ser legível e concisa, com uma sintaxe limpa e familiar a outras linguagens populares.

Devido ao suporte multiplataforma fornecido pelo Flutter (mencionado na seção 4.1.2), o *Dart* é amplamente utilizado para o desenvolvimento de aplicativos móveis e web. Além disso, também é usado para o desenvolvimento de aplicações de servidor, permitindo que desenvolvedores criem sistemas web e APIs de alto desempenho.

4.1.4 NestJS

NestJS é um *framework back-end* de código aberto construído em *TypeScript* e *Node.js*, projetado para criar aplicações escaláveis e eficientes. Lançado em 2017, o *NestJS* foi inspirado na estrutura do Angular, o *framework front-end* do *Google*, e adota conceitos semelhantes para o desenvolvimento de servidor.

O *NestJS* utiliza o padrão de arquitetura de software *Model-View-Controller* (MVC) com uma abordagem modular, na qual o código é organizado em módulos para representar diferentes partes da aplicação. Essa abordagem traz benefícios em termos de organização, facilitando a manutenção e a extensibilidade do código.

O *TypeScript* é a linguagem base do *NestJS*, proporcionando uma experiência de desenvolvimento melhorada com tipagem estática e evitando erros comuns. *Decorators* são amplamente utilizados para adicionar metadados ao código, tornando-o mais claro e expressivo.

Um dos principais princípios do *NestJS* é a injeção de dependências, que facilita a criação de componentes reutilizáveis e simplifica os testes unitários. Isso torna o código mais flexível e fácil de manter.

Além disso, o *NestJS* é altamente extensível, permitindo a integração com outros módulos e bibliotecas de forma simples e eficaz. A comunidade oferece diversos módulos que facilitam a implementação de recursos adicionais, como autenticação, armazenamento em banco de dados, *logging*, entre outros.

Em resumo, o *NestJS* foi escolhido para este projeto pois é uma *framework* poderosa para o desenvolvimento de aplicações *back-end*, proporcionando uma experiência de desenvolvimento organizada e robusta, com ênfase na escalabilidade e manutenção do código. Sua popularidade crescente é resultado de sua eficiência e aprimoramento contínuo pela comunidade de desenvolvedores.

4.1.5 MongoDB

O *MongoDB* é um sistema de gerenciamento de banco de dados *Not Only SQL* (NoSQL) de código aberto, lançado em 2009. Ele foi projetado para atender às demandas

de armazenamento e gerenciamento de dados de aplicativos modernos, que requerem flexibilidade, escalabilidade e desempenho para lidar com grandes volumes de informações.

Ao contrário dos bancos de dados relacionais tradicionais, o *MongoDB* é classificado como um banco de dados NoSQL. Isso significa que ele não segue a estrutura tabular de tabelas e linhas do modelo relacional, optando por um modelo de dados mais flexível baseado em documentos *JavaScript Object Notation* (JSON). Os dados são armazenados como documentos, que podem conter campos e valores heterogêneos, permitindo que as estruturas de dados variem de documento para documento.

Suas principais características são:

1. **Esquema Flexível:** O *MongoDB* é conhecido por sua natureza "*schemaless*", o que significa que os documentos dentro de uma coleção não precisam seguir um esquema rígido. Cada documento pode ter campos diferentes, e os dados podem ser facilmente adicionados ou removidos sem alterar a estrutura geral do banco de dados. Isso proporciona uma grande flexibilidade ao modelar os dados, especialmente em aplicações que evoluem com o tempo.

2. **Alta Performance e Escalabilidade:** O *MongoDB* é projetado para oferecer alta performance e escalabilidade horizontal, o que significa que ele pode ser facilmente distribuído em vários servidores para acomodar grandes volumes de dados e tráfego. Com a replicação e o particionamento automático (*sharding*), é possível garantir a disponibilidade e a escalabilidade da aplicação, permitindo o crescimento conforme necessário.

3. **Consulta Avançada e Indexação:** O *MongoDB* suporta uma ampla variedade de consultas complexas, permitindo buscar e filtrar dados com eficiência. Além disso, ele oferece suporte a indexação para melhorar o desempenho das consultas, tornando as operações de leitura e escrita mais eficientes.

O *MongoDB* foi escolhido para este projeto pois se destaca como uma solução versátil para o gerenciamento de dados em aplicações modernas. Possui flexibilidade, alto desempenho e escalabilidade. Seu modelo NoSQL, combinado com sua comunidade ativa e suporte empresarial, faz do *MongoDB* uma opção atraente para desenvolvedores que buscam um banco de dados robusto e adaptável para atender às demandas em constante evolução do mundo da tecnologia.

O *Figma* é uma ferramenta de design de interface do usuário (UI) que se destaca por sua abordagem baseada na web e recursos avançados de colaboração. Diferente de muitas outras ferramentas de design, o *Figma* opera diretamente no navegador, o que permite que equipes de design trabalhem juntas em tempo real, independentemente da localização geográfica de cada membro.

Uma das principais vantagens do *Figma* é a capacidade de criar designs interativos e protótipos de alta fidelidade. Isso possibilita simular a experiência do usuário final com links, animações e transições, tornando mais fácil para os designers validarem suas ideias e compartilharem a visão do produto com outras partes interessadas.

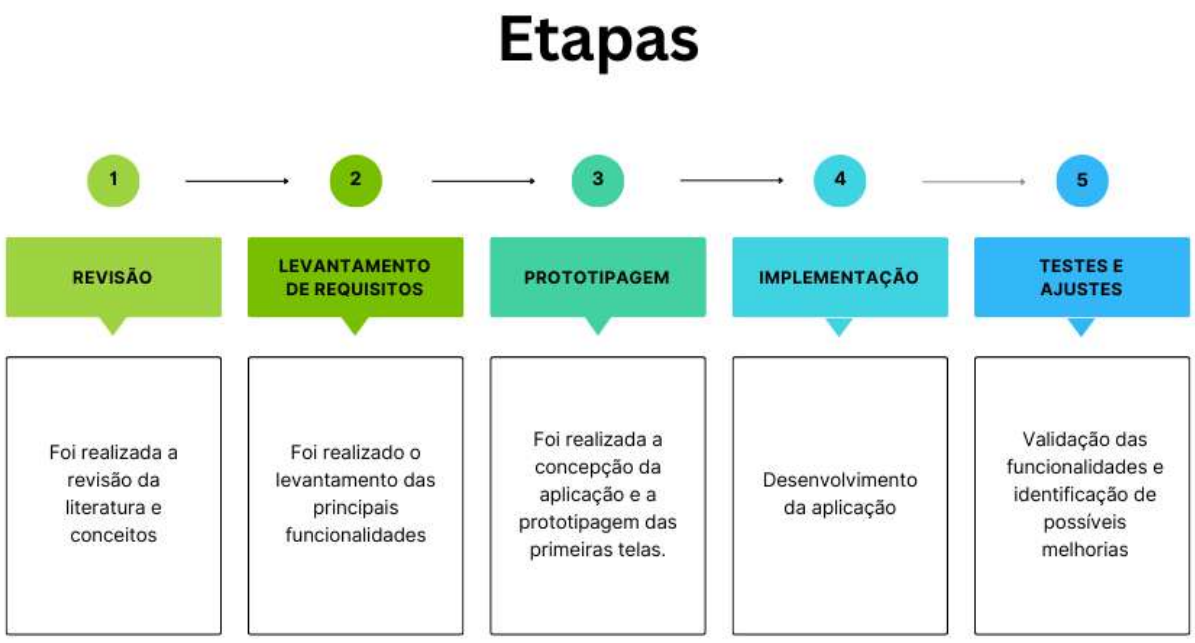
Outro recurso poderoso do *Figma* é a funcionalidade de componentes. Os designers podem criar elementos reutilizáveis, como botões, barras de navegação e ícones, que podem ser atualizados em todos os lugares onde são utilizados. Isso promove a consistência do design e agiliza o processo de criação.

Essas características tornam-no *Figma* a escolha ideal para o desenvolvimento das interfaces e experiência do usuário do aplicativo proposto por este trabalho.

4.2 Etapas da Proposta

Nesta seção serão descritas as etapas realizadas para o desenvolvimento da proposta do aplicativo, conforme demonstrado na Figura 8.

Figura 8 – Diagrama das Etapas da Proposta



Fonte: Autor

4.2.1 Revisão

Nesta etapa, foi realizada uma revisão detalhada dos fundamentos teóricos e dos princípios que embasam a proposta: *Clean Architecture*, *Clean Code* e os princípios SOLID. Adicionalmente, foram analisadas as principais características das tecnologias que comporão o cenário proposto, como a *framework* Flutter, o *NestJS*, a linguagem *Dart* e

o banco de dados *MongoDB*. Esse procedimento visou aprofundar a compreensão sobre tais temáticas, assegurando uma fundamentação consistente e a viabilidade da solução apresentada.

4.2.2 Levantamento de Requisitos

A definição dos requisitos baseia-se na identificação da necessidade de uma ferramenta digital que auxilie professores no gerenciamento de turmas e no acompanhamento do desempenho acadêmico dos alunos. Observa-se que muitos docentes ainda utilizam métodos tradicionais, como registros em papel, o que pode dificultar a organização e o controle de informações essenciais, como notas e frequência.

A proposta tem como objetivo oferecer uma solução prática e acessível, permitindo que os professores possam cadastrar turmas, registrar alunos e gerenciar dados acadêmicos de forma centralizada. Adicionalmente, prevê-se a disponibilização de um acesso para que alunos ou responsáveis possam acompanhar informações como notas e frequência, através de uma interface intuitiva. Para garantir uma experiência satisfatória, serão adotadas diretrizes de design centrado no usuário, priorizando a facilidade de navegação e a acessibilidade.

4.2.3 Prototipagem

Nesta etapa, foram elaborados protótipos que ilustram a concepção visual da proposta, definindo as interfaces e a organização das principais funcionalidades. A estrutura proposta contempla três telas fundamentais: **Turmas**, destinada ao acesso e gerenciamento das turmas; **Alunos**, voltada para o controle dos discentes; e **Acompanhamento**, que permitirá a visualização das informações acadêmicas pelos alunos ou responsáveis.

Foram realizadas avaliações para verificar se os protótipos oferecem uma experiência de usuário satisfatória, garantindo que o design seja atrativo, funcional e intuitivo.

4.2.4 Proposta de Implementação

Nesta etapa, foi apresentada uma proposta de implementação que descreve como o aplicativo poderia ser desenvolvido. O planejamento inclui a definição da estrutura do repositório no *Git*, a configuração inicial do projeto utilizando Flutter e a organização das interfaces gráficas conforme os protótipos elaborados. Ademais, serão delineados os possíveis fluxos de desenvolvimento do *back-end*, a integração com o banco de dados e a implementação das regras de negócio que assegurariam o funcionamento integral da aplicação.

4.2.5 Planejamento de Testes e Ajustes

Considerando a hipótese de uma futura implementação, esta etapa propõe a realização de testes internos teóricos para validar as funcionalidades e identificar potenciais

melhorias. O planejamento prevê a definição de critérios para a avaliação da experiência do usuário e da fluidez na navegação, permitindo que ajustes sejam incorporados à proposta para otimizar tanto a interface quanto a funcionalidade do sistema.

5 PROPOSTA DO TRABALHO

Este capítulo apresentará em detalhes o processo de planejamento e desenvolvimento do projeto, com destaque para as metodologias utilizadas.

5.1 Visão Geral do Sistema

O sistema desenvolvido é um aplicativo móvel voltado para a gestão de turmas, oferecendo aos professores uma ferramenta para organizar seus alunos, registrar frequência e lançar notas de forma prática e acessível. Além disso, o aplicativo possibilita que alunos e responsáveis acompanhem o desempenho acadêmico, garantindo maior transparência nas informações educacionais.

Entre as funcionalidades planejadas, destacam-se o cadastro e gerenciamento de turmas e alunos, o controle de frequência por meio de registros diretos ou leitura de *QR Code*, e a exibição de indicadores acadêmicos, como médias de notas ao longo do período letivo. O sistema também contará com notificações *push*, permitindo que professores enviem lembretes sobre prazos ou comunicados importantes aos alunos.

A proposta visa proporcionar uma solução intuitiva e eficiente, atendendo docentes com diferentes níveis de familiaridade com tecnologia. Nos tópicos a seguir, serão detalhados os requisitos do aplicativo, suas principais funcionalidades e a estrutura técnica utilizada no desenvolvimento da proposta.

5.2 Requisitos Funcionais

5.2.1 Cadastro e Autenticação

- **RF01** : O sistema deve permitir que professores se cadastrem de forma independente, fornecendo nome, e-mail e senha.
- **RF02** : O sistema deve permitir que professores façam login com e-mail e senha.
- **RF03** : O sistema deve permitir a recuperação de senha via e-mail.
- **RF04** : O sistema deve garantir que cada professor tenha uma conta isolada, sem acesso aos dados de outros professores.
- **RF05** : O sistema deve permitir que professores gerem usuários e senhas para alunos/responsáveis, fornecendo acesso à área de acompanhamento acadêmico.
- **RF06** : O sistema deve permitir que alunos/responsáveis façam login em uma tela exclusiva, utilizando o usuário e senha fornecidos pelo professor.

5.2.2 Gestão de Turmas

- **RF07** : O sistema deve permitir que professores criem turmas, informando nome, disciplina e permitindo a seleção dos alunos participantes.
- **RF08** : O sistema deve permitir que professores editem ou excluam turmas existentes.
- **RF09** : O sistema deve permitir que professores visualizem uma lista de todas as suas turmas cadastradas.
- **RF10** : O sistema deve garantir que as turmas de um professor não sejam acessíveis ou visíveis para outros professores.

5.2.3 Gestão de Alunos

- **RF11** : O sistema deve permitir que professores cadastrem alunos em suas contas, informando nome, e-mail e outras informações relevantes.
- **RF12** : O sistema deve permitir que professores editem ou excluam alunos de suas turmas.
- **RF13** : O sistema deve permitir que professores visualizem uma lista de todos os alunos de suas turmas.
- **RF14** : O sistema deve garantir que os dados dos alunos de um professor não sejam acessíveis ou visíveis para outros professores.

5.2.4 Registro de Presença

- **RF15** : O sistema deve permitir que professores gerem um QR Code único para cada aula, válido por um período de tempo limitado.
- **RF16** : O sistema deve permitir que alunos registrem presença escaneando o QR Code gerado pelo professor.
- **RF17** : O sistema deve armazenar o registro de presença de cada aluno, associando-o à turma e à data da aula.
- **RF18** : O sistema deve permitir que professores visualizem o histórico de presença de seus alunos.
- **RF19** : O sistema deve garantir que os registros de presença de um professor não sejam acessíveis ou visíveis para outros professores.

5.2.5 Lançamento de Notas

- **RF20** : O sistema deve permitir que professores lancem notas para seus alunos, associando-as a atividades específicas (ex.: provas, trabalhos).
- **RF21** : O sistema deve permitir que professores editem ou excluam notas já lançadas.
- **RF22** : O sistema deve calcular automaticamente a média de cada aluno com base nas notas lançadas.
- **RF23** : O sistema deve garantir que as notas de um professor não sejam acessíveis ou visíveis para outros professores.

5.2.6 Acompanhamento Acadêmico

- **RF24** : O sistema deve permitir que professores gerem usuários e senhas para alunos/responsáveis, fornecendo acesso à área de acompanhamento acadêmico.
- **RF25** : O sistema deve permitir que alunos/responsáveis acessem uma área exclusiva para visualizar notas, frequência e desempenho acadêmico, utilizando o usuário e senha fornecidos pelo professor.
- **RF26** : O sistema deve exibir gráficos e indicadores de desempenho (ex.: evolução das notas ao longo do tempo, percentual de presença).
- **RF27** : O sistema deve enviar notificações push para alunos e responsáveis sobre prazos de atividades, datas de provas e eventos importantes.
- **RF28** : O sistema deve garantir que os dados de acompanhamento acadêmico de um professor não sejam acessíveis ou visíveis para outros professores.

5.3 Requisitos Não Funcionais

5.3.1 Isolamento de Dados

- **RNF01** : O sistema deve garantir que os dados de cada professor (turmas, alunos, notas, presenças) sejam completamente isolados e inacessíveis para outros professores.
- **RNF02** : O sistema deve utilizar mecanismos de autenticação e autorização para garantir que apenas o professor dono dos dados possa acessá-los.

5.3.2 Usabilidade

- **RNF03** : O sistema deve ter uma interface intuitiva e de fácil navegação, seguindo princípios de design centrado no usuário.

- **RNF04** : O sistema deve ser responsivo e funcionar corretamente em dispositivos móveis (Android e iOS).
- **RNF05** : O sistema deve fornecer telas de login separadas e personalizadas para professores e alunos/responsáveis.

5.3.3 Segurança

- **RNF06** : O sistema deve criptografar dados sensíveis (ex.: senhas, informações de alunos).
- **RNF07** : O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados (LGPD).
- **RNF08** : O sistema deve garantir que as credenciais de alunos/responsáveis sejam geradas de forma segura e única para cada usuário.

5.3.4 Manutenibilidade

- **RNF09** : O código-fonte do sistema deve seguir as boas práticas de Clean Code e SOLID, facilitando a manutenção e evolução.
- **RNF10** : O sistema deve ser modular, permitindo a adição de novas funcionalidades com impacto mínimo no código existente.

5.3.5 Compatibilidade

- **RNF11** : O sistema deve ser compatível com as versões mais recentes do Android e iOS.
- **RNF12** : O sistema deve funcionar em diferentes tamanhos de tela e resoluções.

5.4 Projeto da Interface (UI/UX)

5.4.1 Prototipação e Design

A interface do aplicativo foi projetada utilizando a ferramenta *Figma*, que permitiu a criação de protótipos interativos e de alta fidelidade. O processo de design foi guiado por princípios de *User Experience* (UX) e *User Interface* (UI), visando proporcionar uma experiência intuitiva (RNF03), acessível e agradável para professores, alunos e responsáveis.

5.4.1.1 Identidade Visual

A identidade visual do aplicativo foi cuidadosamente planejada para transmitir confiança e clareza, alinhada ao contexto educacional. A paleta de cores foi escolhida com base na psicologia das cores, onde:

- **Azul (** **)**: Utilizado nas telas do professor, o azul transmite confiança, seriedade e profissionalismo, características essenciais para o ambiente educacional. Além disso, o azul é uma cor que promove a concentração e a clareza mental, ajudando os professores a se manterem focados em suas tarefas, como o gerenciamento de turmas, o lançamento de notas e o acompanhamento do desempenho dos alunos.
- **Azul Turquesa (** **)**: Utilizado nas telas do aluno, o azul turquesa traz uma sensação de calma e tranquilidade, ideal para um ambiente de aprendizado. Ele também é associado à criatividade e à comunicação, características que podem estimular o engajamento dos alunos com o conteúdo acadêmico.
- **Branco e Preto**: Foram utilizados para garantir contraste e legibilidade, melhorando a acessibilidade e a visibilidade dos elementos. Essa combinação clássica também contribui para uma interface moderna e minimalista, que não sobrecarrega visualmente o usuário.
- **Cinza Escuro (** **)**: Utilizado para textos secundários e elementos de fundo, proporcionando um equilíbrio visual. Essa tonalidade ajuda a criar uma hierarquia visual clara, guiando o usuário para as informações mais importantes sem distrações desnecessárias.

As fontes escolhidas foram *Poppins*, *Inter* e *DM Sans*, que combinam modernidade e legibilidade. Essas fontes foram selecionadas por suas características distintas:

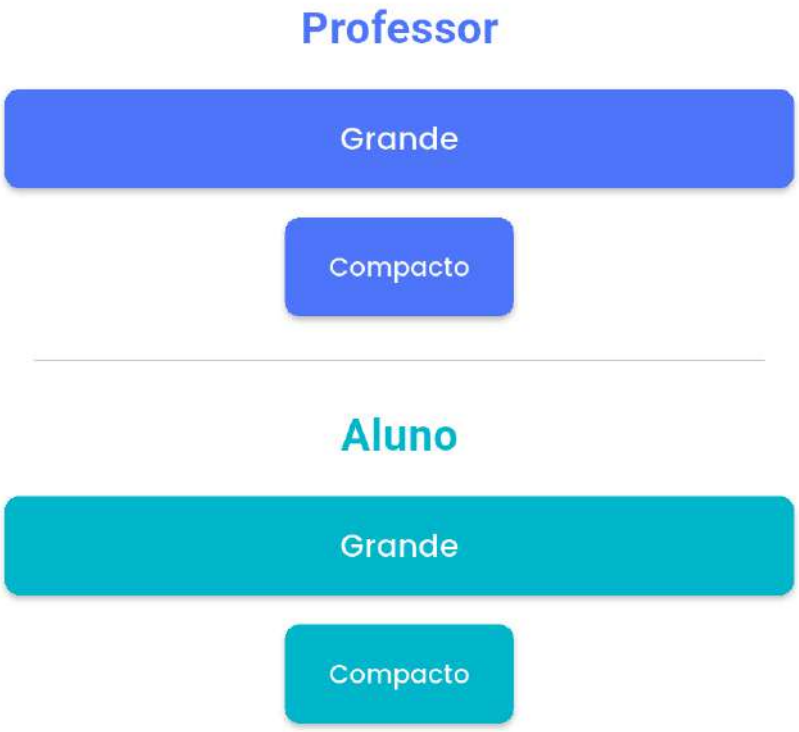
- **Poppins**: Uma fonte geométrica e arredondada, que transmite modernidade e simplicidade. Sua legibilidade em diferentes tamanhos a torna ideal para títulos e textos destacados.
- **Inter**: Uma fonte altamente legível e versátil, projetada para interfaces digitais. Sua neutralidade e clareza a tornam perfeita para textos longos e informações secundárias.
- **DM Sans**: Uma fonte humanista e amigável, que adiciona um toque de personalidade à interface, sem comprometer a legibilidade.

5.4.1.2 Componentes Reutilizáveis

Para garantir consistência e agilidade no desenvolvimento, foram criados componentes reutilizáveis, como:

- **Botões**: Com variações de tamanho (pequeno e grande) e estilos padronizados.

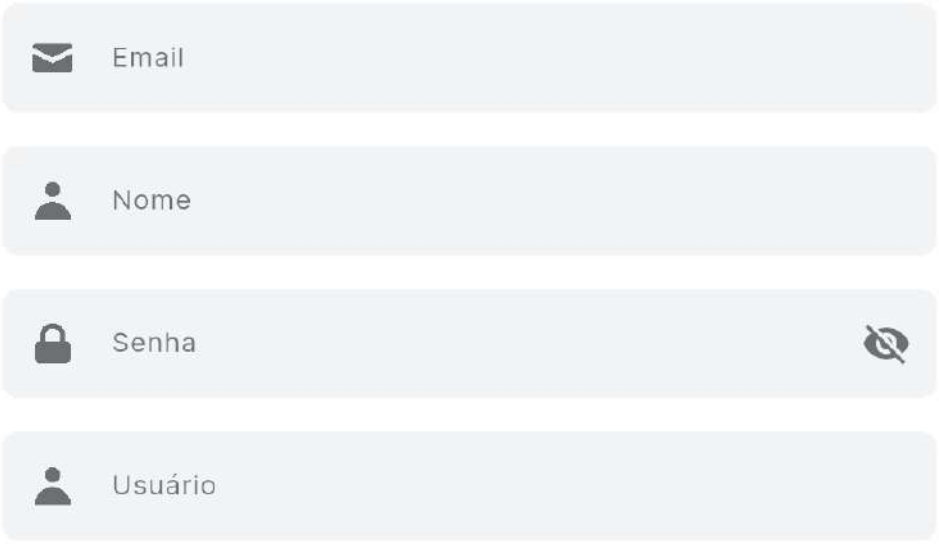
Figura 9 – Variações de botões para professor e aluno



Fonte: Autor

- **Campos de Texto**: Com validação e feedback visual integrados.

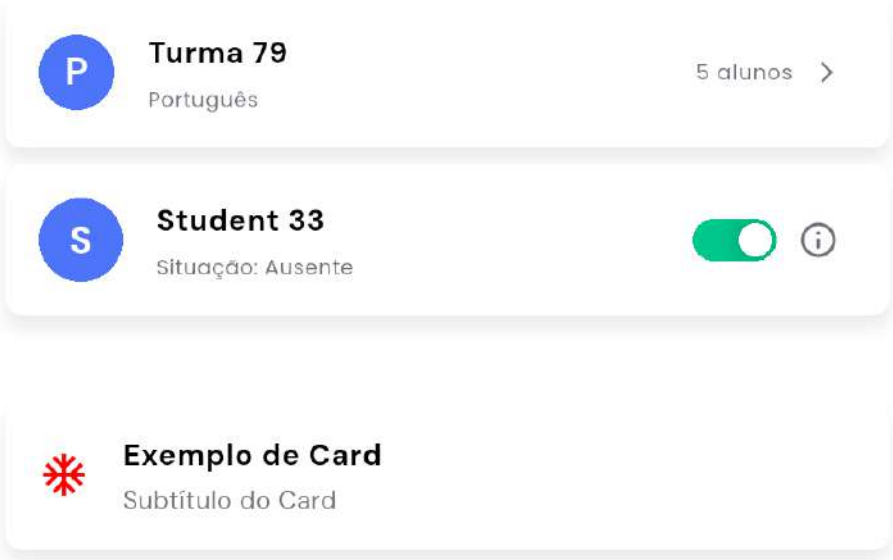
Figura 10 – Campos de textos



Fonte: Autor

- **Containers**: Com estilos pré-definidos para listas e cards.

Figura 11 – Variações e personalização de cards



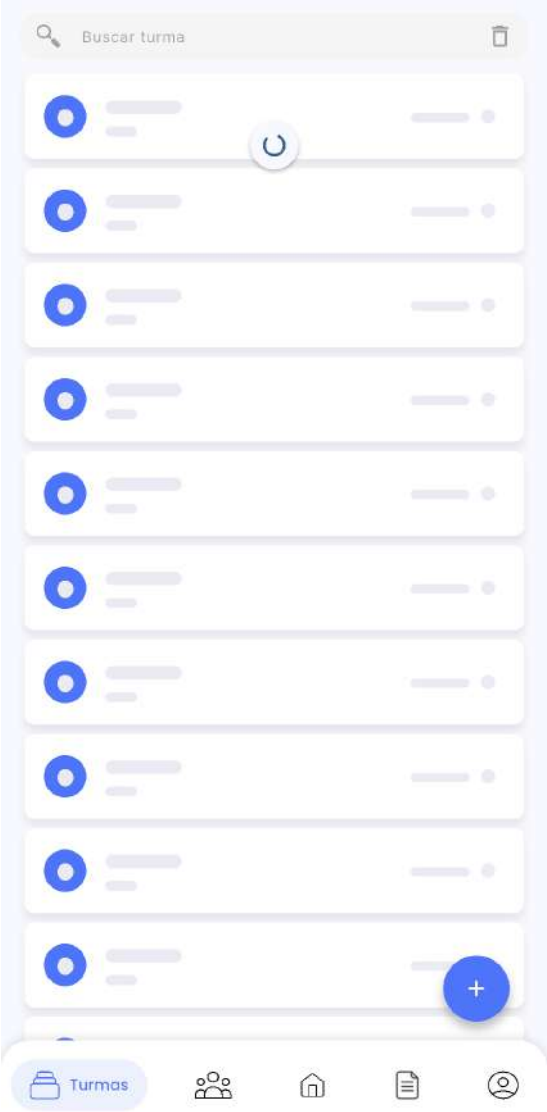
Fonte: Autor

Figura 12 – Variações e personalização de containers

Fonte: Autor

Além disso, técnicas como *Skeleton Screens* (RNF03, RNF12) (telas de carregamento esqueléticas) foram aplicadas para melhorar a percepção de desempenho durante o carregamento de dados.

Figura 13 – Exemplo de Skeleton Screen na aplicação



Fonte: Autor

Diferente de um *loading spinner*, que não fornece contexto visual sobre o que está sendo processado, o Skeleton Screen exibe a estrutura da interface antes dos dados estarem completamente carregados, criando uma percepção de maior velocidade e reduzindo a ansiedade do usuário. No aplicativo proposto, essa técnica está sendo utilizada em seções como o carregamento de turmas, exibição de notas e estatísticas acadêmicas, garantindo que tanto professores quanto alunos tenham acesso rápido às informações essenciais sem interrupções ou sensação de espera prolongada.

5.4.2 Fluxo de Navegação

O fluxo de navegação foi projetado para ser intuitivo e eficiente (RNF03), com telas específicas para professores e alunos/responsáveis (RNF05). A estrutura de navegação é baseada em *bottom bars* (barras inferiores), que permitem o acesso rápido às funcionalidades principais (RNF04).

5.4.2.1 Fluxo do Professor

- **Onboarding → Login (RF02, RNF05) → Tela Principal (RF09)**:
  - A tela principal do professor exibe turmas recentes (RF09), uma mensagem de boas-vindas e uma *bottom bar* para navegação entre as telas de turmas, alunos, tela principal e perfil.
- **Tela de Turmas (RF07)**:
  - Lista de turmas cadastradas (RF09), com opção de pesquisa e um *Floating Action Button* (FAB) para adicionar novas turmas (RF07).
- **Detalhes da Turma (RF15, RF20)**:
  - Dividida em quatro abas: Presença (RF15, RF17), Anotações, Avaliações (RF20) e Configurações (RF08).
  - Cada aba possui um FAB para ações específicas, como registrar presença (RF15), adicionar anotações ou lançar avaliações (RF20).
- **Tela de Alunos (RF11)**:
  - Lista de alunos cadastrados (RF13), com opção de pesquisa e um FAB para adicionar novos alunos (RF11).
- **Perfil do Professor (RF02, RNF06)**:
  - Permite a atualização de dados como usuário, e-mail e senha (RF02).

5.4.2.2 Fluxo do Aluno/Responsável

- **Onboarding → Login (RF06, RNF05) → Tela Principal (RF25)**:
  - A tela principal do aluno exibe gráficos de desempenho (RF26), indicadores acadêmicos (média (RF22), faltas (RF18), menor e maior nota) e uma *bottom bar* para navegação entre as telas de turmas, frequência por QR Code, tela principal e perfil.
- **Tela de Turmas (RF25)**:

  - Lista de turmas vinculadas ao aluno, com detalhes sobre notas (RF22) e frequência (RF18).

- **Tela de Frequência por QR Code (RF16)**:
  - Permite o registro de presença através da leitura de um QR Code gerado pelo professor (RF16).

- **Perfil do Aluno (RF03, RNF08)**:
  - Permite a alteração de senha (RF03).

5.4.3 Telas do Sistema

As telas do sistema foram projetadas para atender às necessidades específicas de cada tipo de usuário (RNF05), com foco na usabilidade (RNF03) e na clareza das informações.

- **Onboarding (RNF05)**: tela inicial com mensagem de boas-vindas e opções para selecionar o papel (professor ou aluno/responsável).

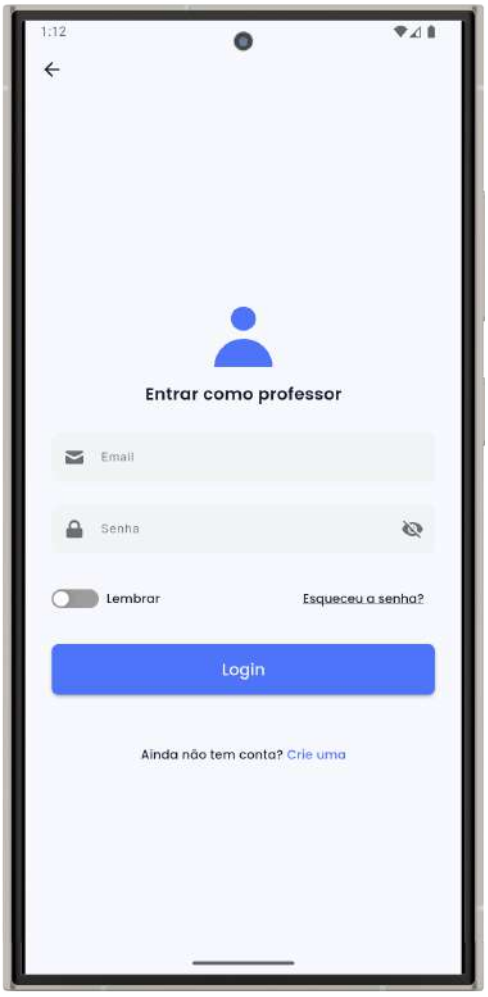
Figura 14 – Tela de boas vindas

Fonte: Autor

5.4.3.1 Telas do Professor

- **Login (RF02), Cadastro (RF01) e Recuperação de Senha (RF03)**:
  - Login: Tela simples com campos para e-mail e senha (RNF05)
  - Cadastro: Formulário contendo os campos: usuario, e-mail, e senha (RF01)
  - Recuperação de Senha: Tela com campo de e-mail, possibilidade a recuperação da senha (RF03)

Figura 15 – Tela de login do professor



Fonte: Autor

Figura 16 – Tela de cadastro do professor

Fonte: Autor

Figura 17 – Tela de recuperação de senha

Fonte: Autor

- **Tela Principal (RF09)**: exibe turmas recentes, uma mensagem de boas-vindas e uma *bottom bar* para navegação.

Figura 18 – Tela principal do professor

Fonte: Autor

- **Perfil (RF02, RNF06)**: exibe os dados do perfil do professor, permitindo sua edição.

Figura 19 – Tela de perfil do professor

Fonte: Autor

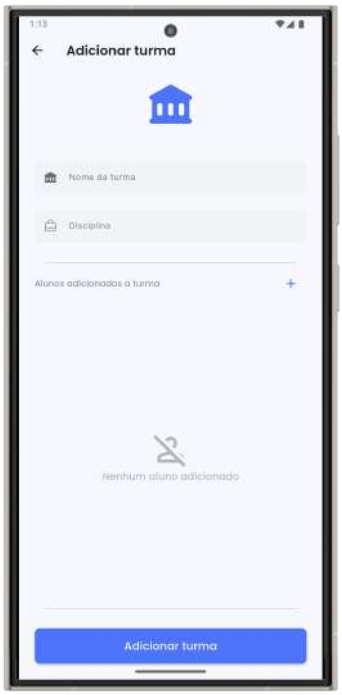
- **Tela de turmas do professor (RF07, RF09)**: exibe as turmas cadastradas

Figura 20 – Tela de turmas

Fonte: Autor

- **Cadastro de turmas (RF07)**: formulário que realiza a criação de uma turma, contém campos como: nome, disciplina e uma lista de alunos vinculados.

Figura 21 – Tela de cadastro de turmas



Fonte: Autor

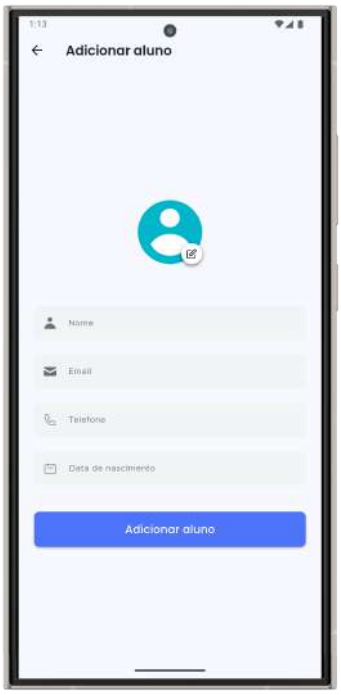
- **Tela de alunos (RF11, RF13)**: exibe os alunos cadastrados.

Figura 22 – Tela de alunos

Fonte: Autor

- **Cadastro de alunos (RF11)**: formulário que realiza o cadastro de um aluno, contém campos como: nome, e-mail, telefone e data de nascimento.

Figura 23 – Tela de cadastro de aluno



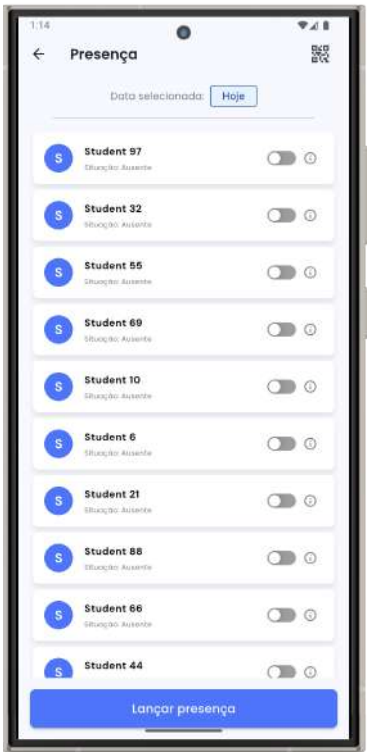
Fonte: Autor

- **Tela de Detalhes da Turma (RF15, RF17, RF20)**: dividida em abas para presença (RF15), anotações, avaliações (RF20) e configurações.

Figura 24 – Tela de presença (RF15)

Fonte: Autor

Figura 25 – Tela de registro de presença (RF15)



Fonte: Autor

Figura 26 – Seleção de data da presença (RF17)

Fonte: Autor

Figura 27 – Tela de geração do QR Code para frequência (RF15)

Fonte: Autor

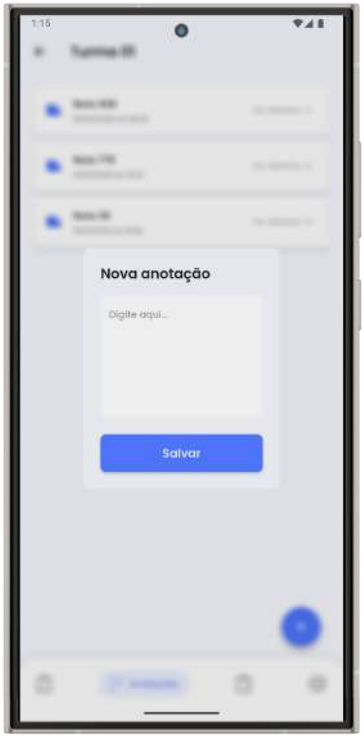
Figura 28 – Justificativa de falta (RF17)

Fonte: Autor

Figura 29 – Tela de anotações

Fonte: Autor

Figura 30 – Registro de anotação

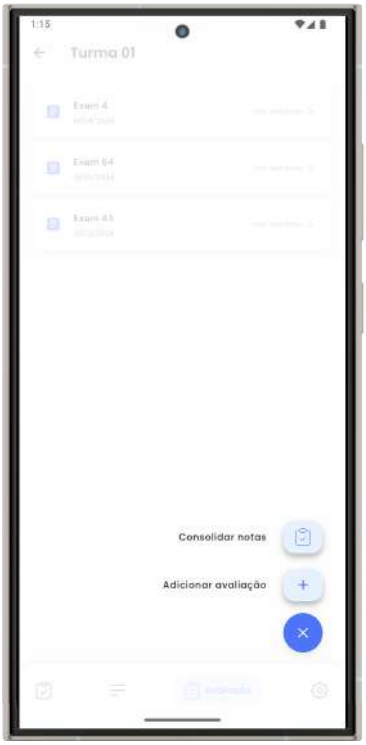


Fonte: Autor

Figura 31 – Tela de avaliações (RF20)

Fonte: Autor

Figura 32 – Ações do Floating Action Button da tela de avaliações (RF20)



Fonte: Autor

Figura 33 – Registro de avaliações (RF20)

Fonte: Autor

Figura 34 – Consolidação das notas (RF22)

Fonte: Autor

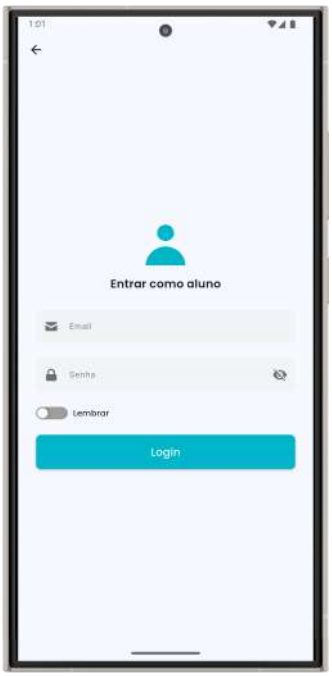
Figura 35 – Configuração da turma (RF08)

Fonte: Autor

5.4.3.2 Telas do Aluno

- **Login (RF06)**: tela simples com campos para e-mail e senha.

Figura 36 – Tela de login do aluno



Fonte: Autor

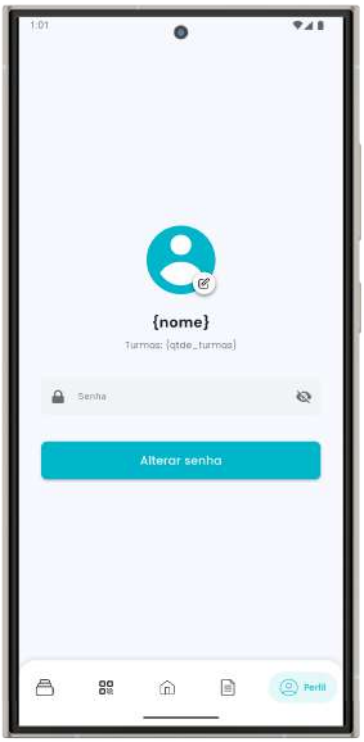
- **Tela Principal (RF25, RF26)**: exibe gráficos de desempenho, indicadores acadêmicos e *bottom bar* para navegação.

Figura 37 – Tela principal do aluno

Fonte: Autor

- **Perfil (RF03, RNF08)**: exibe apenas o campo de senha, passível de edição.

Figura 38 – Tela de perfil do aluno



Fonte: Autor

- **Frequência por QR Code (RF16)**: exibe a imagem capturada pela câmera do dispositivo para a captura do código QR gerado pelo professor.

Figura 39 – Tela de leitura do QR Code pelo aluno

Fonte: Autor

- **Tela de Detalhes da Turma (RF22, RF18)**: Mostra notas e frequência do aluno na turma selecionada.

Figura 40 – Tela de notas do aluno

Fonte: Autor

Figura 41 – Tela de frequência do aluno

Fonte: Autor

5.4.4 Feedback Visual e Acessibilidade

Para garantir uma experiência positiva (RNF03), o aplicativo fornece feedback visual imediato através de:

- **Mensagens de Confirmação**: Após ações como cadastro de turmas (RF07) ou lançamento de notas (RF20)
- **Snackbars**: Para notificações de sucesso ou erro (RNF03)
- **Indicadores de Carregamento (RNF03, RNF12)**: Utilizando *Skeleton Screens* durante o carregamento de dados

A acessibilidade foi priorizada com (RNF03, RNF04):

- Cores contrastantes para melhor visibilidade
- Tamanho de fonte adequado para leitura
- Ícones claros e intuitivos

6 RESULTADOS ESPERADOS

6.1 Impacto na Gestão Acadêmica

O aplicativo proposto tem como objetivo principal otimizar a gestão de turmas e o acompanhamento acadêmico, trazendo benefícios significativos para professores, alunos e responsáveis. A seguir, são descritos os resultados esperados em relação a cada grupo de usuários:

6.1.1 Para Professores

- **Organização Eficiente**: Com a centralização das informações em uma única plataforma, os professores poderão gerenciar turmas, alunos, notas e frequência de forma mais organizada e eficiente.
- **Redução de Tempo**: O uso de funcionalidades como o registro de presença via QR Code e o lançamento automatizado de notas reduzirá o tempo gasto com tarefas repetitivas, permitindo que os professores se concentrem mais no ensino.

6.1.2 Para Alunos e Responsáveis

- **Acompanhamento Contínuo**: Os alunos e responsáveis terão acesso fácil e rápido a informações como notas, frequência e indicadores de desempenho, permitindo um acompanhamento mais próximo do progresso acadêmico.
- **Identificação de Dificuldades**: Com gráficos e indicadores claros, será possível identificar dificuldades de aprendizado de forma precoce, facilitando a tomada de medidas corretivas, como reforço escolar ou ajustes na metodologia de ensino.
- **Engajamento**: O uso de notificações push e uma interface intuitiva aumentará o engajamento dos alunos e responsáveis, garantindo que informações importantes sejam recebidas e visualizadas em tempo hábil.

6.2 Benefícios Técnicos

Além dos benefícios para os usuários, o aplicativo também traz vantagens técnicas que contribuem para sua viabilidade e sustentabilidade:

6.2.1 Arquitetura Modular e Escalável

A adoção da *Clean Architecture* e do padrão de desenvolvimento *Clean Code* garante que o sistema seja modular, de fácil manutenção e escalável. Isso permitirá

a adição de novas funcionalidades no futuro, como integração com outras plataformas educacionais ou suporte a diferentes tipos de avaliações.

6.2.2 Usabilidade e Acessibilidade

O design centrado no usuário, com foco em simplicidade e acessibilidade, garantirá que o aplicativo seja fácil de usar por professores e alunos, independentemente de seu nível de familiaridade com tecnologia. A utilização de cores contrastantes, fontes legíveis e ícones intuitivos contribuirá para uma experiência de usuário positiva.

6.2.3 Segurança e Conformidade

A implementação de mecanismos de segurança, como criptografia de dados e autenticação de usuários, garantirá a proteção das informações acadêmicas. Além disso, o aplicativo estará em conformidade com a Lei Geral de Proteção de Dados (LGPD), reforçando a confiança dos usuários.

6.3 Impacto Social e Educacional

O aplicativo proposto também tem o potencial de gerar impactos positivos no contexto social e educacional:

6.3.1 Democratização do Acesso à Educação

Ao ser desenvolvido como um projeto *open-source*, o aplicativo poderá ser adaptado e utilizado por diferentes instituições de ensino, incluindo escolas públicas e privadas, contribuindo para a democratização do acesso a ferramentas de gestão acadêmica de qualidade.

6.3.2 Transparência e Colaboração

A abordagem *open-source* também promoverá a transparência e a colaboração, permitindo que desenvolvedores e educadores contribuam com melhorias e personalizações, tornando o aplicativo mais alinhado às necessidades reais dos usuários.

6.4 Conclusão dos Resultados Esperados

Em resumo, o aplicativo proposto tem o potencial de revolucionar a gestão de turmas e o acompanhamento acadêmico, trazendo benefícios práticos para professores, alunos e responsáveis, além de vantagens técnicas e impactos sociais positivos. A combinação de uma arquitetura robusta, um design centrado no usuário e uma abordagem *open-source* garante que o aplicativo seja uma solução viável, sustentável e de alto impacto no contexto educacional.

7 CONCLUSÃO

Este trabalho apresentou uma proposta de aplicativo para gestão de turmas e acompanhamento acadêmico, desenvolvido com o objetivo de otimizar a organização de professores e facilitar o acesso de alunos e responsáveis às informações acadêmicas. A solução proposta foi projetada com base em princípios de design centrado no usuário, arquitetura limpa e boas práticas de desenvolvimento, garantindo uma experiência intuitiva, eficiente e escalável.

O aplicativo foi concebido para atender às necessidades de professores independentes e instituições de ensino, oferecendo funcionalidades como cadastro de turmas, registro de presença via QR Code, lançamento de notas e acompanhamento acadêmico. A interface foi cuidadosamente planejada para ser acessível e de fácil uso, com uma paleta de cores que transmite confiança e tranquilidade, além de ícones e fontes que garantem clareza e legibilidade.

A arquitetura do sistema, baseada nos princípios da *Clean Architecture*, foi projetada para ser modular e de fácil manutenção, permitindo a evolução do aplicativo com a adição de novas funcionalidades no futuro. A escolha de tecnologias modernas, como Flutter para o frontend, NestJS para o backend e MongoDB para o banco de dados, garante um desempenho robusto e a capacidade de escalar conforme a demanda.

Além dos benefícios práticos para professores, alunos e responsáveis, o aplicativo proposto também traz impactos positivos no contexto educacional. A transparência no acompanhamento acadêmico, a redução do tempo gasto com tarefas administrativas e a facilitação da comunicação entre os envolvidos no processo educacional são alguns dos resultados esperados. A abordagem *open-source* adotada no projeto promove a colaboração e a democratização do acesso a ferramentas de gestão acadêmica, ampliando seu potencial de impacto.

Em síntese, a proposta deste aplicativo representa um avanço significativo na gestão de turmas e no acompanhamento acadêmico, oferecendo uma solução moderna, eficiente e acessível. Espera-se que, com a implementação das funcionalidades propostas, o aplicativo se torne uma ferramenta valiosa para professores, alunos e responsáveis, contribuindo para a melhoria da qualidade do ensino e da aprendizagem.

8 TRABALHOS FUTUROS

A proposta do aplicativo apresentada neste trabalho representa um ponto de partida para uma solução robusta e escalável de gestão de turmas e acompanhamento acadêmico. No entanto, há diversas oportunidades de melhoria e expansão que podem ser exploradas em trabalhos futuros, visando aprimorar a experiência dos usuários e adicionar novas funcionalidades. A seguir, são descritas algumas das possíveis melhorias que podem ser implementadas em versões futuras do aplicativo.

8.1 Melhorias na Comunicação

Uma das principais demandas em ambientes educacionais é a facilitação da comunicação entre professores, alunos e responsáveis. Para atender a essa necessidade, as seguintes funcionalidades podem ser adicionadas:

8.1.1 Chat Integrado

A implementação de um sistema de chat direto entre professores e alunos/responsáveis permitiria uma comunicação mais ágil e eficiente. Esse recurso poderia ser utilizado para tirar dúvidas, enviar lembretes sobre prazos de atividades ou discutir o desempenho acadêmico. O chat poderia incluir funcionalidades como:

- Envio de mensagens de texto, imagens e arquivos.
- Notificações em tempo real para novas mensagens.
- Histórico de conversas organizado por turma ou aluno.

8.1.2 Tela de Notificações

Uma tela exclusiva para notificações poderia centralizar todas as informações importantes, como avisos sobre prazos de atividades, datas de provas, eventos escolares e mensagens do professor. Essa tela poderia incluir:

- Filtros para organizar notificações por tipo (provas, trabalhos, eventos).
- Opção de marcar notificações como lidas ou não lidas.
- Integração com notificações push para alertas em tempo real.

8.2 Expansão das Funcionalidades para Professores

Para auxiliar os professores no planejamento e execução de suas atividades, novas funcionalidades podem ser adicionadas ao aplicativo:

8.2.1 Planejamento de Aulas

Uma seção dedicada ao planejamento de aulas permitiria que os professores organizem suas atividades de forma mais estruturada. Essa funcionalidade poderia incluir:

- Criação de planos de aula com objetivos, conteúdos e atividades.
- Agendamento de aulas e distribuição de materiais de apoio.
- Integração com o calendário escolar para visualização de datas importantes.

8.2.2 Diversificação de Métodos de Avaliação

Além da atual opção de avaliação por provas, o aplicativo poderia incluir outros métodos de avaliação, como:

- **Trabalhos e Projetos**: Permite o lançamento de notas para trabalhos individuais ou em grupo.
- **Participação em Aula**: Avaliação baseada na participação e engajamento dos alunos durante as aulas.
- **Avaliações Contínuas**: Sistema de avaliação formativa, com feedbacks regulares sobre o desempenho do aluno.

8.3 Melhorias na Experiência do Aluno

Para tornar a experiência do aluno mais completa e engajadora, as seguintes melhorias podem ser consideradas:

8.3.1 Gamificação

A introdução de elementos de gamificação pode aumentar o engajamento dos alunos, transformando o aprendizado em uma experiência mais dinâmica e motivadora. Algumas ideias incluem:

- Sistema de pontuação e recompensas por conclusão de atividades.
- Desafios e missões relacionadas ao conteúdo das aulas.
- Ranking de desempenho (opcional) para estimular a competição saudável.

8.3.2 Acompanhamento Personalizado

Uma funcionalidade de acompanhamento personalizado poderia fornecer recomendações e sugestões de estudo com base no desempenho do aluno. Por exemplo:

- Sugestão de materiais de estudo para tópicos com dificuldades.
- Alertas sobre prazos de atividades e provas.
- Relatórios periódicos de desempenho para alunos e responsáveis.

8.4 Integração com Outras Plataformas

Para ampliar a utilidade do aplicativo, integrações com outras plataformas e ferramentas educacionais podem ser exploradas:

8.4.1 Integração com Google Classroom e Moodle

A integração com plataformas populares de gestão de aprendizagem, como Google Classroom e Moodle, permitiria a sincronização automática de turmas, atividades e notas, reduzindo a necessidade de inserção manual de dados.

8.4.2 API para Desenvolvedores

A criação de uma API aberta permitiria que desenvolvedores e instituições de ensino criassem integrações personalizadas, adaptando o aplicativo às suas necessidades específicas.

8.5 Expansão para Outros Níveis de Ensino

Embora o aplicativo tenha sido projetado inicialmente para o ensino básico e superior, ele pode ser adaptado para outros níveis de ensino, como:

8.5.1 Educação Infantil

Adaptações para a educação infantil poderiam incluir funcionalidades como:

- Registro de atividades lúdicas e desenvolvimento cognitivo.
- Acompanhamento de habilidades sociais e emocionais.
- Comunicação direta com os pais ou responsáveis.

8.5.2 Cursos Livres e Profissionalizantes

Para cursos livres e profissionalizantes, o aplicativo poderia incluir:

- Gestão de certificados e diplomas.
- Avaliação de competências técnicas.
- Integração com plataformas de empregabilidade.

REFERÊNCIAS

ALENCAR, A. de S. et al. O moodle como ferramenta didática. **Portal de Periódicos da Faculdade de Letras - UFMG**, 2011. Disponível em: <http://www.periodicos.letras.ufmg.br/index.php/ueadsl/article/viewFile/2919/2878>.

AMANKWAH-AMOAH, J. et al. Covid-19 and digitalization: The great acceleration. **Journal of Business Research**, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0148296321005725>.

BONIFáCIO, B. et al. Avaliação de usabilidade de aplicativos móveis: Um estudo de caso sobre um aplicativo de ensino. In: **Conferências Ibero-Americanas WWW/Internet e Computação Aplicada**. IADIS, 2014. p. 11–18. Disponível em: <https://www.researchgate.net/publication/280729293_AVALIACAO_DE_USABILIDADE_DE_APLICATIVOS_MOVEIS_UM_ESTUDO_DE_CASO_SOBRE_UM_APLICATIVO_DE_ENSINO>.

BUCZYńSKI, S. **Implementing the Clean Architecture**. [S.l.]: Leanpub, 2020.

BUENO, C. E. de O. **Desenvolvimento de um aplicativo utilizando o framework flutter e arquitetura limpa**. TCC — PUC Goiás, 2021. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/handle/123456789/1861>.

MARTIN, R. C. **Clean Code: A Handbook of Agile Software Craftsmanship**. [S.l.]: Prentice Hall PTR, 2008.

MARTIN, R. C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. [S.l.]: Pearson, 2017.

OLIVEIRA, H. P. **Desenvolvimento de uma Ferramenta Web para Suporte ao Design Visual de Interface de Usuário de Aplicativos Móveis**. Trabalho de Conclusão de Curso — Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2024. Disponível em: <https://repositorio.ufsc.br/handle/123456789/255904>.

OLORUNTOBA, S. **SOLID: The First 5 Principles of Object Oriented Design**. 2020. Disponível em: <https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>. Acesso em: 18 Jan. 2025.

PIRES, E. **SOLID – Teoria e Prática**. 2015. Disponível em: <https://www.eduardopires.net.br/2015/01/solid-teoria-e-pratica/>. Acesso em: 18 Jan. 2025.

SOUSA, E. P. **Gestão Educacional e Inovação: o uso das plataformas digitais na escola**. Dissertação (Mestrado) — Universidade Católica Portuguesa, Braga, Portugal, 2020.

VALENTE, M. T. **Construindo Sistemas com uma Arquitetura Limpa**. 2020. Disponível em: <https://engsoftmoderna.info/artigos/arquitetura-limpa.html>. Acesso em: 18 Jan. 2025.

VIEIRA, M. H. de A.; SILVA, F. J. F. da. Uso de aplicativos educacionais em escolas públicas de ensino fundamental e médio. **Anais do Congresso Nacional de Educação - CONEDU**, p. 1–12, 2020. Disponível em: <https://editorarealize.com.br/editora/anais/conedu/2020/TRABALHO_EV140_MD1_SA19_ID7773_01102020121824.pdf>.