



Matheus de Oliveira Alves

**DESENVOLVIMENTO DE UM SISTEMA PARA REPORTAR
INCIDENTES DE INFRAESTRUTURA EM CAMPI
UNIVERSITÁRIOS**

Monografia de Graduação

Ouro Preto, 2025

Matheus de Oliveira Alves

DESENVOLVIMENTO DE UM SISTEMA PARA REPORTAR INCIDENTES DE INFRAESTRUTURA EM CAMPI UNIVERSITÁRIOS

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro(a) de Controle e Automação.

Universidade Federal de Ouro Preto

Orientador: Prof. Danny Augusto Vieira Tonidandel.

Coorientador: Profa. Valéria de Carvalho Santos.

Ouro Preto

2025



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E
AUTOMACAO



FOLHA DE APROVAÇÃO

Matheus de Oliveira Alves

Desenvolvimento de um Sistema para Reportar Incidentes de Infraestrutura em Campi Universitários

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Engenharia de Controle e Automação

Aprovado em 20 de março de 2025.

Integrantes da banca

[Doutor] - Danny Augusto Vieira Tonidandel - Orientador (Universidade Federal de Ouro Preto)

[Doutora] - Valéria de Carvalho Santos - Coorientadora (Universidade Federal de Ouro Preto)

[Doutora] - Adrielle de Carvalho Santana - (Universidade Federal de Ouro Preto)

[Doutora] - Luciana Gomes Castanheira - (Universidade Federal de Ouro Preto)

Danny Augusto Vieira Tonidandel, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 20/03/2025.



Documento assinado eletronicamente por **Danny Augusto Vieira Tonidandel, PROFESSOR DE MAGISTERIO SUPERIOR**, em 20/03/2025, às 18:16, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0875118** e o código CRC **CE049946**.

Agradecimentos

Primeiramente, gostaria de expressar minha mais sincera gratidão aos meus pais, **Nicanor** e **Edicilma**, por todo o amor, dedicação e incentivo que me permitiram chegar até aqui. Agradeço também à minha irmã **Jessica**, que sempre esteve ao meu lado, oferecendo conselhos, apoio e motivação ao longo desta caminhada.

Manifesto igualmente meu reconhecimento a todo o corpo docente da **UFOP**, em especial ao **DECAT**, pela excelência no ensino e por proporcionarem um ambiente acadêmico que contribuiu não apenas para meu crescimento intelectual, mas também pessoal. Aos meus amigos do curso de **Engenharia de Controle e Automação**, fica meu agradecimento pela parceria, pelas conversas inspiradoras e pela convivência que tornou essa trajetória muito mais enriquecedora.

Não poderia deixar de agradecer ao meu orientador, **Danny**, e à minha coorientadora, **Valeria**, pela orientação cuidadosa, confiança e paciência durante todo o desenvolvimento deste trabalho. Finalmente, estendo meu abraço fraterno aos meus irmãos da **República Consulado**, que estiveram presentes em todos os momentos, oferecendo camaradagem e tornando cada etapa dessa jornada mais leve e significativa.

A todos, meu mais sincero e profundo obrigado.

Resumo

A comunicação eficiente no ambiente universitário é fundamental para o bom funcionamento das atividades acadêmicas e administrativas, especialmente no que se refere à identificação e resolução de problemas estruturais e funcionais que impactam diretamente o dia a dia dos usuários da instituição. Este trabalho tem como objetivo o desenvolvimento de um sistema voltado à Universidade Federal de Ouro Preto (UFOP), onde estudantes, professores e a equipe administrativa possam reportar irregularidades relacionadas à infraestrutura, permitindo que todos os membros da comunidade acadêmica participem ativamente na melhoria do campus por meio da criação de reportes sobre problemas como equipamentos danificados, falhas na iluminação e questões relacionadas à acessibilidade, entre outros. Além de registrar as ocorrências, o sistema possibilita que os usuários acompanhem o status dos problemas relatados em tempo real, promovendo maior transparência no processo de resolução.

Palavras-chave: aplicativo, infraestrutura, universidade, comunicação, manutenção.

Abstract

Efficient communication in the university environment is essential for the proper functioning of academic and administrative activities, especially in identifying and resolving structural and functional issues that directly impact the daily lives of the institution's users. This work aims to develop a system for the Federal University of Ouro Preto (UFOP), where students, professors, and the administrative team can report infrastructure-related irregularities, allowing all members of the academic community to actively participate in campus improvements by creating reports on issues such as damaged equipment, lighting failures, and accessibility concerns, among others. In addition to recording occurrences, the system enables users to track the status of reported problems in real time, promoting greater transparency in the resolution process.

Key-words: application, infrastructure, university, communication, maintenance.

Lista de ilustrações

Figura 1 – Representação da Arquitetura Limpa. Fonte: Blog Robert C. Martin . . .	19
Figura 2 – Comparação de Popularidade entre Flutter, React Native, Xamarin, MAUI e Ionic. Fonte: Google Trends (GOOGLE TRENDS, 2024).	26
Figura 3 – Quadro Kanban no Trello com as tarefas do projeto <i>Reporta UFOP</i> . . .	28
Figura 4 – Protótipo do sistema dos estudantes: Tela de Login e Cadastro.	30
Figura 5 – Protótipo do sistema dos estudantes: Tela Home, Detalhes do Reporte e Reportes Feitos pelo Usuário.	31
Figura 6 – Protótipo do sistema dos administradores: Tela de Login e Home. . . .	31
Figura 7 – Protótipo do sistema dos administradores: Menu e Gerenciamento de Reportes.	31
Figura 8 – Protótipo do sistema dos administradores: Relatório.	32
Figura 9 – Estrutura inicial do projeto e configuração no Git.	33
Figura 10 – Protótipos das telas de login e configuração de roteamento.	34
Figura 11 – Configurações no Firebase e desenvolvimento das telas de cadastro. . .	35
Figura 12 – Tela Inicial e Configuração do Firebase Storage	36
Figura 13 – Detalhes do Reporte e Banco de Dados	37
Figura 14 – Desenvolvimento da Tela “Meus Reportes”	38
Figura 15 – Desenvolvimento da Tela Home	39
Figura 16 – Tela de Gerenciamento de Reportes	40
Figura 17 – Funcionalidade de Relatórios	40
Figura 18 – Funcionalidade de Logout	41
Figura 19 – Tela de login da aplicação móvel para estudantes.	44
Figura 20 – Tela de login da aplicação móvel para administradores.	45
Figura 21 – Tela de Cadastro da Aplicação Móvel para Estudantes.	46
Figura 22 – Tela Inicial (Home) da Aplicação Móvel para Estudantes.	47
Figura 23 – Fluxo de captura de imagem para reporte.	48
Figura 24 – Tela de Detalhes do Reporte.	49
Figura 25 – Tela de Meus Reportes.	50
Figura 26 – Tela Central dos Reportes.	52
Figura 27 – Tela Central dos Reportes.	53
Figura 28 – Tela de Gerenciar Reporte.	54
Figura 29 – Tela de Relatório da Aplicação para Administradores.	56
Figura 30 – Repositório do projeto Reporta UFOP no GitHub.	57
Figura 31 – Licença MIT atribuída ao repositório.	58

Lista de tabelas

Tabela 1	– Caso de uso: Criação de Conta e <i>Login</i> (Estudantes e Professores)	21
Tabela 2	– Caso de uso: Submissão de Reportes (Estudantes e Professores)	22
Tabela 3	– Caso de uso: Acompanhamento de Reportes (Estudantes e Professores)	22
Tabela 4	– Caso de uso: <i>logout</i> (Estudantes e Professores)	23
Tabela 5	– Caso de uso: Gestão de Acessos (Administradores)	23
Tabela 6	– Caso de uso: Visualização e Filtragem de Reportes (Administradores) .	24
Tabela 7	– Caso de uso: Edição e Resolução de Reportes (Administradores)	24
Tabela 8	– Caso de uso: <i>Logout</i> (Administrador)	25
Tabela 9	– Bibliotecas Utilizadas no Desenvolvimento do Projeto	29

Lista de Siglas

UFOP Universidade Federal de Ouro Preto

Web World Wide Web

UI User Interface (Interface do Usuário)

Kit Conjunto de Ferramentas

iOS Sistema Operacional da Apple

IOS iPhone Operating System (Sistema Operacional do iPhone)

MacOS Sistema Operacional para Computadores da Apple

APK Android Package (Pacote Android)

MIT Massachusetts Institute of Technology (Instituto de Tecnologia de Massachusetts)

Commit Registro de Mudança no Repositório

IDE Integrated Development Environment (Ambiente de Desenvolvimento Integrado)

BaaS Backend as a Service (Backend como Serviço)

URL Uniform Resource Locator (Localizador Uniforme de Recursos)

Git Sistema de Controle de Versão Distribuído

Sumário

	Lista de Siglas	8
1	INTRODUÇÃO	11
1.1	Justificativa e Relevância	11
1.2	Objetivo Geral	11
1.2.1	Objetivos Específicos	11
1.3	Organização e Estrutura	12
2	REFERENCIAL TEÓRICO	13
2.1	Impactos da Infraestrutura no Âmbito da Educação	13
2.1.1	A Importância das Tecnologias da Informação e Comunicação na Gestão Pública	13
2.2	Base Tecnológica e Metodológica do Projeto	14
2.2.1	<i>Software</i>	15
2.2.2	Método Kanban	15
2.2.2.1	Coluna: A Fazer	15
2.2.2.2	Coluna: Em Andamento	16
2.2.2.3	Coluna: Concluído	16
2.2.3	Código Limpo	16
2.2.4	Desenvolvimento multiplataforma	17
2.2.5	<i>Software</i> Aberto	17
2.2.6	Sistema Android	18
2.2.7	Arquitetura Limpa	18
2.2.8	Controle de Versão	20
3	DESENVOLVIMENTO	21
3.1	Casos de Uso	21
3.2	Ferramentas Utilizadas	25
3.2.1	Dart	25
3.2.2	Flutter	25
3.2.3	Visual Studio Code	26
3.2.4	Firebase	26
3.2.5	Git	27
3.2.6	Trello	27
3.2.7	Bibliotecas	28
3.2.8	Bibliotecas Utilizadas	28

3.3	Prototipagem	29
3.3.0.1	Funcionalidades Destinadas a Estudantes	30
3.3.0.2	Funcionalidades destinadas a Administradores	31
3.3.1	Ciclo de Desenvolvimento do Projeto	32
3.3.1.1	Etapa 1: Configuração Inicial	32
3.3.1.2	Etapa 2: Desenvolvimento das Telas de Login e Roteamento	33
3.3.1.3	Etapa 3: Telas de Cadastro e Configurações no Firebase	34
3.3.1.4	Etapa 4: Desenvolvimento da Tela Inicial e Funcionalidades para Estudantes	35
3.3.1.5	Etapa 5: Configuração do Serviço de Armazenamento (Firebase Storage)	35
3.3.1.6	Etapa 6: Tela de Detalhes do Reporte e Integração com Firebase Storage	36
3.3.1.7	Etapa 7: Configuração do Banco de Dados Firebase	37
3.3.1.8	Etapa 8: Tela “Meus Reportes”	38
3.3.1.9	Etapa 9: Desenvolvimento da Tela Home e Funcionalidades dos Administradores	38
3.3.1.10	Etapa 10: Implementação da Tela de Gerenciamento de Reportes	39
3.3.1.10.1	Funcionalidades Implementadas:	39
3.3.1.10.2	Integração com Google Maps:	39
3.3.1.11	Etapa 11: Implementação da Funcionalidade de Relatórios	40
3.3.1.12	Etapa 12: Implementação da Funcionalidade de Logout	41
3.3.1.12.1	Funcionalidades Implementadas:	41
3.3.1.13	Etapa 13: Testes e Finalização do Projeto	41
4	RESULTADOS	43
4.1	Arquitetura do Sistema	43
4.2	Funcionalidade de Login	43
4.2.1	Login para Estudantes	43
4.2.2	Login para Administradores	44
4.2.3	Cadastro	45
4.2.4	Home - Estudantes	46
4.2.5	Adicionar Reporte - Estudantes	47
4.2.6	Meus Reportes - Estudantes	49
4.2.7	Central dos Reportes - Administradores	51
4.2.8	Gerenciar Reporte - Administradores	53
4.2.9	Relatório - Administradores	55
4.3	Repositório	56
4.3.1	Benefícios do Sistema para as Universidades	58
5	CONSIDERAÇÕES FINAIS	60
6	TRABALHOS FUTUROS	61
	Referências	62

1 Introdução

1.1 Justificativa e Relevância

A Universidade Federal de Ouro Preto (UFOP), conforme indicado no Plano de Integridade de 2023 ([UNIVERSIDADE FEDERAL DE OURO PRETO, 2023](#)), possui aproximadamente 12.016 alunos de graduação. A falta de um canal eficiente para reportar e acompanhar problemas estruturais no campus compromete diretamente a experiência acadêmica dos estudantes. Estudos recentes indicam que diversas universidades federais brasileiras têm desenvolvido aplicativos móveis para aprimorar a gestão acadêmica e administrativa, facilitando a comunicação entre a instituição e a comunidade acadêmica ([DANTAS; MOREIRA; SOUZA, 2023](#)).

Por exemplo, foram identificados aproximadamente 200 aplicativos desenvolvidos por universidades federais no Brasil, dos quais 28% são provenientes da região Sudeste, onde a UFOP está localizada ([DANTAS; MOREIRA; SOUZA, 2023](#)). Esses aplicativos têm se mostrado eficazes na melhoria de processos internos e na satisfação dos usuários. No entanto, o estudo também aponta que apenas 6,5% desses aplicativos possuem registro no Instituto Nacional de Propriedade Industrial (INPI), indicando uma oportunidade para a UFOP não apenas desenvolver, mas também formalizar e proteger sua solução tecnológica ([DANTAS; MOREIRA; SOUZA, 2023](#)).

Neste contexto, o desenvolvimento de um sistema que simplifique a comunicação de incidentes e facilite o acompanhamento dessas questões pode contribuir significativamente para a melhoria do cotidiano da comunidade universitária. Dessa forma, este trabalho justifica-se pela necessidade de criar uma solução que otimize essa comunicação e promova um ambiente acadêmico mais organizado e funcional.

1.2 Objetivo Geral

Este trabalho tem como objetivo desenvolver uma aplicação multiplataforma que funcione para sistemas Android e Web, destinada a discentes, docentes e administradores da UFOP, proporcionando uma ferramenta eficiente para facilitar a comunicação e o reporte de problemas relacionados à infraestrutura da universidade, garantindo que esses problemas sejam resolvidos de forma mais ágil e transparente.

1.2.1 Objetivos Específicos

- Desenvolver o sistema seguindo as melhores práticas de desenvolvimento de *software*.

- Publicar o *software* em um repositório de código aberto, permitindo que colaboradores contribuam com novas funcionalidades e melhorias.

1.3 Organização e Estrutura

O presente trabalho está organizado em seis capítulos, cada um com foco específico que contribui para a compreensão e desenvolvimento do projeto *Reporta UFOP*. A estrutura é delineada da seguinte forma:

- **Capítulo 1** - Este capítulo introduz o trabalho, apresentando a justificativa, relevância e os objetivos do projeto. Além disso, descreve a organização estrutural da monografia.
- **Capítulo 2** - Neste capítulo, são abordados os conceitos e fundamentos necessários para compreender o trabalho. A discussão inclui os impactos da infraestrutura no âmbito educacional e a importância das tecnologias da informação e comunicação na gestão pública. A análise técnica e metodológica das tecnologias e metodologias aplicadas no desenvolvimento do sistema também são detalhadas aqui.
- **Capítulo 3** - Este capítulo descreve o desenvolvimento do sistema, desde a definição dos requisitos até a implementação, utilizando as ferramentas e metodologias selecionadas para o projeto.
- **Capítulo 4** - Os resultados obtidos com a implementação do sistema são detalhados neste capítulo, destacando as funcionalidades implementadas.
- **Capítulo 5** - Contém as conclusões do trabalho, refletindo sobre o desenvolvimento e a implementação do sistema, bem como as principais aprendizagens e desafios encontrados.
- **Capítulo 6** - Propõe direções para a evolução e expansão do sistema.

2 Referencial Teórico

Neste capítulo, são apresentados os conceitos e fundamentos necessários para compreender o trabalho. A primeira parte aborda os impactos da infraestrutura no âmbito da educação, bem como a importância da tecnologia da informação e comunicação na gestão pública. A segunda parte é dedicada à análise técnica e metodológica, com o objetivo de detalhar e aprofundar o entendimento sobre as tecnologias e metodologias utilizadas na construção deste trabalho.

2.1 Impactos da Infraestrutura no Âmbito da Educação

Conforme destaca [Soares Neto, Jesus, Karino e Andrade \(2013\)](#) por meio de seus estudos sobre infraestrutura no ambiente educacional:

Promover a educação requer a garantia de um ambiente com condições para que a aprendizagem possa ocorrer. É importante proporcionar um ambiente físico, aqui denominado infraestrutura escolar, que estimule e viabilize o aprendizado, além de favorecer as interações humanas.

Dessa forma, é evidente como a qualidade da infraestrutura impacta diretamente o ambiente educacional, afetando tanto a experiência dos estudantes quanto a eficiência administrativa. No caso das universidades federais, como a Universidade Federal de Ouro Preto (UFOP), esses desafios se tornam ainda mais relevantes, considerando as características de seus campi e as demandas específicas de seus usuários.

Um exemplo prático dessa situação foi discutido na monografia de [Oliveira e Barbosa Neto \(2024\)](#), realizada por estudantes do curso de Engenharia de Produção da UFOP, campus João Monlevade. O estudo apontou que 60% dos alunos entrevistados se sentem inseguros devido à falta de iluminação no campus, evidenciando a importância de uma infraestrutura adequada para o bem-estar da comunidade acadêmica. Esse resultado reforça a necessidade de implementar ferramentas que permitam identificar e reportar problemas estruturais de maneira eficiente.

2.1.1 A Importância das Tecnologias da Informação e Comunicação na Gestão Pública

As Tecnologias da Informação e Comunicação (TICs) têm transformado significativamente a forma como indivíduos e instituições interagem, redefinindo processos e

ampliando as possibilidades de acesso à informação e comunicação. Segundo [Oliveira e Moura \(2016\)](#), as TICs englobam todos os meios técnicos utilizados para processar informações e facilitar a comunicação, sendo essenciais para atender às demandas de uma sociedade cada vez mais conectada.

Com a popularização de dispositivos como smartphones e computadores, aliada à crescente acessibilidade dessas tecnologias, as TICs têm se consolidado como ferramentas indispensáveis na modernização de diferentes tipos de gestão, sejam elas empresariais, pessoais ou governamentais. No âmbito da gestão pública, essas tecnologias têm um impacto ainda mais evidente, promovendo maior transparência, eficiência nos processos e melhoria na prestação de serviços.

No Brasil, a adoção de tecnologias digitais tem crescido exponencialmente. Uma pesquisa realizada pela [Fundação Getúlio Vargas \(2024\)](#) revelou que o país possui atualmente mais de 480 milhões de dispositivos digitais em uso. Esse número abrange smartphones, notebooks, tablets e computadores, evidenciando a forte penetração das tecnologias digitais na vida cotidiana e sua capacidade de potencializar soluções inovadoras.

Um estudo realizado por [Brognoli \(2018\)](#) no Instituto Federal de Santa Catarina (IFSC) – campus Criciúma – avaliou o impacto das TICs no instituto, com uma pesquisa abrangente envolvendo estudantes e funcionários da instituição. Os resultados mostraram altos índices de satisfação, com as notas 7 e 8 (em uma escala de 0 a 10) empatadas como as mais atribuídas pelos entrevistados. Entre as características mais destacadas, a agilidade nos processos foi apontada como um dos principais benefícios, demonstrando o potencial das TICs para otimizar fluxos de trabalho e melhorar a experiência administrativa.

Além disso, [Fernandes \(2009\)](#) destaca que “uma forma de acompanhar a gestão pública relacionada com a alocação de recursos e a forma como os bens e serviços são disponibilizados é através de indicadores de desempenho”. Isso reforça a importância de ferramentas tecnológicas que auxiliem na transparência e eficiência administrativa das universidades, o que está diretamente alinhado com os objetivos do *Reporta UFOP*.

2.2 Base Tecnológica e Metodológica do Projeto

Nesta seção, são apresentadas as tecnologias e metodologias utilizadas no desenvolvimento do projeto, detalhando como cada ferramenta contribuiu para a sua implementação e sucesso. O capítulo abrange desde *framework* de desenvolvimento e sistemas operacionais até padrões de arquitetura e práticas de controle de versão. Ao explorar essas tecnologias, busca-se demonstrar a fundamentação técnica que sustenta o projeto e sua aplicação prática no contexto do *Reporta UFOP*.

2.2.1 Software

As aplicações têm revolucionado a forma como os seres humanos interagem com o mundo digital. Com o crescente uso de smartphones, a criação e utilização de sistemas voltados tanto para dispositivos móveis quanto para computadores, tornaram-se cada vez mais populares. Segundo [Stair e Reynolds \(2002\)](#), um sistema pode ser definido como “um conjunto de elementos ou componentes que interagem para se atingir objetivos.

Essas definições reforçam a importância dos sistemas como base para o desenvolvimento de soluções tecnológicas que atendam às necessidades de usuários de dispositivos móveis e computadores, contribuindo para a eficiência e praticidade nas interações com o mundo digital. Este entendimento é complementado pelos estudos de [Araújo e Gouveia \(2016\)](#), que fazem uma revisão sobre os princípios da Teoria Geral dos Sistemas, destacando a interdependência e a interação como fundamentos para a construção de sistemas eficientes.

2.2.2 Método Kanban

O método Kanban é uma das principais metodologias utilizadas para gerenciamento de projetos, originário do Japão, cujo nome significa “cartão” ou “sinal visual”. Esta ferramenta permite a visualização clara e o controle detalhado dos processos, promovendo um fluxo contínuo de trabalho e facilitando a gestão de tarefas.

No desenvolvimento de *software*, o método Kanban proposto por Anderson ([ANDERSON, 2010](#)) é amplamente utilizado para melhorar a eficiência e o fluxo de trabalho, conhecida como “*Kanban para desenvolvimento de software*” ([ANDERSON, 2010](#)). Anderson aprimorou o método original, inicialmente utilizado pela Toyota na gestão de linhas de produção, para atender às demandas específicas do desenvolvimento ágil de *software*. Essa adaptação visa, sobretudo, melhorar a eficiência do fluxo de trabalho e a entrega contínua de valor, respeitando a capacidade da equipe e promovendo ajustes constantes nos processos.

Na sua forma mais básica, o Kanban organiza o trabalho em colunas que representam os diferentes estágios do fluxo de atividades. Essas colunas, dispostas em um quadro visual, facilitam a identificação do estado atual de cada tarefa, ajudando a evitar sobrecargas de trabalho e gargalos no processo. No contexto do **Reporta UFOP**, as principais colunas foram:

2.2.2.1 Coluna: A Fazer

Esta coluna agrupa as tarefas que foram identificadas e priorizadas, mas que ainda não foram iniciadas. O foco nesta fase é organizar e planejar as demandas de forma clara, evitando acúmulos desnecessários de atividades.

2.2.2.2 Coluna: Em Andamento

A coluna “Em Andamento” contém as tarefas que estão sendo executadas no momento. Uma das principais práticas do Kanban é limitar a quantidade de trabalho em progresso, ou seja, controlar o número de tarefas nessa coluna. Essa limitação evita a sobrecarga dos desenvolvedores e garante que o fluxo de trabalho seja constante e equilibrado.

2.2.2.3 Coluna: Concluído

A coluna “Concluído” reúne todas as tarefas que já foram finalizadas e aprovadas. Aqui, a equipe pode verificar o progresso do projeto de forma tangível e transparente. Esta visualização permite também um ciclo de *feedback* contínuo, essencial para o aprimoramento dos processos e entregas.

2.2.3 Código Limpo

Com a evolução dos *softwares*, a complexidade dos sistemas também aumentou significativamente. Os códigos tornaram-se mais extensos e os sistemas passaram a ser cada vez mais integrados, resultando em desafios maiores para a manutenção e evolução dos projetos. Diante dessa crescente complexidade, tornou-se indispensável a adoção de boas práticas de desenvolvimento de *software*, que não apenas facilitam a inserção de novos colaboradores, mas também garantem uma manutenção contínua e eficaz dos sistemas. Nesse contexto, surge o conceito de código limpo, que promove clareza, simplicidade e organização no código, tornando-o mais compreensível e sustentável a longo prazo.

Segundo Mikel ([ORS, 2022](#)), para que o código seja considerado limpo, ele deve possuir certas características essenciais que refletem a sua qualidade. Entre essas características, destacam-se:

- **Importância:** A qualidade do código é tão importante quanto o desempenho ou a funcionalidade, uma vez que um código bem escrito evita problemas como bugs e facilita futuras evoluções.
- **Legibilidade:** O código limpo deve ser fácil de ler para qualquer desenvolvedor. A clareza é fundamental para garantir que outros profissionais consigam entender o que foi implementado sem grandes dificuldades.
- **Modificabilidade:** Deve ser simples para qualquer desenvolvedor modificar o código. Isso inclui a facilidade de refatoração e ajustes sem que haja impacto negativo em outras partes do sistema.

- **Cuidado na escrita:** O código limpo reflete o cuidado e a atenção de quem o escreveu, demonstrando um compromisso com a qualidade e a manutenção do sistema.
- **Previsibilidade:** O código faz exatamente o que se espera dele, sem causar surpresas ou comportamentos inesperados. A previsibilidade é crucial para evitar erros e garantir a confiança dos desenvolvedores no sistema.

Essas características destacam a importância de um código bem estruturado, que não só garante a funcionalidade do sistema, mas também facilita a colaboração e a escalabilidade, sendo um dos pilares fundamentais de um desenvolvimento de *software* de qualidade.

2.2.4 Desenvolvimento multiplataforma

Tendo em vista o surgimento de diversos sistemas operacionais, como Android e iOS, foram desenvolvidas ferramentas que facilitam a criação de aplicações capazes de utilizar um mesmo código em múltiplas plataformas. Assim, surgiu o conceito de desenvolvimento multiplataforma, que, segundo [El-Kassas et al. \(2017\)](#), consiste em desenvolver um aplicativo uma única vez e executá-lo em diferentes sistemas operacionais.

Uma das principais ferramentas utilizadas atualmente é o **Flutter**, um kit de ferramentas de interface do usuário (UI) criado pelo Google ([GOOGLE, 2024](#)). De acordo com a documentação oficial, o Flutter permite o desenvolvimento de aplicativos nativamente compilados para diversas plataformas, como dispositivos móveis, web e desktop, a partir de uma única base de código. Além disso, o Flutter oferece uma experiência de desenvolvimento ágil e produtiva, permitindo que as alterações no código sejam visualizadas quase instantaneamente.

2.2.5 Software Aberto

O desenvolvimento de *software* aberto tem ganhado crescente popularidade na comunidade de desenvolvedores, pois essa prática facilita a colaboração, manutenção e melhoria contínua de programas já estabelecidos. Um exemplo notável é o sistema operacional Linux, um dos casos mais emblemáticos de *software* de código aberto, amplamente utilizado por usuários em todo o mundo.

O conceito de *software* aberto baseia-se na disponibilidade do código-fonte para qualquer pessoa interessada em acessá-lo, modificá-lo e distribuí-lo. Segundo a [Open Source Initiative \(OSI\) \(2001\)](#), o *software* aberto segue princípios que garantem a liberdade de uso e distribuição, sem restrições quanto à venda ou cobrança de taxas. Além disso, as licenças de *software* aberto devem assegurar que o código-fonte seja acessível, permitindo sua

modificação e redistribuição. Alterações ao código original devem ser claramente indicadas, preservando a integridade do autor original.

Outro princípio fundamental é a não discriminação, ou seja, as licenças de *software* aberto não podem restringir o uso por grupos específicos de pessoas nem limitar o tipo de uso que será dado ao *software*. As licenças também se aplicam a todas as redistribuições, independentemente do pacote no qual o *software* esteja inserido, sem interferir em outros programas que possam ser distribuídos conjuntamente.

2.2.6 Sistema Android

Segundo o *Android Developers Developers* (2024), o sistema Android é um sistema operacional baseado em Linux, com suporte a múltiplos usuários. Desde o seu lançamento, o Android tornou-se um dos sistemas operacionais mais utilizados em todo o mundo. Em 2016, o Android ultrapassou a marca de 2 bilhões de dispositivos ativos globalmente (ANDROID, 2024).

A linguagem oficial de desenvolvimento para Android é o Kotlin, sendo o Java também amplamente utilizado. No entanto, existem diversos *frameworks* que permitem a conversão de códigos escritos em outras linguagens e sua compilação para Android nativo, como o Flutter e o React Native.

Outro fator positivo do ecossistema Android é a ampla variedade de dispositivos que utilizam o sistema operacional. Atualmente, existem mais de 24.000 dispositivos Android diferentes, fabricados por cerca de 1.300 marcas ao redor do mundo (ANDROID, 2024). Essa diversidade facilita o desenvolvimento de aplicações que atingem um público maior, possibilitando a adaptação de soluções para uma vasta gama de usuários e dispositivos.

2.2.7 Arquitetura Limpa

Atividades acadêmicas e empresariais voltadas à tecnologia de programação geralmente seguem especificações arquiteturais para garantir que o *software* seja escalável e de fácil manutenção. Segundo Ivanics (2016), o desenvolvimento de *software* é, na maioria dos casos, um trabalho colaborativo em equipe. Nesse contexto, torna-se imprescindível adotar padrões de arquitetura, pois eles facilitam a compreensão das funcionalidades do sistema ao organizá-las em camadas bem definidas.

Uma das arquiteturas mais amplamente difundidas nesse cenário é a Arquitetura Limpa, proposta por Martin (2012), que visa separar responsabilidades e promover a independência entre as camadas, tornando o sistema mais modular, compreensível e sustentável a longo prazo.

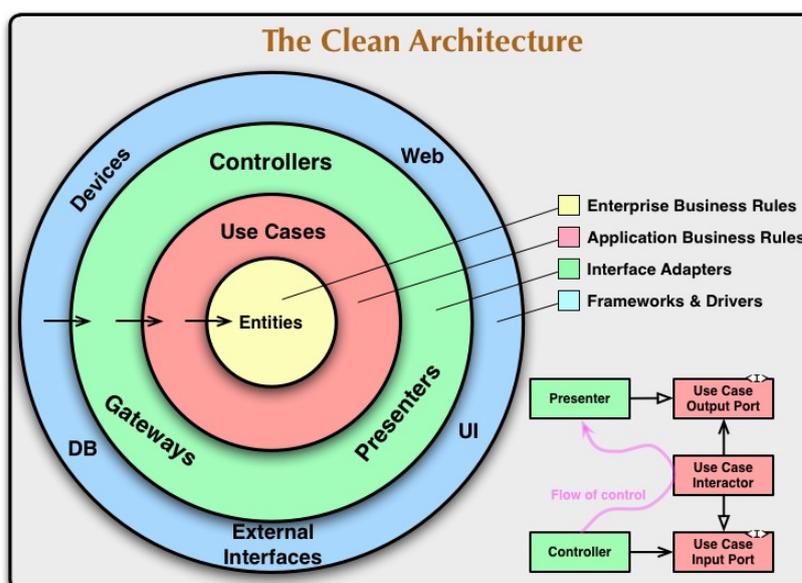


Figura 1 – Representação da Arquitetura Limpa. Fonte: Blog Robert C. Martin

Conforme ilustrado na Figura 1, a Arquitetura Limpa organiza o sistema em camadas concêntricas, onde cada camada tem suas responsabilidades bem definidas e depende apenas de camadas internas. As camadas mais centrais lidam com as regras de negócio, enquanto as externas cuidam de detalhes de implementação, como interfaces de usuário, bancos de dados e *frameworks*. Isso garante que as mudanças nas camadas externas (como a troca de *frameworks* ou tecnologias de banco de dados) não afetem as camadas internas mais críticas.

Componentes da Arquitetura Limpa:

Entidades: Localizadas no centro do modelo, as entidades representam as regras de negócio mais fundamentais e independentes da aplicação. Elas não conhecem detalhes da implementação e podem ser usadas em diferentes contextos.

Casos de uso (*Use Cases*): Responsáveis por coordenar as interações entre entidades e outras partes do sistema. São as regras de negócio específicas da aplicação e garantem que as funcionalidades atendam aos requisitos do negócio.

Adaptadores de Interface: Adaptam as entradas e saídas da aplicação, convertendo dados entre os casos de uso e as interfaces externas, como a interface gráfica do usuário (UI), APIs ou sistemas externos.

***Frameworks* e *Drivers*:** Camada mais externa, que interage com ferramentas, bibliotecas e *frameworks*. Aqui estão os detalhes técnicos que podem ser trocados ou modificados sem afetar as camadas internas.

2.2.8 Controle de Versão

Segundo a documentação da Microsoft ([MICROSOFT, 2024a](#)), um sistema de controle de versão é uma ferramenta que auxilia no controle das mudanças realizadas em códigos de *software* ao longo do tempo. Quando um programador altera um determinado código, ele pode realizar um *commit*, que funciona como uma marcação no histórico do projeto, capturando o estado do código naquele momento. Esses *commits* geram um histórico detalhado das modificações realizadas, facilitando a identificação de mudanças, correção de bugs e o compartilhamento de projetos entre equipes.

O controle de versão é uma prática essencial no desenvolvimento de *software*, pois torna o processo de iteração mais organizado e eficiente. Sem essa ferramenta, os programadores tendem a salvar versões de seus códigos em vários arquivos separados, o que dificulta a organização e o acompanhamento das etapas de desenvolvimento ao longo do tempo. Com o uso de um sistema de controle de versão, é possível gerenciar diferentes versões de um projeto de forma clara, garantindo que todas as modificações sejam rastreáveis e reversíveis, além de facilitar o trabalho colaborativo.

3 Desenvolvimento

Este capítulo descreve as etapas e escolhas das técnicas realizadas durante o desenvolvimento do projeto, abordando desde a definição dos requisitos até a aplicação das ferramentas e metodologias utilizadas para a implementação da solução.

3.1 Casos de Uso

Após a idealização do projeto, a próxima etapa foi a definição dos casos de uso da aplicação, levando em consideração as necessidades dos usuários e as funcionalidades propostas. Dessa forma, foram estabelecidos os seguintes casos de uso:

Tabela 1 – Caso de uso: Criação de Conta e *Login* (Estudantes e Professores)

Caso de uso	Criação de Conta e <i>Login</i> (Estudantes e Professores)
Descrição	O estudante deve poder criar uma conta ou fazer <i>login</i> no sistema.
Atores	Estudantes e Professores
Objetivo	Garantir que os usuários possam acessar o sistema para reportar problemas e acompanhar seus reportes.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O usuários insere os dados pessoais e cria uma conta. 2. O usuários faz <i>login</i> com e-mail e senha. 3. O sistema autentica o usuário e permite o acesso à plataforma.

Tabela 2 – Caso de uso: Submissão de Reportes (Estudantes e Professores)

Caso de uso	Submissão de Reportes (Estudantes e Professores)
Descrição	Os usuários podem reportar problemas que identificam no campus.
Atores	Estudantes e Professores
Objetivo	Permitir que os estudantes registrem problemas, incluindo descrição, imagem e localização.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O usuários acessa a opção de adicionar reporte. 2. O sistema navega para a tela de captura de imagens. 3. O usuários captura uma imagem do problema. 4. O sistema navega para a tela de detalhes do reporte. 5. O usuários adiciona uma descrição detalhada do problema. 6. A localização geográfica é registrada automaticamente pelo sistema. 7. O usuários envia o reporte. 8. O reporte é salvo no Firebase Storage.

Tabela 3 – Caso de uso: Acompanhamento de Reportes (Estudantes e Professores)

Caso de uso	Acompanhamento de Reportes (Estudantes e Professores)
Descrição	Os usuários podem acompanhar o status de seus reportes.
Atores	Estudantes e Professores
Objetivo	Manter o usuários informado sobre o progresso do reporte enviado.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O usuários clica em Ver Meus Reportes. 2. O sistema navega para a tela de Meus Reportes. 3. O usuários verifica o status do reporte (Resolvido, Em Andamento ou Pendente). 4. O sistema apresenta o status atualizado e outras informações relevantes.

Tabela 4 – Caso de uso: *logout* (Estudantes e Professores)

Caso de uso	<i>logout</i> (Estudantes e Professores)
Descrição	O estudante pode encerrar sua sessão no aplicativo, retornando para a tela de <i>login</i> .
Atores	Estudantes e Professores
Objetivo	Permitir que o estudante encerre sua sessão de forma segura.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O usuário clica no botão “<i>logout</i>” na tela principal do aplicativo. 2. O sistema exibe uma mensagem de confirmação para encerrar a sessão. 3. O usuário confirma a ação. 4. O sistema encerra a sessão do estudante e o redireciona para a tela de <i>login</i>.

Tabela 5 – Caso de uso: Gestão de Acessos (Administradores)

Caso de uso	Gestão de Acessos (Administradores)
Descrição	Os administradores podem criar contas e fazer <i>login</i> para acessar a plataforma.
Atores	Administrador
Objetivo	Garantir que apenas usuários administradores possam gerenciar os reportes.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O administrador faz <i>login</i> na plataforma web. 2. O sistema autentica o administrador e concede acesso ao painel de controle.

Tabela 6 – Caso de uso: Visualização e Filtragem de Reportes (Administradores)

Caso de uso	Visualização e Filtragem de Reportes (Administradores)
Descrição	Administradores podem visualizar e filtrar todos os reportes submetidos.
Atores	Administrador
Objetivo	Permitir que os administradores organizem e priorizem os problemas para ação.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O administrador acessa a lista de reportes. 2. O administrador filtra os reportes por status (Resolvido, Em Andamento ou Pendente). 3. O administrador visualiza os detalhes do reporte, como descrição, imagem e localização do problema.

Tabela 7 – Caso de uso: Edição e Resolução de Reportes (Administradores)

Caso de uso	Edição e Resolução de Reportes (Administradores)
Descrição	Administradores podem alterar o status de um reporte e marcar problemas como resolvidos.
Atores	Administrador
Objetivo	Facilitar a resolução dos problemas reportados pelos estudantes.
Fluxo de eventos	<ol style="list-style-type: none"> 1. O administrador visualiza um reporte submetido. 2. O administrador atualiza o status para Resolvido, Em Andamento ou Pendente. 3. O sistema atualiza o status do reporte e notifica o estudante.

Tabela 8 – Caso de uso: *Logout* (Administrador)

Caso de uso	<i>Logout</i> (Administrador)
Descrição	O administrador pode encerrar sua sessão no sistema, retornando para a tela de <i>login</i> .
Atores	Administrador
Objetivo	Permitir que o administrador encerre sua sessão de forma segura.
Fluxo de eventos	<ol style="list-style-type: none">1. O administrador clica no botão “<i>logout</i>” na tela principal do sistema.2. O sistema exibe uma mensagem de confirmação para encerrar a sessão.3. O administrador confirma a ação.4. O sistema encerra a sessão do administrador e o redireciona para a tela de <i>login</i>.

3.2 Ferramentas Utilizadas

Com os casos de uso definidos, avançamos para a etapa de escolha das ferramentas necessárias para a construção do aplicativo. As ferramentas selecionadas foram:

3.2.1 Dart

Dart é uma linguagem de programação multiparadigma e multiplataforma criada pelo Google em 2011, com o lançamento da sua versão estável em 2013. A linguagem ganhou popularidade com o surgimento do *framework* Flutter, sendo amplamente utilizada no desenvolvimento de aplicações modernas.

3.2.2 Flutter

Flutter é um *framework* para o desenvolvimento de aplicativos multiplataforma, conforme discutido no capítulo 2.2.4. A escolha do Flutter foi motivada pelo fato de que o *framework* compila diretamente para código nativo, proporcionando um desempenho superior em comparação aos seus concorrentes. Além disso, o Flutter vem ganhando grande popularidade nos últimos anos, ultrapassando seu principal concorrente, o React Native. Esse crescimento pode ser observado nas métricas de popularidade do Google Trends, conforme ilustrado na Figura 2 (GOOGLE TRENDS, 2024).

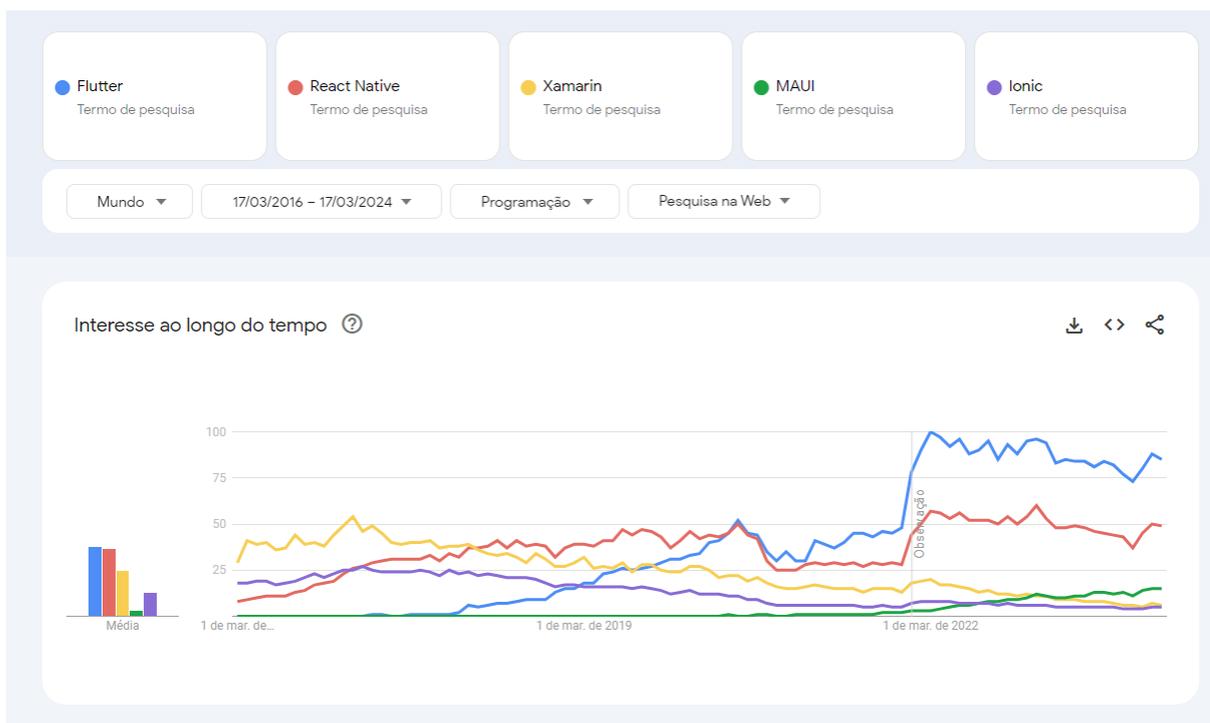


Figura 2 – Comparação de Popularidade entre Flutter, React Native, Xamarin, MAUI e Ionic. Fonte: Google Trends ([GOOGLE TRENDS, 2024](#)).

3.2.3 Visual Studio Code

A IDE (*Integrated Development Environment*) escolhida para o desenvolvimento deste projeto foi o Visual Studio Code. De acordo com a própria documentação ([MICROSOFT, 2024b](#)), o Visual Studio Code é um editor de código-fonte leve, mas extremamente poderoso. Ele está disponível para os sistemas operacionais Windows, macOS e Linux.

Uma das principais vantagens de utilizar o Visual Studio Code é a vasta gama de extensões que ele oferece, as quais ajudam a aumentar significativamente a produtividade no desenvolvimento. Além disso, o editor possui suporte nativo para o Flutter, o que facilita o desenvolvimento de aplicativos multiplataforma.

3.2.4 Firebase

O Firebase é uma plataforma de *Backend-as-a-Service* (BaaS), fornecendo uma estrutura completa de back-end que pode ser integrada a projetos de desenvolvimento. Essa plataforma foi essencial para o desenvolvimento da aplicação *Reporta UFOP*, uma vez que diversos serviços do Firebase foram utilizados. Entre eles, destacam-se:

- **Authentication:** Este serviço oferece um sistema robusto de cadastro e autenticação de usuários, facilitando o controle de acesso à aplicação.

- **Cloud Firestore:** Este serviço funciona como um banco de dados NoSQL, onde foram armazenadas as seguintes informações relacionadas aos reportes:
 - *imageUrl*: URL da imagem associada ao reporte.
 - *latitude*: Latitude do local onde o problema foi identificado.
 - *longitude*: Longitude do local onde o problema foi identificado.
 - *name*: Nome do usuário que criou o reporte.
 - *observation*: Observações e detalhes adicionais sobre o problema reportado.
 - *status*: Status atual do reporte (ex.: em andamento, resolvido).
 - *timestamp*: Data e hora em que o reporte foi criado.
- **Storage:** Este serviço oferece uma solução eficiente para o armazenamento de imagens, possibilitando o upload e a recuperação de fotos associadas aos reportes.

3.2.5 Git

O Git é um sistema de controle de versão distribuído, conforme explicado na Seção 2.2.8 (*Controle de Versão*). No desenvolvimento do projeto *Reporta UFOP*, o Git foi fundamental para gerenciar o código-fonte de maneira eficiente. Ele permitiu o controle rigoroso das alterações no código ao longo do tempo, possibilitando a criação de *branches* para diferentes funcionalidades e correções, bem como a integração de alterações de forma colaborativa e segura. O uso do Git também facilitou o rastreamento de modificações, a resolução de conflitos de código e o histórico detalhado de cada iteração do projeto, garantindo um fluxo de desenvolvimento mais organizado e confiável.

3.2.6 Trello

Para organizar o desenvolvimento do projeto com base na metodologia Kanban, utilizou-se o Trello, uma ferramenta de gestão visual de trabalho (TRELLO, 2024). No Trello, foram criadas três colunas principais: *A Fazer*, *Em Andamento* e *Concluído*. As tarefas foram distribuídas e movidas entre as colunas conforme seu progresso, permitindo uma visão clara e rápida do andamento do projeto, desde o planejamento até a execução e conclusão.

A Figura 3 ilustra o quadro de tarefas no Trello, onde as atividades relacionadas ao desenvolvimento do sistema *Reporta UFOP* foram organizadas.

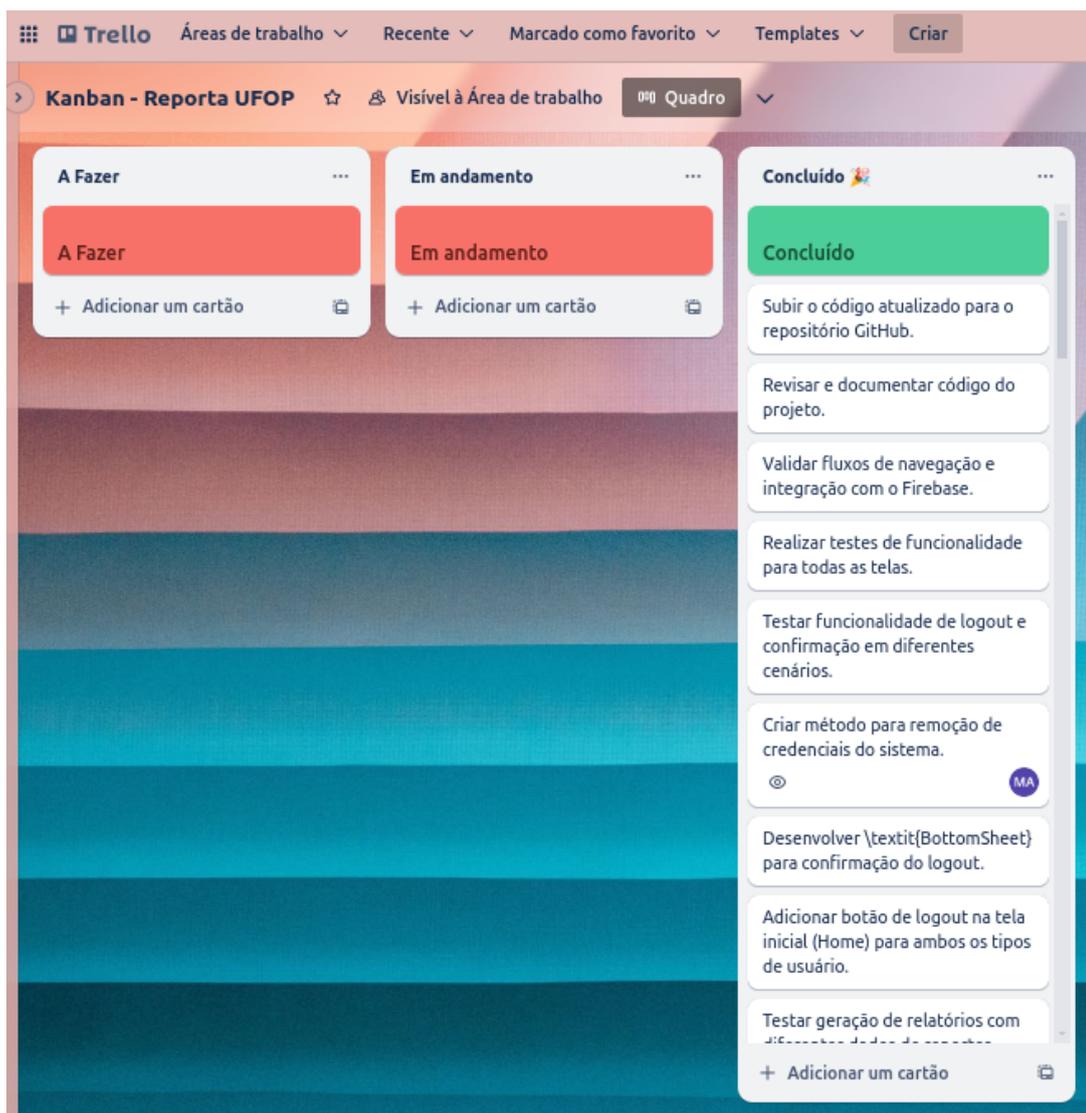


Figura 3 – Quadro Kanban no Trello com as tarefas do projeto *Reporta UFOP*.

3.2.7 Bibliotecas

Durante o levantamento dos requisitos, foi identificado que seriam necessárias bibliotecas específicas para lidar com funcionalidades como gerenciamento de estado, manipulação de imagens, integração com o Firebase, entre outras. A seguir, detalha-se o uso de cada uma das bibliotecas utilizadas no desenvolvimento do projeto.

3.2.8 Bibliotecas Utilizadas

Como ilustrado na Tabela 9, as bibliotecas listadas foram essenciais para a implementação das funcionalidades do projeto, proporcionando suporte à autenticação, gerenciamento de estado, manipulação de imagens, acesso a recursos do dispositivo e armazenamento de dados.

Biblioteca	Descrição
<code>firebase_core</code> e <code>firebase_auth</code>	Responsáveis pela integração com o Firebase, proporcionando autenticação e acesso aos serviços centrais da plataforma.
<code>get</code>	Utilizada para gerenciamento de estado e navegação entre rotas de forma eficiente.
<code>flutter_svg</code>	Empregada para a manipulação e exibição de imagens no formato SVG.
<code>camera</code>	Fornecer acesso à câmera do dispositivo, permitindo a captura de fotos e vídeos.
<code>mobx</code>	Utilizada para o gerenciamento reativo de estado, facilitando a reatividade no aplicativo.
<code>image_picker</code>	Biblioteca responsável pela seleção de imagens da galeria ou captura diretamente pela câmera.
<code>flutter_bloc</code>	Arquitetura de controle de estado baseada no padrão Bloc, promovendo organização e manutenibilidade.
<code>firebase_storage</code>	Utilizada para o armazenamento de arquivos na nuvem, integrado ao Firebase.
<code>permission_handler</code>	Gerencia as permissões necessárias no sistema, como acesso à câmera e localização.
<code>path_provider</code>	Acessa o sistema de arquivos para armazenar e recuperar dados localmente.
<code>geolocator</code>	Usada para obter a localização geográfica do dispositivo, crucial para algumas funcionalidades do projeto.
<code>firebase_app_check</code>	Implementa verificações de integridade para garantir a autenticidade do aplicativo.
<code>url_launcher</code>	Permite abrir URLs externas diretamente a partir do aplicativo, como links de navegação ou sites.
<code>cloud_firestore</code>	Banco de dados em tempo real da plataforma Firebase, utilizado para armazenamento e sincronização de dados.

Tabela 9 – Bibliotecas Utilizadas no Desenvolvimento do Projeto

Essas bibliotecas foram essenciais para a implementação das funcionalidades e para garantir a escalabilidade e a manutenção do projeto ao longo do tempo.

Além dos aspectos técnicos, o desenvolvimento levou em consideração os requisitos funcionais e não funcionais, visando garantir que o aplicativo atenda às necessidades dos usuários de maneira eficiente e escalável.

3.3 Prototipagem

A prototipagem é uma etapa essencial no desenvolvimento de produtos e sistemas, permitindo a simulação inicial do conceito e a validação de funcionalidades antes da implementação final. Conforme destaca [Palhais \(2015\)](#), “a prototipagem auxilia no processo

de desenvolvimento de produto, uma vez que possibilita a simulação do produto em escala real.” Essa abordagem é fundamental para identificar falhas precocemente, ajustar funcionalidades e alinhar o projeto às expectativas dos usuários.

No contexto do Reporta UFOP, a ferramenta [Figma \(2025\)](#) foi utilizada para realizar a prototipagem do *layout* da aplicação. Essa plataforma permite criar protótipos de alta fidelidade, com simulação de interações e fluxos de navegação, proporcionando uma visão clara e realista da interface proposta.

Com base nos casos de uso apresentados, os protótipos foram desenvolvidos para atender às especificações técnicas e funcionais da aplicação. Os protótipos foram organizados em duas categorias principais: sistema dos estudantes e sistema dos administradores.

3.3.0.1 Funcionalidades Destinadas a Estudantes

O sistema dos estudantes inclui as telas de login, cadastro, home, detalhes do reporte e os reportes feitos pelo usuário. As figuras a seguir ilustram o *design* dessas telas.

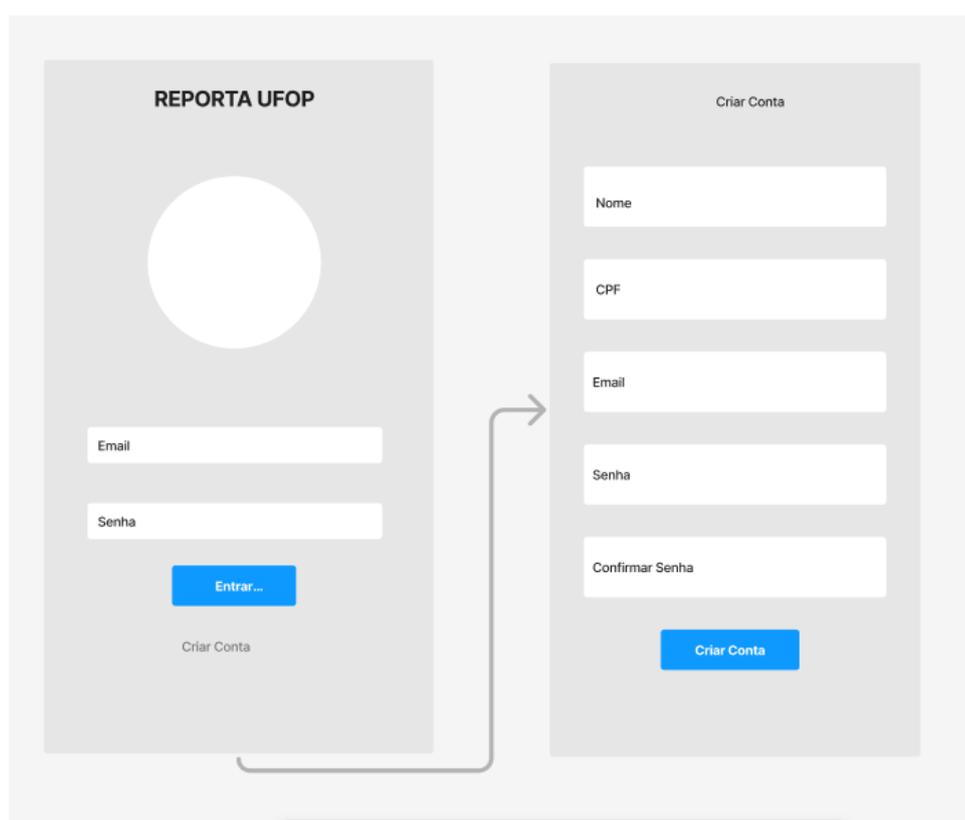


Figura 4 – Protótipo do sistema dos estudantes: Tela de Login e Cadastro.

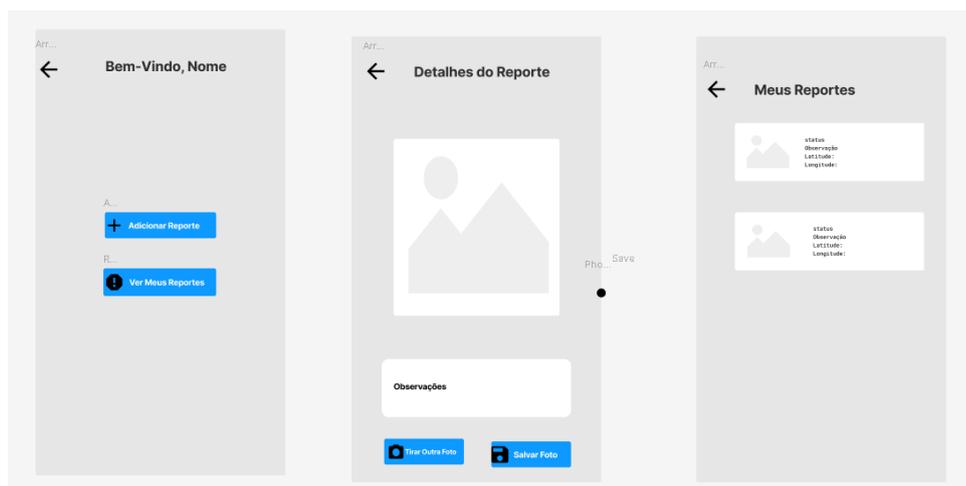


Figura 5 – Protótipo do sistema dos estudantes: Tela Home, Detalhes do Reporte e Reportes Feitos pelo Usuário.

3.3.0.2 Funcionalidades destinadas a Administradores

O sistema dos administradores inclui as telas de login, home, menu de gerenciamento de reportes e relatórios. A seguir, estão os protótipos dessas telas.

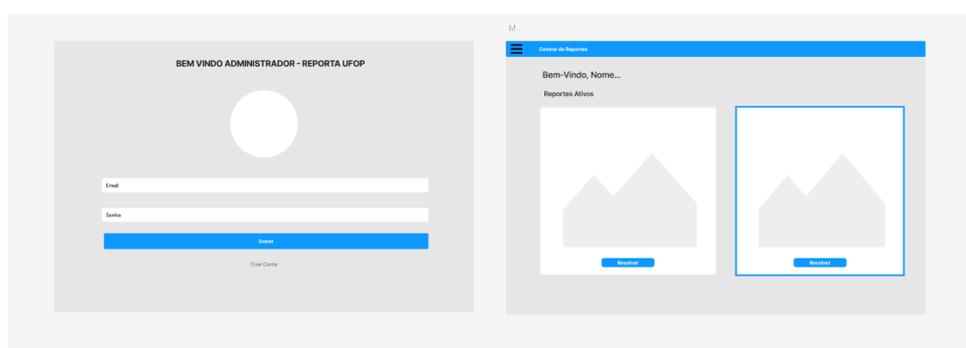


Figura 6 – Protótipo do sistema dos administradores: Tela de Login e Home.

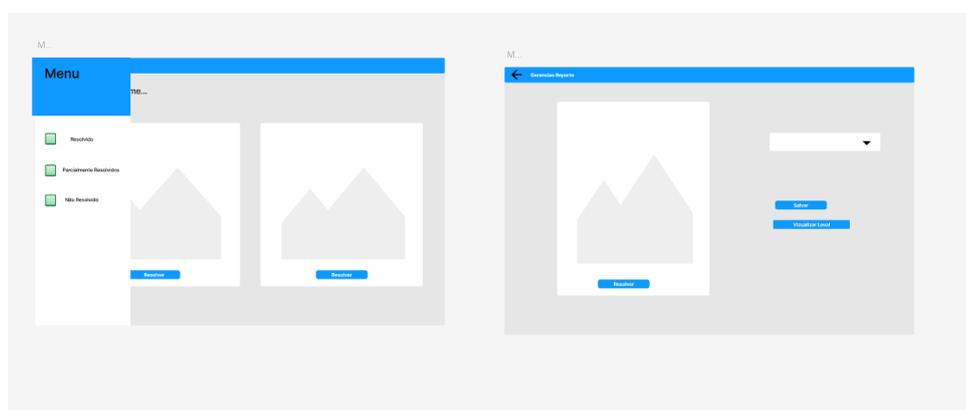


Figura 7 – Protótipo do sistema dos administradores: Menu e Gerenciamento de Reportes.

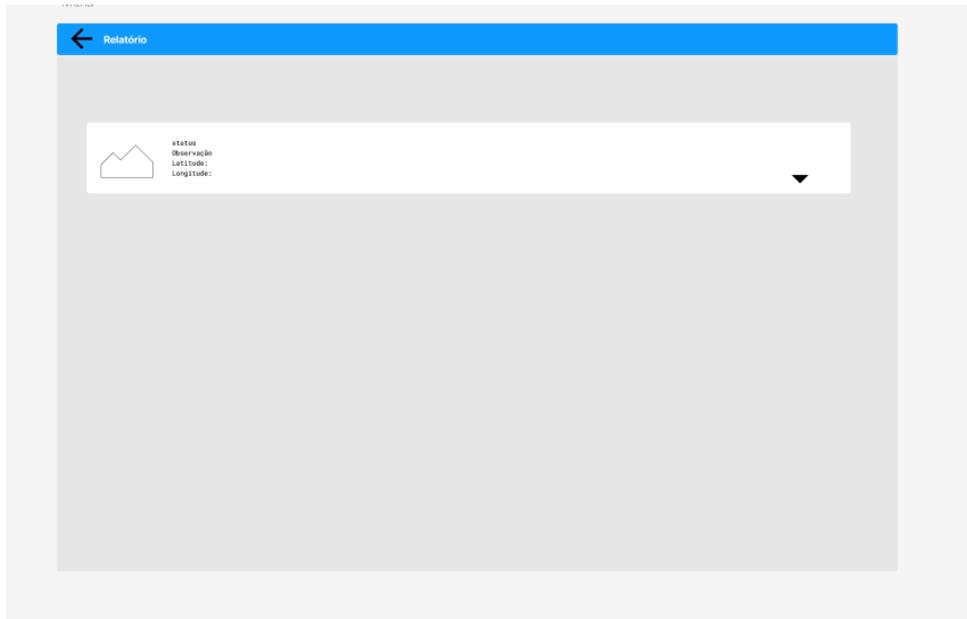


Figura 8 – Protótipo do sistema dos administradores: Relatório.

3.3.1 Ciclo de Desenvolvimento do Projeto

Para organizar o ciclo de desenvolvimento do projeto *Reporta UFOP*, foi adotado o método Kanban, caracterizado pela flexibilidade e pelo uso de um quadro visual que permite gerenciar o fluxo contínuo das tarefas. Essa abordagem foi escolhida devido à sua capacidade de adaptação às mudanças ao longo do desenvolvimento, permitindo a reorganização dinâmica das prioridades, otimizando o gerenciamento do projeto e facilitando a visualização das etapas concluídas e pendentes.

3.3.1.1 Etapa 1: Configuração Inicial

A primeira etapa do desenvolvimento consistiu na criação do projeto e na implementação da estrutura de pastas, seguindo o padrão de arquitetura limpa [Martin \(2012\)](#), adaptado ao contexto específico do projeto. Além disso, foi configurado o repositório no Git, possibilitando o controle de versões, e adicionadas as dependências necessárias para atender às funcionalidades definidas nos casos de uso. A [Figura 9](#) ilustra a estrutura inicial do projeto e a configuração no Git.

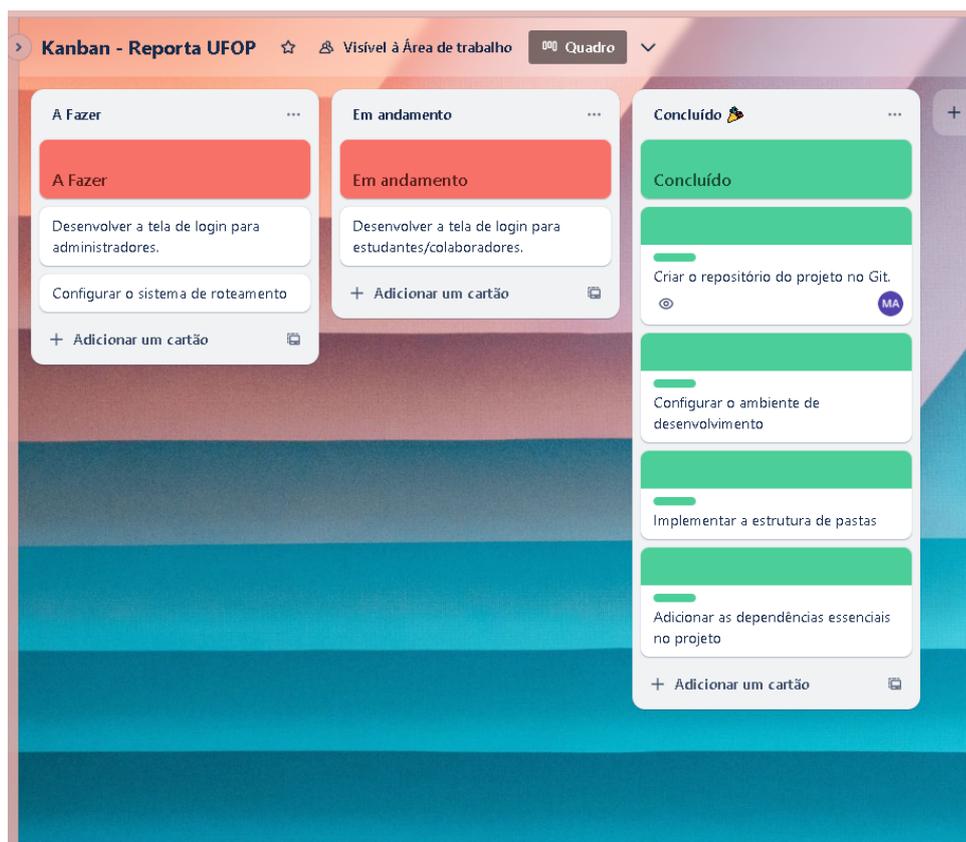


Figura 9 – Estrutura inicial do projeto e configuração no Git.

3.3.1.2 Etapa 2: Desenvolvimento das Telas de Login e Roteamento

Na segunda etapa, foram desenvolvidas as telas de login para os estudantes e administradores. Durante essa fase, foi configurado o sistema de roteamento das telas, diferenciando as funcionalidades com base no contexto de uso. Caso a aplicação seja executada via APK, as rotas são configuradas para as funcionalidades dos estudantes; se acessada via navegador, direcionam para as funcionalidades dos administradores. A Figura 10 mostra os protótipos das telas de login e a configuração do roteamento.



Figura 10 – Protótipos das telas de login e configuração de roteamento.

3.3.1.3 Etapa 3: Telas de Cadastro e Configurações no Firebase

Na terceira etapa, foram desenvolvidas as telas de cadastro para ambos os perfis, além de criar o projeto no Firebase e realizar as configurações necessárias. No serviço *Authentication*, foram implementadas as funcionalidades de criação de conta e login, com validação por e-mail e senha. A Figura 11 apresenta as configurações no Firebase e o desenvolvimento das telas de cadastro.

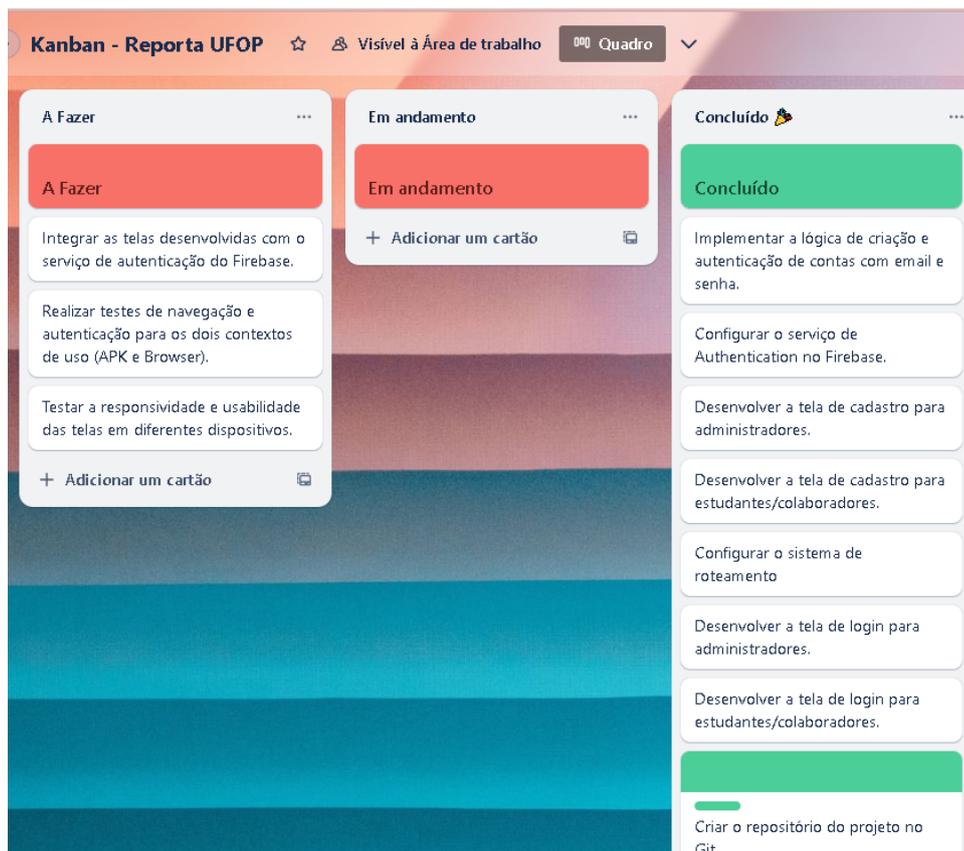


Figura 11 – Configurações no Firebase e desenvolvimento das telas de cadastro.

3.3.1.4 Etapa 4: Desenvolvimento da Tela Inicial e Funcionalidades para Estudantes

Na quarta etapa, foi criada a **Tela Inicial** destinada aos estudantes e colaboradores. Esta tela possui dois botões principais: um para acessar a câmera do dispositivo e outro para visualizar os reportes enviados pelo usuário.

Além disso, foi desenvolvida a tela da câmera, o que exigiu ajustes no arquivo `AndroidManifest.xml` para solicitar permissões específicas para uso da câmera em dispositivos Android. Ferramentas para ativar e capturar a geolocalização do usuário também foram configuradas, permitindo registrar as coordenadas de latitude e longitude no momento da submissão dos reportes.

3.3.1.5 Etapa 5: Configuração do Serviço de Armazenamento (Firebase Storage)

Na quinta etapa, foi configurado o **Firebase Storage**, que funciona como um banco de dados para o armazenamento de arquivos. Neste projeto, o serviço foi configurado para receber e armazenar as fotos capturadas pelos usuários, além dos respectivos metadados, como coordenadas de latitude e longitude. Essa integração garantiu o armazenamento seguro e estruturado dos dados, acessíveis tanto para os administradores quanto para os usuários. A Figura 12 exemplifica a Tela Inicial e a configuração do Firebase Storage.

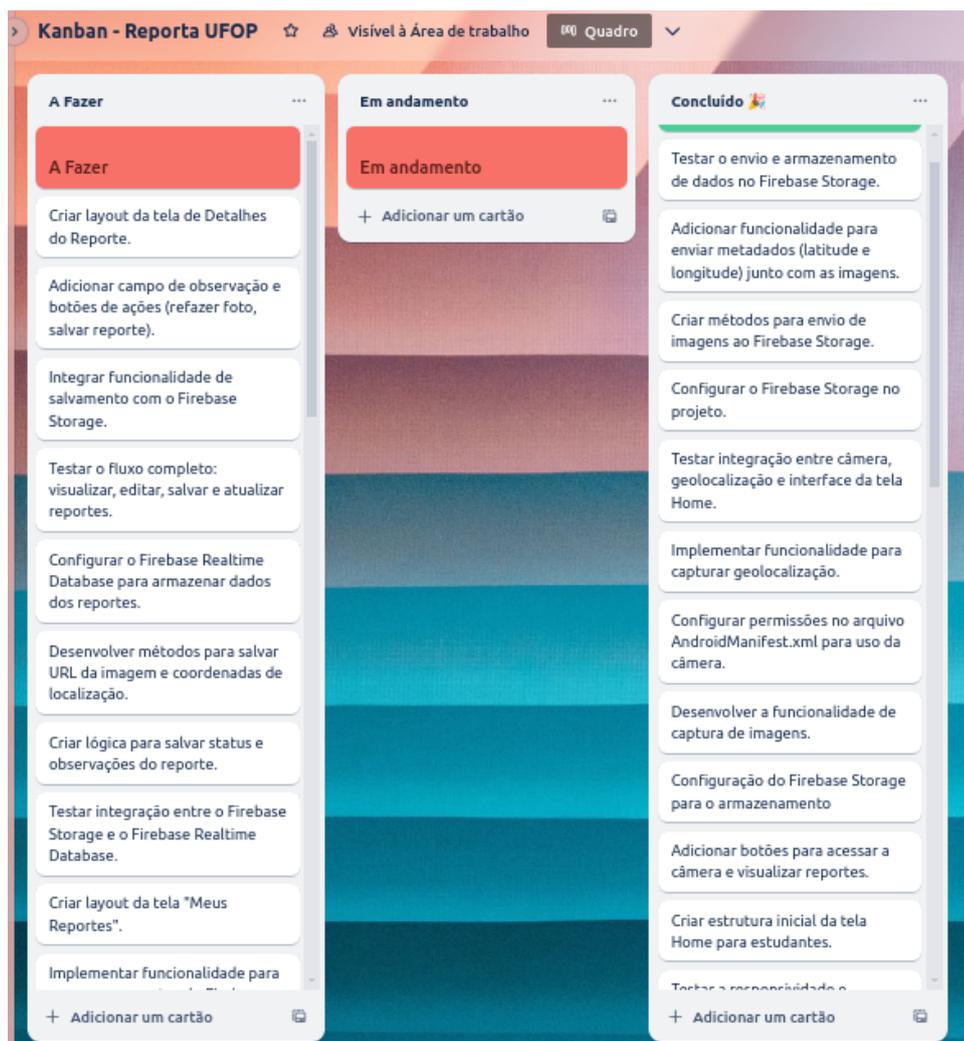


Figura 12 – Tela Inicial e Configuração do Firebase Storage

3.3.1.6 Etapa 6: Tela de Detalhes do Reporte e Integração com Firebase Storage

Na sexta etapa, foi criada a **Tela de Detalhes do Reporte**, permitindo ao usuário:

- Visualizar um **preview** da foto registrada;
- Inserir uma observação descritiva sobre o problema reportado;
- Escolher entre refazer a foto ou salvar o reporte.

A funcionalidade de salvamento foi integrada com os métodos do Firebase Storage, garantindo o armazenamento da imagem e de seus metadados. Essa etapa também envolveu o desenvolvimento do repositório e do *Presenter* responsável pelo envio dos dados ao Firebase.

3.3.1.7 Etapa 7: Configuração do Banco de Dados Firebase

Na sétima etapa, foi configurado o **Firestore Database**, utilizado para armazenar os dados relacionados aos reportes, como:

- URL da imagem capturada;
- Coordenadas de latitude e longitude;
- Observações adicionadas pelo usuário;
- Status do reporte (Pendente, Em Andamento ou Resolvido);
- Data e hora do registro do reporte.

A Figura 13 ilustra a Tela de Detalhes do Reporte e a configuração do Banco de Dados.

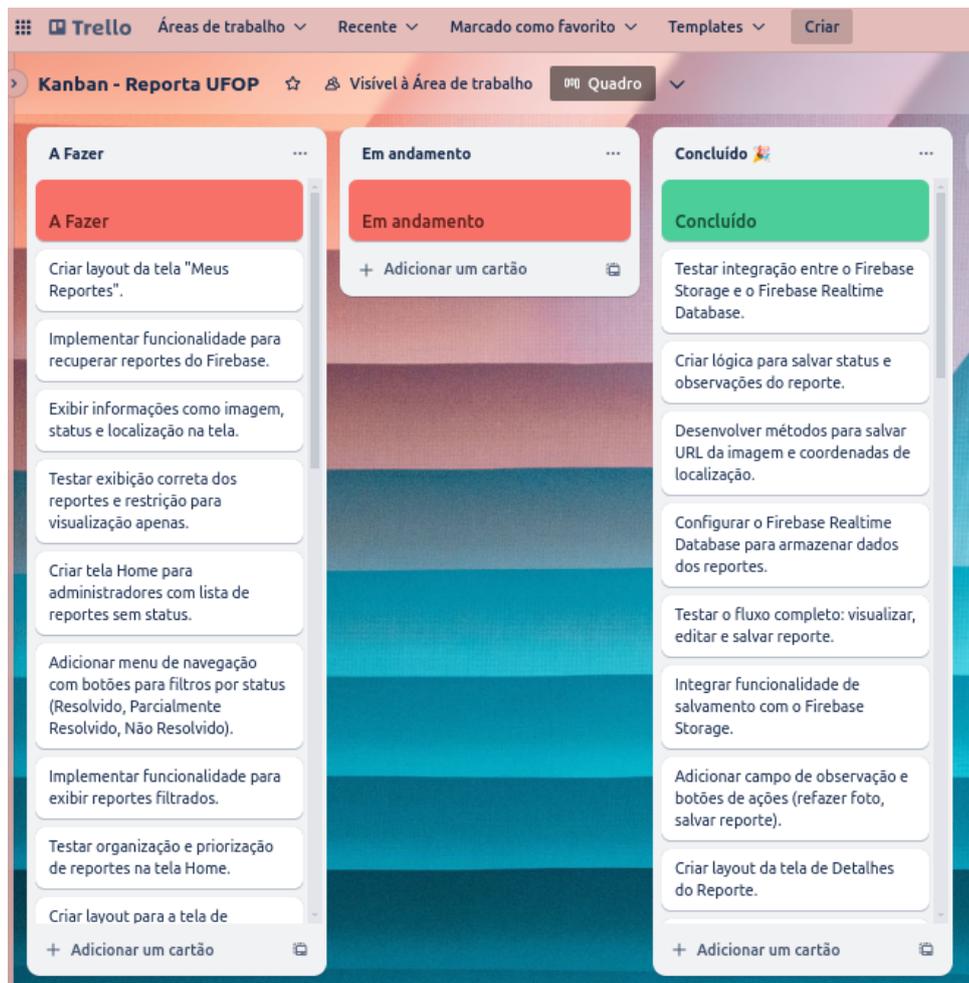


Figura 13 – Detalhes do Reporte e Banco de Dados

3.3.1.8 Etapa 8: Tela “Meus Reportes”

Na oitava etapa, foi desenvolvida a tela **Meus Reportes**, uma funcionalidade destinada exclusivamente aos estudantes e colaboradores. Esta tela foi projetada para permitir que os usuários visualizem os reportes que eles mesmos submeteram. Para isso, foram integrados serviços do Firebase e criados métodos específicos para recuperar e exibir os dados armazenados.

É importante ressaltar que, no caso dos estudantes e colaboradores, o acesso à tela se limita apenas à **visualização** dos reportes, sem permissão para realizar edições ou alterações nos dados registrados. A Figura 14 mostra a tela “Meus Reportes” em funcionamento.



Figura 14 – Desenvolvimento da Tela “Meus Reportes”

3.3.1.9 Etapa 9: Desenvolvimento da Tela Home e Funcionalidades dos Administradores

A nona etapa focou no desenvolvimento da **Tela Home** destinada aos administradores. Essa tela foi projetada para exibir imagens dos reportes que ainda não possuem status definido, permitindo que os administradores tenham uma visão geral das pendências. Para isso, foram implementados métodos de integração com o Firebase, responsáveis por recuperar e exibir os dados relevantes.

Além disso, foi construído um **menu de navegação**, contendo três componentes clicáveis, que representam os diferentes status de resolução dos reportes:

- **Resolvidos:** Reportes cuja solução foi completamente implementada.
- **Parcialmente Resolvidos:** Reportes em que uma solução parcial foi aplicada.

- **Não Resolvidos:** Reportes que ainda aguardam ação administrativa.

Essas funcionalidades foram desenvolvidas para organizar e priorizar a resolução dos problemas reportados, proporcionando maior eficiência na gestão por parte dos administradores. A Figura 15 apresenta o desenvolvimento da Tela Home para os administradores.

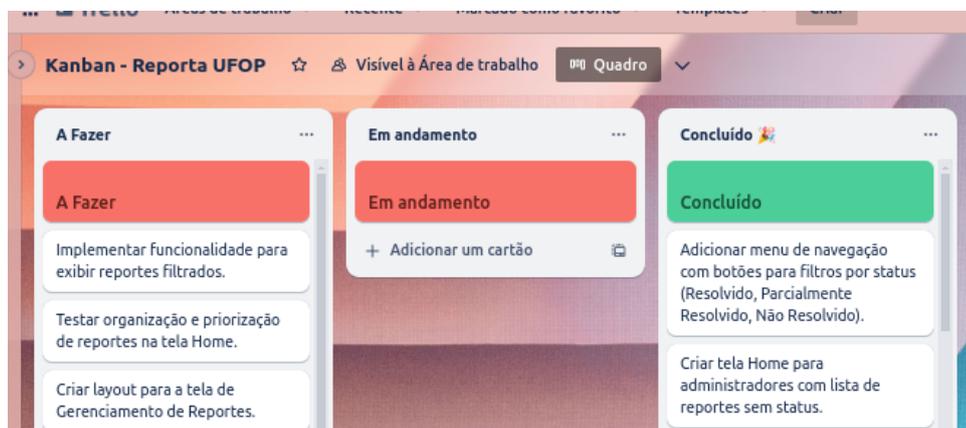


Figura 15 – Desenvolvimento da Tela Home

3.3.1.10 Etapa 10: Implementação da Tela de Gerenciamento de Reportes

Nesta etapa, foi desenvolvida a tela de **Gerenciamento de Reportes**, destinada aos administradores para auxiliar na resolução e acompanhamento de reportes submetidos. Essa tela apresenta as informações do reporte selecionado e inclui funcionalidades específicas para sua gestão.

3.3.1.10.1 Funcionalidades Implementadas:

- **Dropdown Button:** Permite ao administrador selecionar o status de resolução do reporte (*Resolvido*, *Em Andamento*, *Não Resolvido*).
- **Botão Salvar:** Salva as alterações feitas no status do reporte, atualizando os dados no banco de dados *Firebase*.
- **Botão Visualizar Local:** Direciona o administrador ao *Google Maps*, exibindo a localização do reporte com base nos parâmetros de latitude e longitude registrados.

3.3.1.10.2 Integração com Google Maps:

Para permitir a visualização do local do reporte, foi implementada a API do **Google Maps**. Ao clicar no botão “Visualizar Local”, o sistema utiliza os dados de latitude e longitude do reporte e abre o *Google Maps* diretamente na localização correspondente.

Essa funcionalidade facilita a identificação exata do local, permitindo que ações mais precisas sejam tomadas.

A Figura 16 mostra a Tela de Gerenciamento de Reportes.

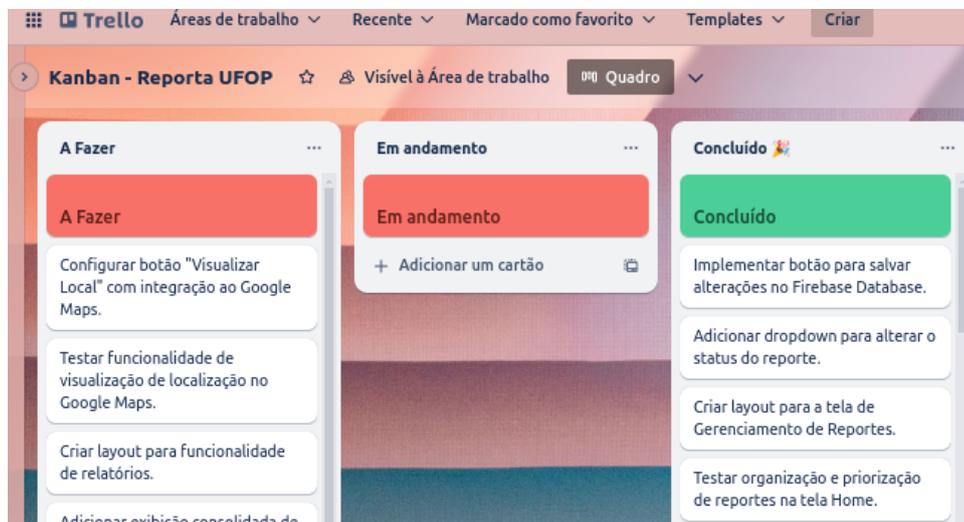


Figura 16 – Tela de Gerenciamento de Reportes

3.3.1.11 Etapa 11: Implementação da Funcionalidade de Relatórios

O foco da décima primeira etapa foi a construção da funcionalidade de **Relatórios** destinada aos administradores. Essa funcionalidade foi projetada para fornecer uma visão consolidada dos status dos reportes inseridos pelo administrador logado no sistema. A Figura 17 apresenta a funcionalidade de Relatórios implementada.

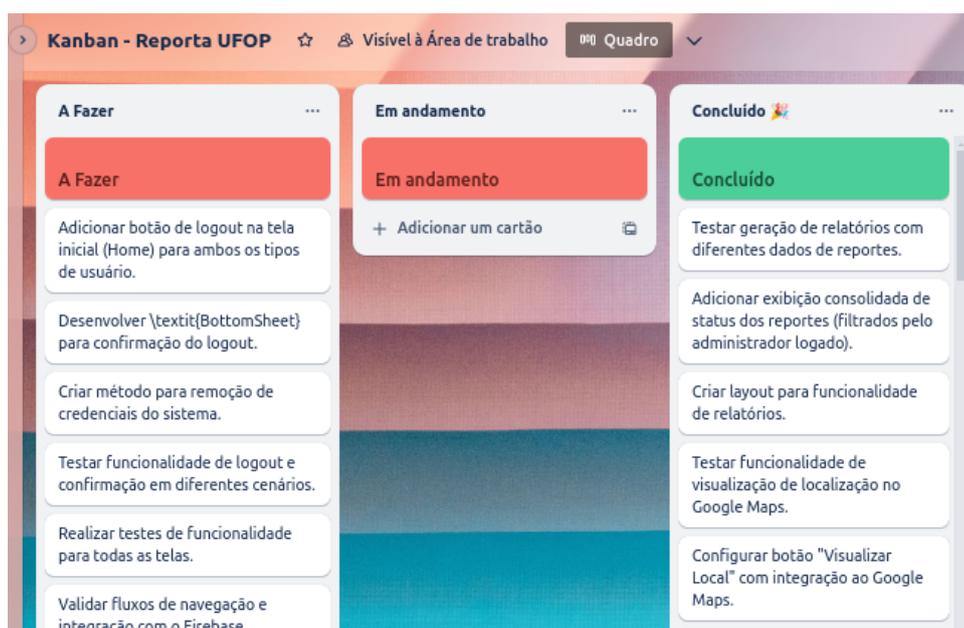


Figura 17 – Funcionalidade de Relatórios

3.3.1.12 Etapa 12: Implementação da Funcionalidade de Logout

A penúltima etapa do desenvolvimento foi dedicada à implementação da funcionalidade de **Logout**, tanto para os administradores quanto para os estudantes/colaboradores. Para isso, foram realizadas as seguintes ações:

3.3.1.12.1 Funcionalidades Implementadas:

- **Botão de Logout:** Adicionado na tela inicial (*Home*) de ambos os tipos de usuários, localizado na parte superior da interface para facilitar o acesso.
- **Confirmação de Logout:** Foi desenvolvido um *BottomSheet* para exibir uma mensagem de confirmação, permitindo que o usuário confirme se realmente deseja encerrar a sessão.
- **Remoção de Credenciais:** Métodos específicos foram criados para remover as credenciais armazenadas no sistema, garantindo a segurança e evitando acesso não autorizado após o logout.

A Figura 18 ilustra essa funcionalidade em execução.



Figura 18 – Funcionalidade de Logout

3.3.1.13 Etapa 13: Testes e Finalização do Projeto

Na última etapa, foram realizados **testes de funcionalidade** para assegurar que todas as implementações atendam aos requisitos especificados no projeto. Os testes focaram nos seguintes aspectos:

- **Verificação de Funcionalidades:** Testes para garantir o funcionamento correto das telas, fluxos de navegação e integração com o Firebase.
- **Testes de Logout:** Validação da remoção de credenciais e funcionamento do *BottomSheet* de confirmação.
- **Testes de Integração:** Garantia de que as funcionalidades de login, gerenciamento de reportes, geração de relatórios e logout operam de forma integrada e sem erros.

Além disso, o código do projeto foi versionado e enviado para o repositório no **GitHub**, com uma documentação completa para facilitar a colaboração de outros desenvolvedores interessados em contribuir para o sistema *Reporta UFOP*. A documentação inclui:

- **Guia de Configuração:** Instruções para configurar o ambiente de desenvolvimento.
- **Descrição das Funcionalidades:** Detalhes sobre cada funcionalidade implementada no sistema.
- **Orientações para Contribuição:** Regras e diretrizes para colaboradores que desejam contribuir com o projeto.

4 Resultados

Este capítulo detalha os resultados obtidos com o desenvolvimento do sistema “Reporta UFOP”, enfocando nas funcionalidades implementadas e os testes realizados para avaliar sua eficácia.

4.1 Arquitetura do Sistema

O projeto foi desenvolvido com base na arquitetura limpa, adaptada às necessidades específicas do sistema. A estrutura foi organizada nas seguintes camadas:

- **Camada de Data:** Responsável pelo gerenciamento de dados, incluindo acesso e persistência.
- **Camada de Domain:** Encarregada da lógica e regras de negócio.
- **Camada de Presenter:** Gerencia a comunicação entre a UI e a Domain, controlando o fluxo de dados.
- **Camada de UI:** Focada na interface do usuário, garantindo responsividade e coerência visual.
- **Core da Aplicação:** Contém configurações cruciais que asseguram a funcionalidade do sistema.

Esta arquitetura facilitou a escalabilidade do sistema, permitindo a adição eficiente de novas funcionalidades. A cor predominante escolhida para o tema da aplicação foi  #962038, alinhada à paleta de cores da UFOP.

4.2 Funcionalidade de Login

As interfaces de login para estudantes e administradores foram desenvolvidas seguindo um design minimalista, de acordo com os protótipos discutidos anteriormente.

4.2.1 Login para Estudantes

Ao inicializar a aplicação, a **Tela de Login** para estudantes apresenta a logomarca da UFOP, seguida de campos para inserção de e-mail e senha. Além disso, disponibiliza botões para “Entrar” e “Cadastrar”:

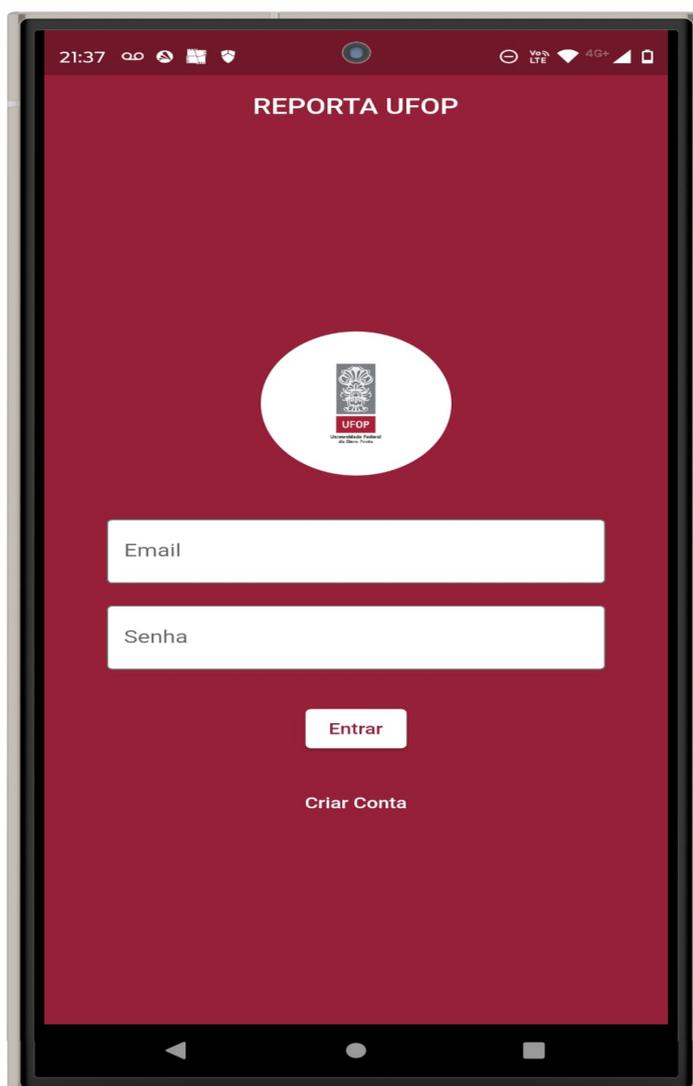


Figura 19 – Tela de login da aplicação móvel para estudantes.

4.2.2 Login para Administradores

De maneira similar, a tela de login para administradores adota o mesmo padrão visual, promovendo uma experiência de usuário coesa e intuitiva:

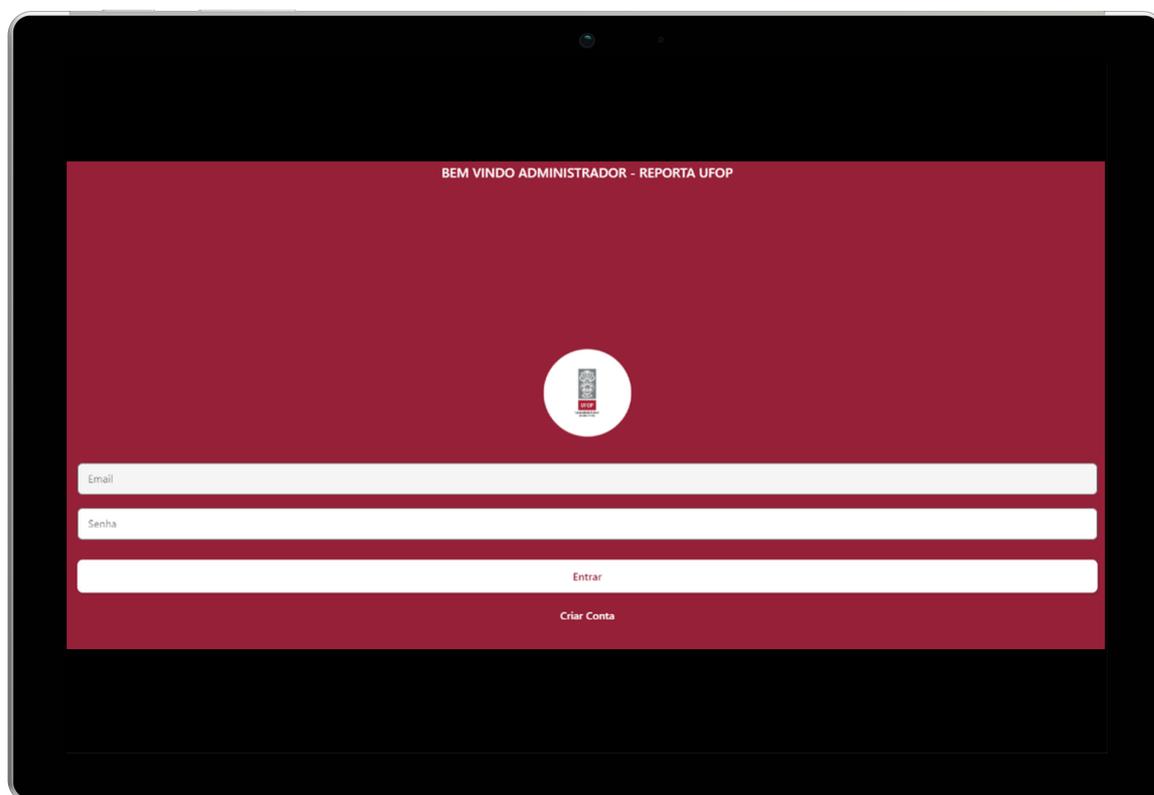


Figura 20 – Tela de login da aplicação móvel para administradores.

Caso o usuário já esteja cadastrado, ele pode inserir seu e-mail e senha e clicar em “Entrar”. O sistema então verificará as credenciais utilizando o Firebase Authentication e, se corretas, permitirá o acesso à tela inicial da aplicação.

4.2.3 Cadastro

Na **Tela de Cadastro**, o usuário deve preencher campos obrigatórios como nome, e-mail, CPF, senha e confirmação de senha. Após completar esses campos, o botão de cadastro estará disponível para concluir o processo:

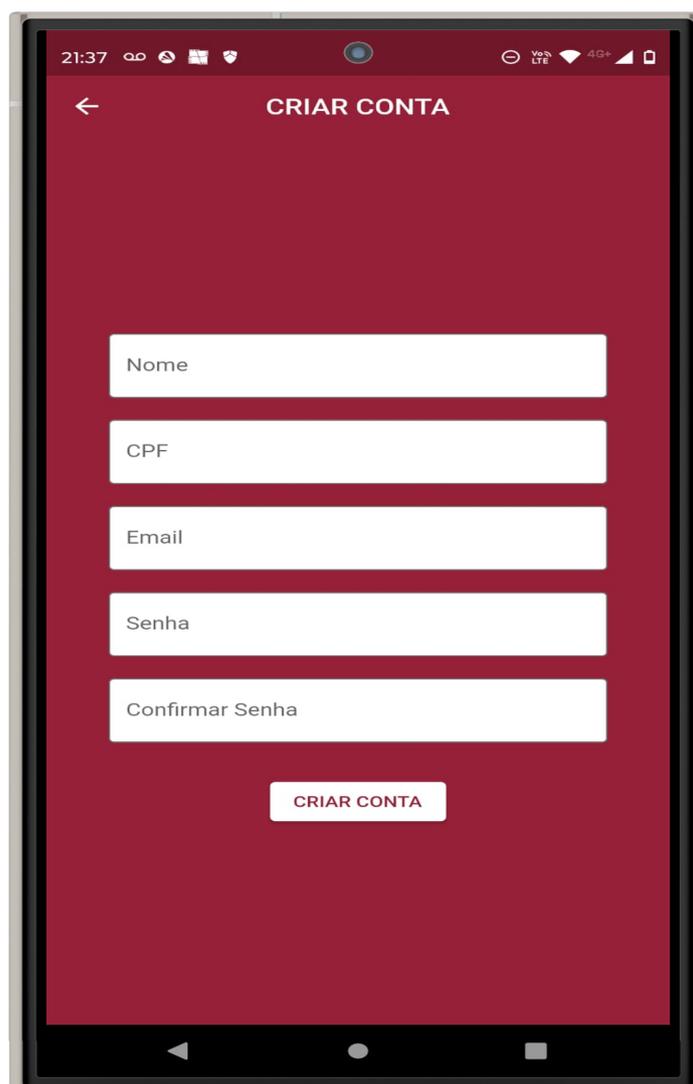


Figura 21 – Tela de Cadastro da Aplicação Móvel para Estudantes.

Uma vez clicado, as informações são registradas no Firebase e o usuário é automaticamente redirecionado para a tela Home.

4.2.4 Home - Estudantes

Na **Tela Inicial (Home)**, após o login, o usuário é recebido com uma mensagem de boas-vindas, que não só cria uma conexão amigável mas também orienta sobre as funcionalidades principais: adicionar novos reportes e visualizar reportes existentes. Esta tela é crucial para a navegação eficiente dentro do aplicativo, permitindo acesso rápido e intuitivo às funções principais:

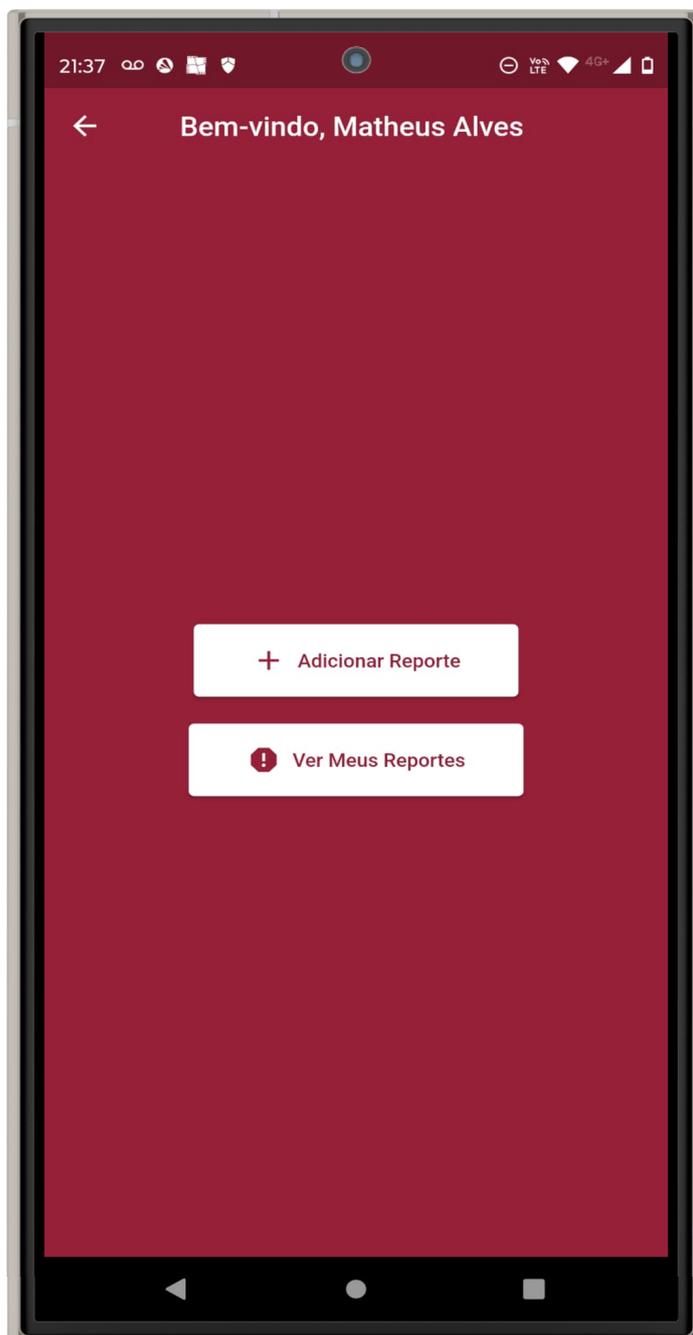


Figura 22 – Tela Inicial (Home) da Aplicação Móvel para Estudantes.

4.2.5 Adicionar Reporte - Estudantes

Esta funcionalidade permite que os estudantes sejam agentes ativos na manutenção e melhoria do campus, facilitando a comunicação de incidentes de maneira imediata e documentada.

Ao selecionar a opção *Adicionar Reporte*, o sistema inicialmente solicitará ao usuário permissão para acessar a câmera do dispositivo e a localização. Esta etapa é fundamental para garantir a privacidade e conformidade com as políticas de uso de dados:

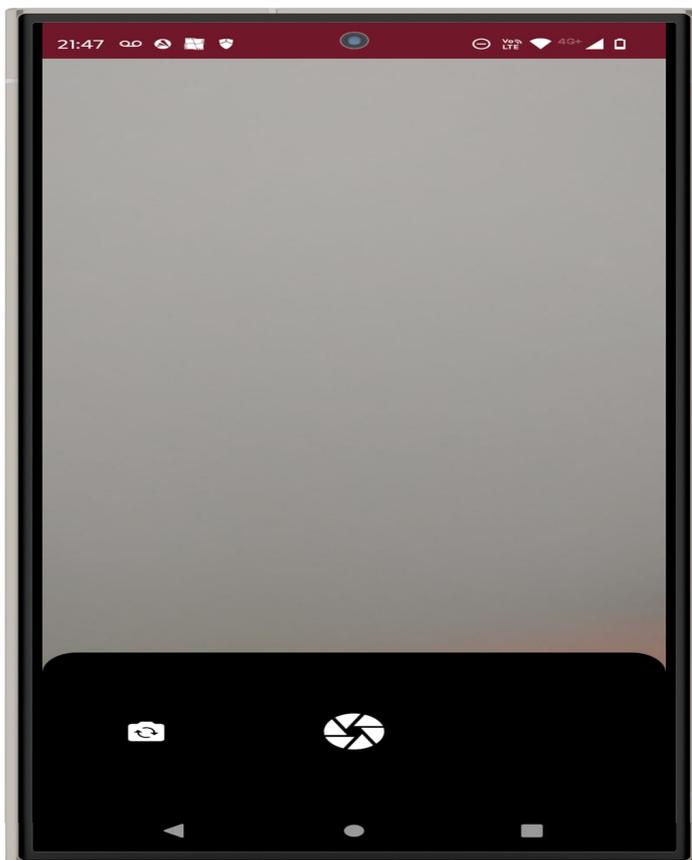


Figura 23 – Fluxo de captura de imagem para reporte.

Uma vez concedida a permissão, o usuário pode capturar a imagem do problema a ser reportado. A captura de imagens direta pelo aplicativo garante que os administradores recebam evidências visuais claras e precisas do problema, o que é crucial para uma avaliação e resolução eficazes.

Após a captura da imagem, o usuário é automaticamente direcionado para a **Tela de Detalhes do Reporte**. Nesta tela, o usuário tem a opção de revisar a imagem capturada, adicionar observações detalhadas sobre o problema e escolher entre capturar outra foto ou salvar o reporte. A possibilidade de adicionar observações permite uma descrição mais rica do problema, facilitando o entendimento e a priorização por parte dos responsáveis pela manutenção:

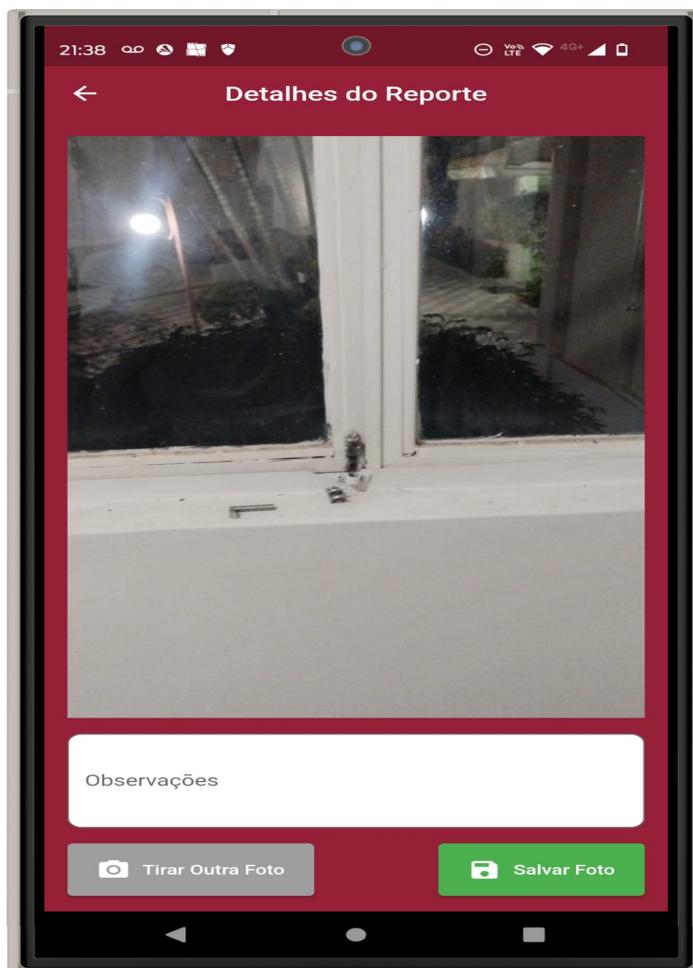


Figura 24 – Tela de Detalhes do Reporte.

Caso o usuário opte por capturar outra foto, ele será redirecionado de volta ao fluxo de captura, permitindo aprimorar a documentação do problema com múltiplas imagens. Se decidir salvar o reporte, a imagem junto com as observações e as coordenadas geográficas do local do registro são enviadas ao Firebase Storage. Este armazenamento no Firebase não só assegura a integridade e a disponibilidade dos dados mas também facilita a gestão e o acesso rápido às informações pelos administradores.

Esta funcionalidade de reporte é crucial para a operacionalidade do sistema, pois permite uma resposta rápida e eficiente aos problemas relatados, melhorando a experiência no campus e garantindo que questões críticas sejam resolvidas prontamente.

4.2.6 Meus Reportes - Estudantes

A funcionalidade *Ver Meus Reportes* desempenha um papel fundamental no engajamento dos estudantes com o ambiente universitário, proporcionando transparência e

acompanhamento contínuo dos problemas reportados.

Ao acessar a **Tela de Meus Reportes**, os estudantes são apresentados com uma lista completa dos incidentes que eles mesmos reportaram. Esta visualização centralizada não apenas confirma que suas preocupações foram registradas, mas também permite acompanhar o progresso na resolução desses problemas, como ilustrado na seguinte figura:

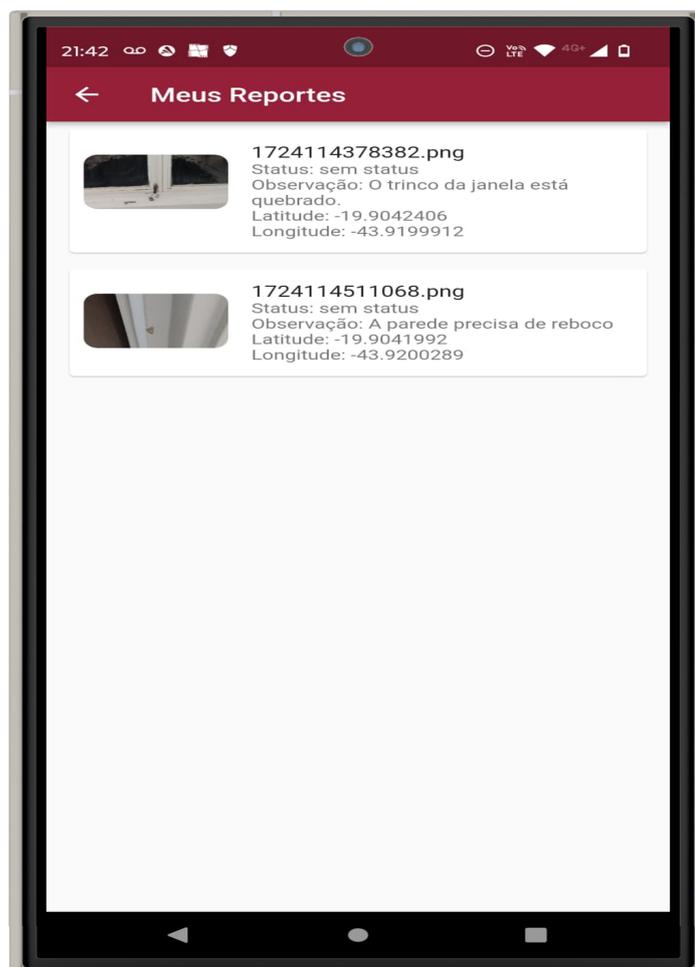


Figura 25 – Tela de Meus Reportes.

Cada reporte listado nesta tela inclui a imagem do incidente reportado, as observações detalhadas fornecidas pelo estudante no momento do reporte e o status atual do problema. O status de cada reporte é atualizado pelos administradores e reflete a atenção dada ao incidente. Os diferentes status atribuídos a cada reporte são:

- **Sem status:** Indica que o reporte ainda não foi avaliado por um administrador, permanecendo em aberto para análise.

- **Resolvido:** Confirma que as medidas necessárias foram tomadas e que o problema foi completamente solucionado.
- **Parcialmente resolvido:** Mostra que houve uma intervenção que mitigou o problema, mas ainda existem aspectos que requerem atenção adicional.
- **Não resolvido:** Reflete que, apesar do reconhecimento do problema, ainda não foi possível encontrar uma solução.

Esta funcionalidade não só melhora a comunicação entre os estudantes e a administração do campus, mas também fortalece a responsabilidade e a resposta da administração aos problemas reportados. Ao fornecer visibilidade sobre o estado de suas solicitações, os estudantes sentem-se mais valorizados e envolvidos na gestão e manutenção do campus, criando um ambiente mais colaborativo e atento às necessidades dos usuários.

4.2.7 Central dos Reportes - Administradores

A **Tela Central dos Reportes** serve como o núcleo operacional para os administradores gerenciarem eficientemente os incidentes reportados dentro do campus. Esta interface centralizada é crucial para manter a organização e agilizar o processo de resposta aos problemas.

Após realizar o *login*, os administradores são direcionados para esta tela, onde são saudados com uma mensagem de boas-vindas. Esta saudação não apenas proporciona uma interação amigável mas também reafirma a importância do papel do administrador na manutenção da qualidade e segurança do campus. A tela apresenta todos os reportes ativos, cada um acompanhado de um botão de interação, permitindo que os administradores avaliem, atualizem o status ou tomem medidas imediatas, conforme ilustrado na Figura 26.

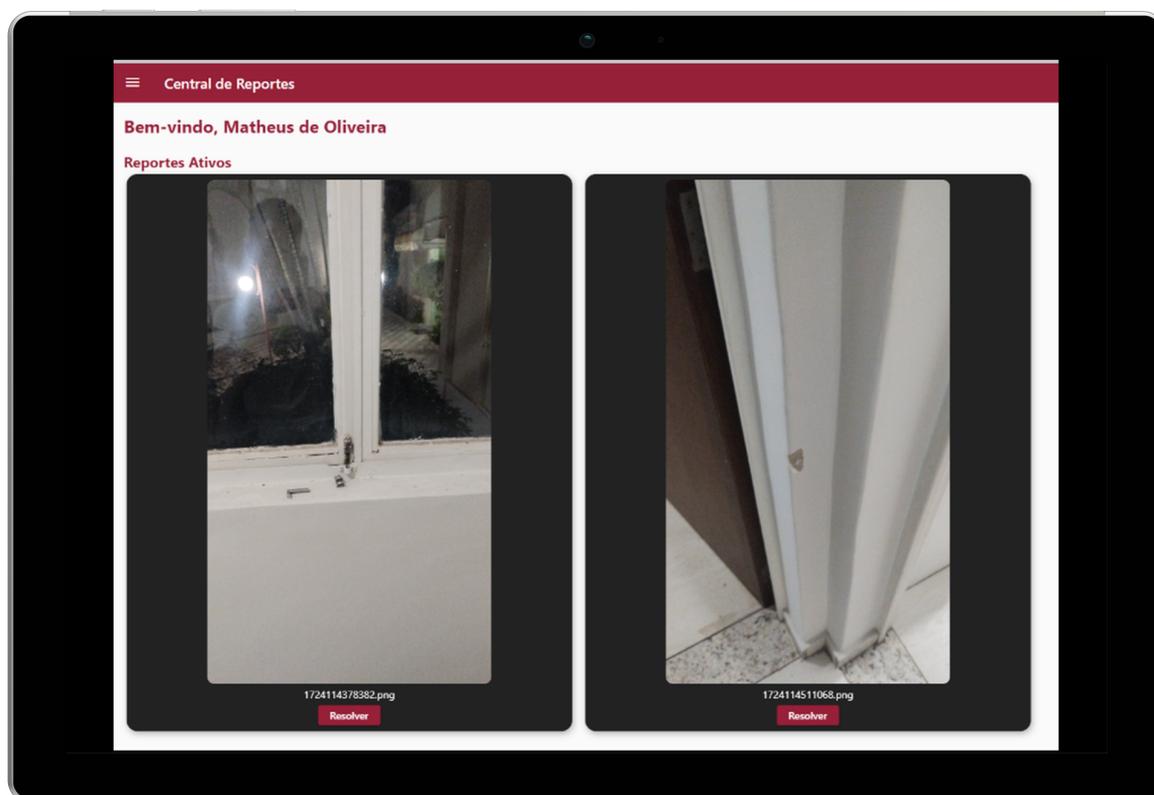


Figura 26 – Tela Central dos Reportes.

Essencialmente, cada reporte ativo pode ser gerenciado individualmente. Isso permite aos administradores uma abordagem detalhada e personalizada para cada situação, aumentando a eficácia das soluções aplicadas. Além disso, um botão de menu, detalhadamente desenhado para ser intuitivo, abre uma lista de opções que classifica os reportes por status: *Resolvido*, *Parcialmente Resolvido* e *Não Resolvido*. Esta classificação facilita o monitoramento dos casos, conforme mostrado na figura seguinte:

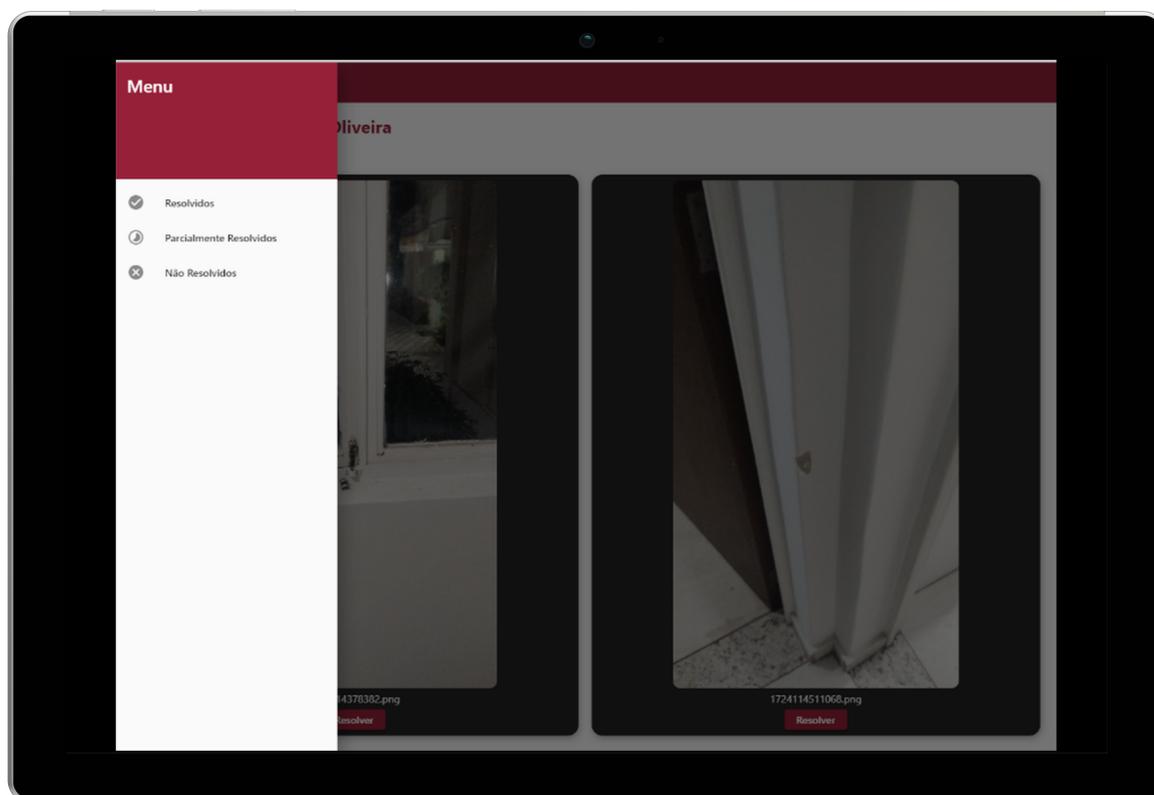


Figura 27 – Tela Central dos Reportes.

Esta funcionalidade de classificação não apenas ajuda na organização dos reportes mas também oferece aos administradores a capacidade de priorizar casos urgentes não foram resolvidos. Ao proporcionar uma visão geral clara e acessível de todos os reportes, a Tela Central dos Reportes é instrumental para a administração na tomada de decisões informadas e na aplicação de recursos de maneira eficaz para resolver problemas e garantir um ambiente seguro e bem mantido para todos os usuários do campus.

4.2.8 Gerenciar Reporte - Administradores

A **Tela de Gerenciar Reporte** é uma ferramenta essencial para que os administradores possam efetivamente intervir e resolver os problemas reportados pelos estudantes.

Ao selecionar o botão de “Resolver” nos reportes ativos da Central de Reportes, o administrador é direcionado para esta tela, onde são exibidos a imagem do incidente, as observações feitas pelo usuário, e opções para atualizar o status do reporte.

Nesta interface, os administradores têm a capacidade de atribuir um dos seguintes status ao reporte: *Resolvido*, indicando que o problema foi completamente corrigido; *Parcialmente Resolvido*, indicando que houve uma intervenção, mas ainda existem questões pendentes; ou *Não Resolvido*, usado quando o problema ainda não foi atendido. A eficiência desta funcionalidade é crucial para manter a transparência e responsabilidade dentro do sistema de gestão de reportes:

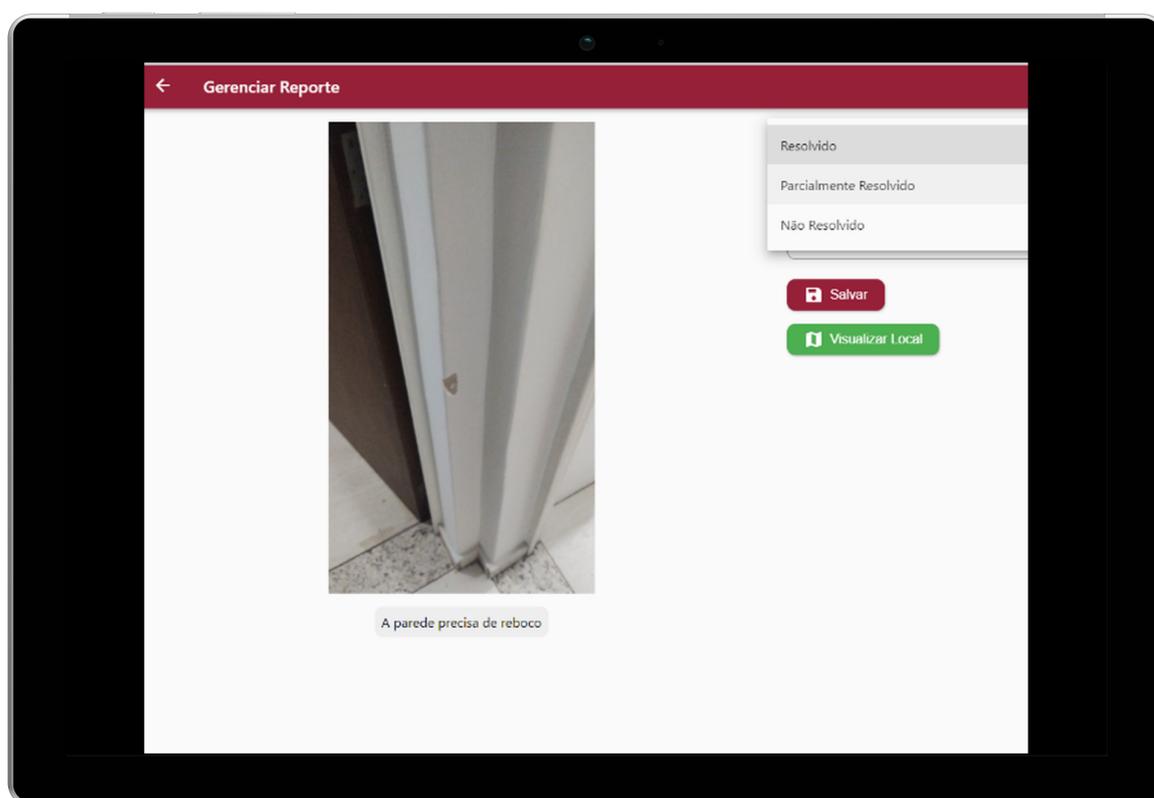


Figura 28 – Tela de Gerenciar Reporte.

Adicionalmente, a tela fornece dois botões principais: um botão de “Salvar”, que confirma e registra o status atribuído, e um botão de “Visualizar Local”, que integra o *Google Maps* para mostrar a localização exata do incidente. Esta integração é instrumental para a rápida localização e resolução dos problemas reportados, aumentando a eficácia das ações tomadas pelos administradores.

4.2.9 Relatório - Administradores

A **Tela de Relatório** é uma importante ferramenta analítica para os administradores, fornecendo um panorama detalhado de todos os reportes com status já atribuído. Esta tela agrega informações críticas como as coordenadas de latitude e longitude do local do reporte, as observações detalhadas, o status atual, a data de atualização do status, e a imagem relacionada ao reporte.

Os dados consolidados nesta tela permitem aos administradores monitorar a efetividade das respostas dadas e identificar padrões ou áreas de recorrência de problemas, facilitando a tomada de decisões estratégicas e a alocação de recursos. Além disso, esta funcionalidade contribui significativamente para o controle de qualidade e a melhoria contínua dos serviços prestados no campus.

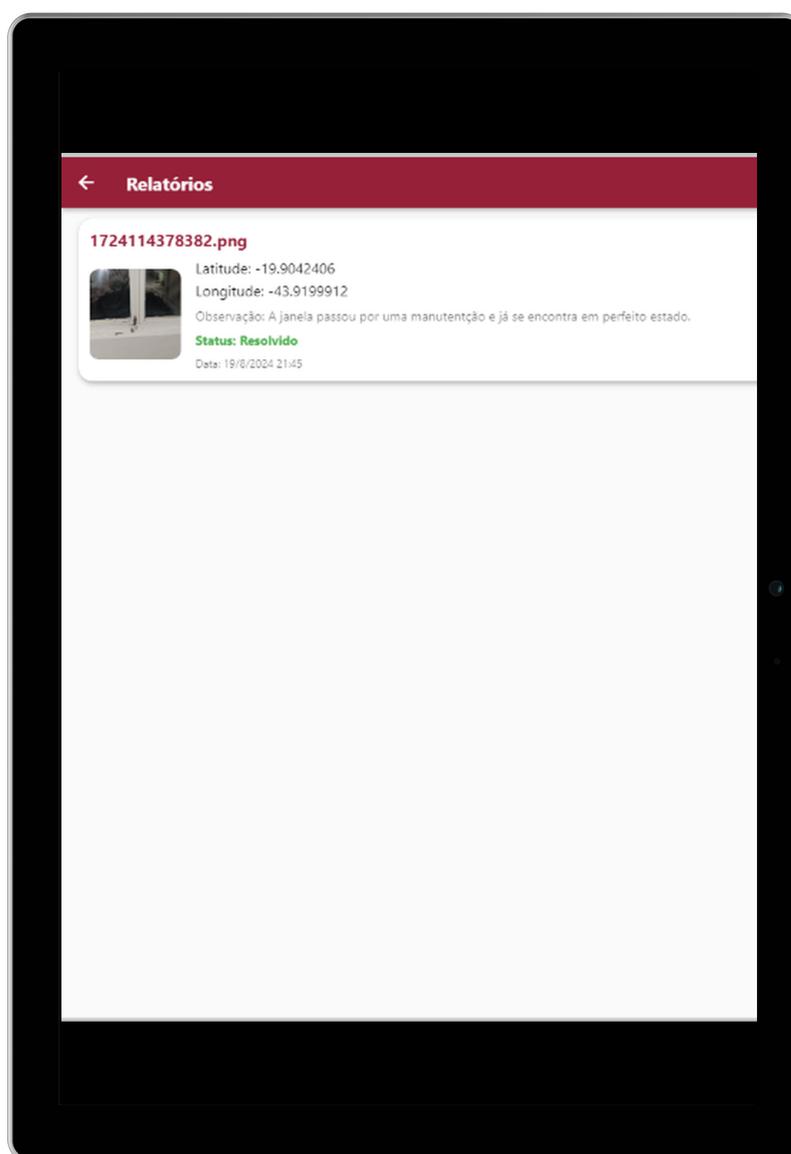


Figura 29 – Tela de Relatório da Aplicação para Administradores.

4.3 Repositório

O software do projeto *Reporta UFOP* foi disponibilizado em um repositório público no GitHub, permitindo acesso aberto ao código-fonte e facilitando a colaboração contínua com a comunidade acadêmica e o público em geral. A escolha do GitHub como plataforma de hospedagem reforça o compromisso com a transparência e a colaboração no desenvolvimento de software. A interface do repositório é ilustrada na Figura 30.

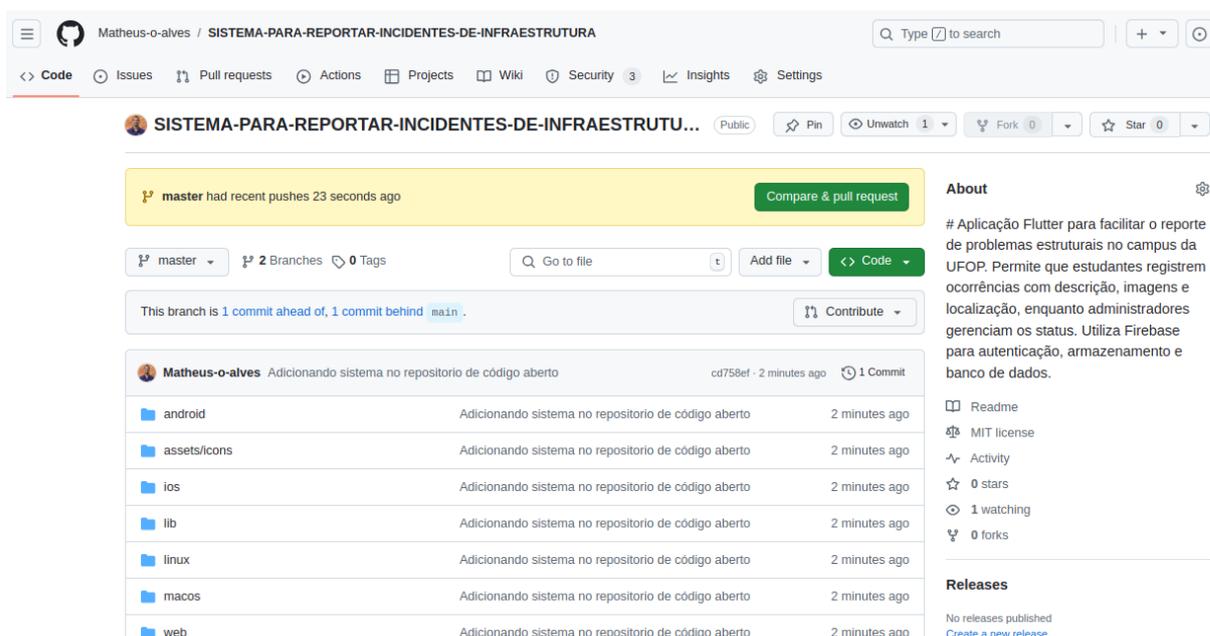
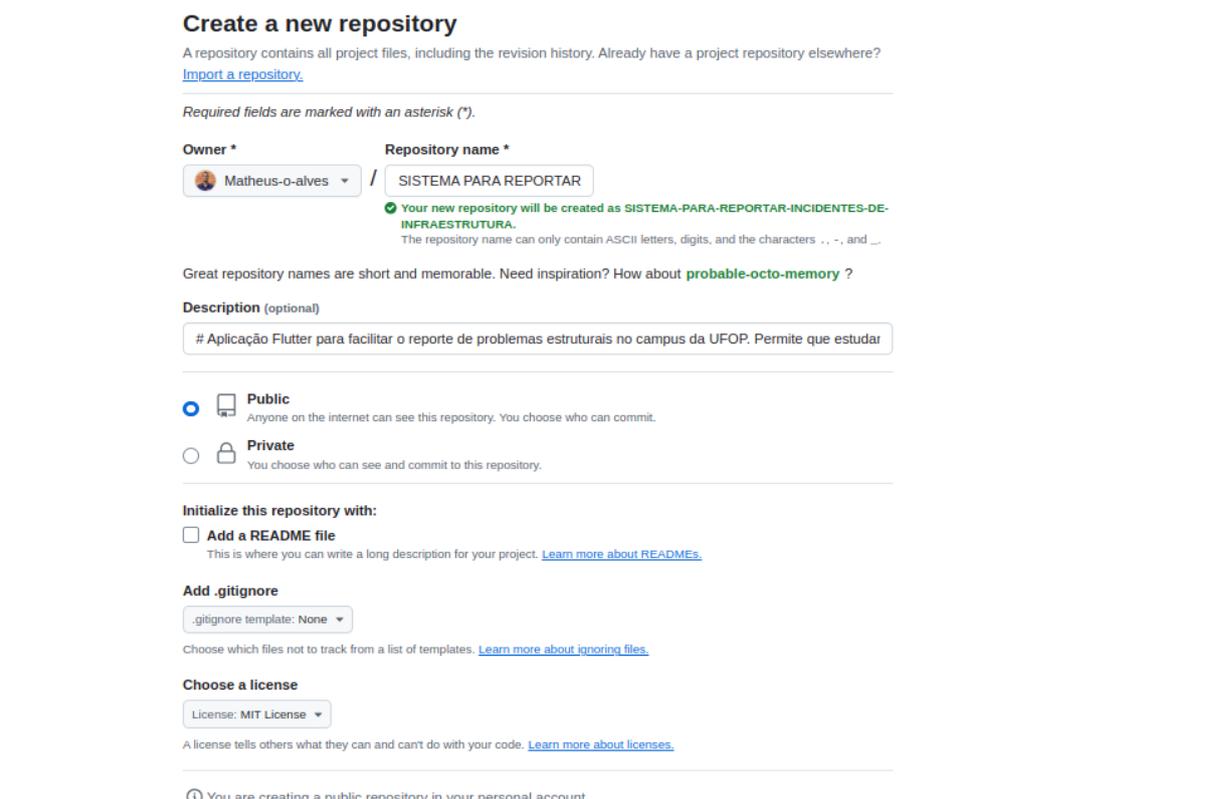


Figura 30 – Repositório do projeto Reporta UFOP no GitHub.

Além disso, o projeto adota a Licença MIT, uma das licenças de *software* mais permissivas, desenvolvida pelo Instituto de Tecnologia de Massachusetts (MIT). Esta licença permite o uso, modificação e distribuição do software sem restrições significativas, promovendo assim a adoção e a evolução do projeto por outros desenvolvedores. A Licença MIT também exige que a licença original seja incluída com qualquer distribuição do *software*, garantindo que os direitos autorais e permissões sejam devidamente reconhecidos, como mostrado na Figura 31.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

 Matheus-o-alves / SISTEMA PARA REPORTAR

✔ Your new repository will be created as SISTEMA-PARA-REPORTAR-INCIDENTES-DE-INFRAESTRUTURA.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about [probable-octo-memory](#) ?

Description (optional)

Aplicação Flutter para facilitar o reporte de problemas estruturais no campus da UFOP. Permite que estudar

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Figura 31 – Licença MIT atribuída ao repositório.

O repositório também inclui uma documentação detalhada, crucial para orientar os usuários na instalação, configuração e operação do *software*. Essa documentação é projetada para ser compreensível e acessível, garantindo que usuários de diferentes níveis técnicos possam utilizar e contribuir para o desenvolvimento do projeto. O repositório pode ser acessado diretamente por meio do <https://github.com/Matheus-o-alves/SISTEMA-PARA-REPORTAR-INCIDENTES-DE-INFRAESTRUTURA/tree/master>, facilitando o acesso rápido ao código-fonte e a todos os recursos relacionados ao projeto.

4.3.1 Benefícios do Sistema para as Universidades

A adoção de um sistema como o *Reporta UFOP* pode trazer diversos benefícios para as universidades:

- **Melhoria na Gestão de Infraestrutura:** Ao permitir que a comunidade acadêmica reporte problemas de forma rápida e organizada, a instituição pode responder de maneira mais ágil às demandas, assegurando um ambiente de ensino mais seguro e funcional.
- **Transparência e Participação:** Estudantes e servidores têm acesso ao status dos reportes, o que promove um sentimento de envolvimento e transparência sobre as

ações tomadas pela administração.

- **Otimização de Recursos:** Com a possibilidade de priorizar e categorizar os incidentes reportados, a instituição consegue alocar recursos de forma mais eficiente, resolvendo primeiro os problemas mais críticos.
- **Monitoramento e Dados Estatísticos:** A ferramenta de relatórios permite à universidade visualizar padrões de incidentes e tomar decisões baseadas em dados confiáveis, auxiliando na identificação de áreas de maior necessidade de investimentos ou melhorias.

5 Considerações Finais

Este trabalho abordou o desenvolvimento de um sistema para simplificar o reporte e gestão de problemas de infraestrutura na Universidade Federal de Ouro Preto (UFOP). Com um enfoque em eficiência e usabilidade, o sistema foi projetado com uma interface minimalista e intuitiva, facilitando a interação dos usuários.

Durante o desenvolvimento, práticas recomendadas de programação foram adotadas, apoiadas por uma extensa revisão de materiais online. A escolha do *framework* Flutter, devido ao seu suporte comunitário ativo, provou ser essencial para superar desafios técnicos e implementar uma solução multiplataforma. O sistema resultante serve tanto a administradores, por meio de uma aplicação web, quanto a discentes, via aplicativo móvel para Android, beneficiando-se de uma base de código amplamente compartilhada para facilitar futuras manutenções e expansões.

Um desafio notável foi a ausência de um sistema MacOS, que limitou os testes ao ambiente Android, impedindo a validação completa em dispositivos iOS. Esta restrição destacou a importância de acessar todos os recursos tecnológicos necessários para um desenvolvimento abrangente multiplataforma.

Embora o lançamento do sistema nas lojas de aplicativos, a realização de testes abrangentes e a implementação de estratégias de marketing não fossem objetivos iniciais do projeto, a inclusão dessas etapas teria potencializado a validação e o impacto do sistema. A falta desses elementos restringiu a possibilidade de obter *feedback* direto dos usuários finais, que seria crucial para iterar e aprimorar o sistema.

O back-end, desenvolvido utilizando Firebase, facilitou significativamente a prototipagem rápida e a escalabilidade. O repositório no GitHub, organizado sob a licença MIT, foi bem documentado, encorajando contribuições futuras, apesar da promoção inicial limitada.

6 Trabalhos Futuros

Para avançar, a implementação do sistema em um ambiente real é crucial. Até o momento, o sistema não foi publicado em hospedagens adequadas nem disponibilizado em lojas de aplicativos, como a Google Play Store. Essa publicação é vital para testar o sistema em cenários reais, coletando feedback valioso para refinamentos futuros.

Propõe-se para trabalhos futuros:

- **Expansão para dispositivos iOS:** Isso aumentaria a acessibilidade, permitindo que usuários de várias plataformas utilizem o sistema.
- **Inclusão de novas funcionalidades:** Por exemplo, notificações em tempo real que informariam discentes e administradores sobre atualizações de status dos reportes.
- **Integração com sistemas institucionais:** Conectar o sistema com recursos já existentes na UFOP para agilizar ainda mais a resolução de problemas.
- **Adaptação para outras instituições de ensino:** Modificar o sistema para que possa ser implementado em diferentes contextos educacionais, ampliando seu impacto e utilidade.
- **Validação de reportes duplicados:** Garantir que o sistema não permita adicionar o mesmo reporte mais de uma vez, evitando redundâncias no cadastro e otimizando a gestão dos incidentes.

Estes aprimoramentos visam não apenas melhorar a funcionalidade do sistema mas também expandir seu alcance e eficácia, contribuindo substancialmente para a infraestrutura acadêmica e a experiência universitária em geral.

Referências

ANDERSON, David J. *Kanban: successful evolutionary change for your technology business*. Sequim: Blue Hole Press, 2010. Citado 2 vezes na página 15.

ANDROID. *Android para todos*. 2024. Acesso em: 22 out. 2024. Disponível em: https://www.android.com/intl/pt-BR_br/everyone/. Citado 2 vezes na página 18.

ARAÚJO, Andréa Cristina Marques de; GOUVEIA, Luis Borges. Uma revisão sobre os princípios da teoria geral dos sistemas. *Estação Científica*, n. 16, jul. 2016. Disponível em: <https://bit.ly/35yCZNv>. Acesso em: 16 nov. 2020. Citado 1 vez na página 15.

BROGNOLI, Tainara da Silva. *Tecnologias da Informação e Comunicação na Gestão Pública: Uma Análise no IFSC - Câmpus Criciúma*. 2018. Acesso em: 7 jan. 2025. Disponível em: <https://repositorio.ifsc.edu.br/bitstream/handle/123456789/782/TCC%20-%20Tainara%20da%20Silva%20Brognoli%20%2826.11.2018%29.pdf>. Citado 1 vez na página 14.

DANTAS, Luiz Fernando; MOREIRA, Juliana Campos; SOUZA, André Luiz. Uso de Aplicativos Móveis Desenvolvidos por Universidades Federais Como Suporte à Gestão Acadêmica. *Revista Acadêmica de Sistemas de Informação*, Universidade Federal Fluminense, Rio de Janeiro, v. 11, n. 1, p. 23–41, 2023. Disponível em: <https://www.rasi.vr.uff.br/index.php/rasi/article/view/720/190>. Citado 3 vezes na página 11.

DEVELOPERS, Android. *Componentes fundamentais do Android*. 2024. Acesso em: 22 out. 2024. Disponível em: <https://developer.android.com/guide/components/fundamentals?hl=pt-br>. Citado 1 vez na página 18.

FERNANDES, José Lúcio Tozetti. *Indicadores para a Avaliação da Gestão das Universidades Federais Brasileiras: Um estudo da influência dos gastos sobre a qualidade das atividades acadêmicas do período 1998-2006*. 2009. Dissertação de Mestrado – Universidade de Brasília. Disponível em: http://icts.unb.br/jspui/bitstream/10482/9100/1/2009_JoseLucioTozettiFernandes.pdf. Citado 1 vez na página 14.

FIGMA. *Figma: A Collaborative Interface Design Tool*. 2025. Acesso em: 26 fev. 2025. Disponível em: <https://www.figma.com>. Citado 1 vez na página 30.

FUNDAÇÃO GETULIO VARGAS. *Pesquisa revela que Brasil tem 480 milhões de dispositivos digitais em uso, sendo 2,2 por habitante*. 2024. Acesso em: 7 jan. 2025. Disponível em: <https://portal.fgv.br/noticias/pesquisa-revela-brasil-tem-480-milhoes-dispositivos-digitais-uso-sendo-22-habitante>. Citado 1 vez na página 14.

GOOGLE. *Flutter Documentation*. Acesso em: 30 set. 2024. 2024. Disponível em: <https://flutter.dev/docs>. Citado 1 vez na página 17.

- GOOGLE TRENDS. *Comparação de Popularidade entre Flutter, React Native, Xamarin, MAUI e Ionic*. 2024. Acesso em: 23 out. 2024. Disponível em: <https://trends.google.com/trends/explore?cat=31&date=2016-03-17%202024-03-17&q=Flutter,React%20Native,Xamarin,MAUI,Ionic>. Citado 1 vez nas páginas 25, 26.
- EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. *Ain Shams Engineering Journal*, v. 8, n. 2, p. 163–190, jun. 2017. ISSN 20904479. DOI: [10.1016/j.asej.2015.12.002](https://doi.org/10.1016/j.asej.2015.12.002). Citado 1 vez na página 17.
- MARTIN, Robert C. *The Clean Code Blog: by Robert C. Martin (Uncle Bob)*. 2012. Acesso em: Junho 2023. Disponível em: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>. Citado 2 vezes nas páginas 18, 32.
- MICROSOFT. *O que é controle de versão?* 2024. Acessado em: 29 de outubro de 2024. Disponível em: <https://learn.microsoft.com/pt-br/devops/develop/git/what-is-version-control>. Citado 1 vez na página 20.
- MICROSOFT. *Visual Studio Code - Editor de Código Gratuito e Poderoso*. 2024. Acesso em: 24 out. 2024. Disponível em: <https://visualstudio.microsoft.com/pt-br/>. Citado 1 vez na página 26.
- OLIVEIRA, Cláudio de; MOURA, Samuel Pedrosa. *TIC's na educação: A utilização das tecnologias da informação e comunicação na aprendizagem do aluno*. 2016. Acesso em: 7 jan. 2025. Disponível em: <https://periodicos.pucminas.br/index.php/pedagogiacao/article/view/11019>. Citado 1 vez na página 14.
- OLIVEIRA, João Pedro de Almeida; BARBOSA NETO, Abdon Senen. *Qualidade em serviços: uma análise da percepção estudantil do curso de Engenharia de Produção (UFOP) por meio do modelo SERVPERF*. 2024. Monografia (Graduação em Engenharia de Produção) – Instituto de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade, 2024. Acesso em: 7 jan. 2025. Disponível em: <https://www.monografias.ufop.br/handle/35400000/6995>. Citado 1 vez na página 13.
- OPEN SOURCE INITIATIVE (OSI). *Open Source Initiative (OSI) Home Page*. 2001. Acesso em: 2001. Disponível em: <http://www.opensource.org>. Citado 1 vez na página 17.
- ORS, Mikel. *Código Limpo — Uma abordagem prática*. Publicado no Medium, Acesso em: 2 out. 2024. Abril 2022. Disponível em: <https://medium.com/clarityai-engineering/clean-code-a-practical-approach-896546435235>. Citado 1 vez na página 16.
- PALHAIS, Catarina Bela Cardoso. *Prototipagem: Uma abordagem ao processo de desenvolvimento de um produto*. 2015. Acesso em: 7 jan. 2025. Disponível em: https://repositorio.ulisboa.pt/bitstream/10451/29163/2/ULFBA_TES_942.pdf. Citado 1 vez na página 29.

SOARES NETO, Joaquim José et al. *Uma escala para medir a infraestrutura escolar*. 2013. Estudos em Avaliação Educacional, vol. 24, no. 54, pp. 78–99. Acesso em: 7 jan. 2025. Disponível em: <https://publicacoes.fcc.org.br/eae/article/view/1903>. Citado 1 vez na página 13.

STAIR, Ralph M.; REYNOLDS, George W. *Princípios de sistemas de informação: uma abordagem gerencial*. Trad. Alexandre Melo de Oliveira. Rio de Janeiro: LTC, 2002. Citado 1 vez na página 15.

TRELLO. *Trello 101 - Trello Guide*. 2024. Acessado em: 12 de novembro de 2024. Disponível em: <https://trello.com/guide/trello-101>. Citado 1 vez na página 27.

UNIVERSIDADE FEDERAL DE OURO PRETO. *Plano de Integridade UFOP 2023*. 2023. Acesso em: 25 set. 2024. Disponível em: https://www.cppo.ufop.br/sites/cppo/files/versao_final_plano_de_integridade_ufop_2023_0.pdf. Citado 1 vez na página 11.

GOOGLE. *Flutter Documentation*. Disponível em: <https://flutter.dev/docs>. Acesso em: 30 set. 2024.