UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

FRANCISCO JERFERSON MARTINS DA SILVA

REFATORAÇÃO E ESCALABILIDADE NO QUIXALERT!: UM PROJETO DE SOFTWARE PARA MONITORAMENTO E GESTÃO AMBIENTAL URBANA

QUIXADÁ

2025

FRANCISCO JERFERSON MARTINS DA SILVA

REFATORAÇÃO E ESCALABILIDADE NO QUIXALERT!: UM PROJETO DE SOFTWARE PARA MONITORAMENTO E GESTÃO AMBIENTAL URBANA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Profa. Ma. Antonia Diana Braga Nogueira.

QUIXADÁ

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58r Silva, Francisco Jerferson Martins da.
Refatoração e escalabilidade no QuixAlert!: um projeto de software para monitoramento e gestão ambiental urbana / Francisco Jerferson Martins da Silva. – 2025.
140 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2025.
Orientação: Profa. Ma. Antonia Diana Braga Nogueira.

1. Engenharia de software. 2. Arquitetura de software. 3. Refatoração. 4. Modularização. 5. Software cívico. I. Título.

CDD 005.1

FRANCISCO JERFERSON MARTINS DA SILVA

REFATORAÇÃO E ESCALABILIDADE NO QUIXALERT!: UM PROJETO DE SOFTWARE PARA MONITORAMENTO E GESTÃO AMBIENTAL URBANA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: 17/11/2025.

BANCA EXAMINADORA

Profa. Ma. Antonia Diana Braga Nogueira (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulyne Matthews Jucá
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Armando Cavalcante Aguilar
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Aos meus pais, Adeuzon França e Maria Luisa, à minha irmã, Maria Lorane, à minha avó, Maria Dilurdes, e à minha madinha, Viviane, por serem meu alicerce inabalável, a razão da minha força e a fonte do meu amor. Por todos os sacrifícios, incentivos, por todos os cuidados e por acreditarem no meu potencial, mesmo quando eu duvidei. Este trabalho é fruto da semente que vocês plantaram.

Ao meu amor, Larissa Alves, por ser meu farol e refúgio seguro. Por me acompanhar e me apoiar incondicionalmente em toda esta trajetória, oferecendo a paciência e a compreensão necessárias nos momentos de maior pressão. Sua presença leve e incentivadora foi a minha grande fonte de inspiração e motivação diária, fundamental para que eu pudesse manter o equilíbrio e alcançar esta vitória. Seu cuidado foi essencial para garantir que eu chegasse até aqui bem. Eu amo tudo em você!

Ao meu inseparável "Trio do Caos-- Ariadne, Maria Luisa e Guilherme. Vocês foram a melhor desorganização que essa jornada poderia me dar. Por serem o refúgio imediato, a dose de insanidade controlada e os cúmplices perfeitos nas fugas necessárias. Em cada surto, em cada crise e nos incontáveis cafés da tarde, vocês me lembraram que a universidade é apenas uma fase e que a nossa amizade é o que realmente importa e que é para além da vida. Guilherme com a sua sensibilidade, você me ensinou que as coisas mais belas do mundo, podem ser encontradas nos momentos mais simples vividos. Ariadne com sua inabalável resistência e coragem, aprendi com você que para superar as mais complexas adversidades da vida, é preciso mais do que força, é preciso bons amigos. Maria Luisa, com sua incrível sabedoria, todos nós aprendemos milhares de coisas com você, mas talvez a principal lição seja que a vida não precisa ser pesada nem artificial; é possível vivê-la de forma leve e única, desde que se busque sempre o lado bom de tudo. Obrigado por transformarem o peso da academia em risadas altas e inesquecíveis. Vocês são o alívio, a energia e o sorriso que me permitem avançar cada vez mais longe.

Aos meus amados amigos, Daniel Lucas, Daniel Christopher, Kauan e Marcos Nascimento, por serem os amigos de ouro que me acompanharam de perto nesta desafiadora jornada. Mais do que amigos, vocês foram parceiros de resistência em vários momentos, que compartilharam a tensão comigo e, principalmente, ofereceram as risadas e o suporte emocional para que a pressão não me derrubasse. A amizade e a lealdade de vocês foram vitais para que eu pudesse manter a saúde mental, o foco e a motivação necessários para a conclusão deste trabalho.

Ao meu irmão de outra mãe, Kleyton Ferreira, por ser a prova de que laços de alma

superam os de sangue. Sua amizade inestimável e o incentivo constante foram um pilar de força nos momentos mais difíceis da graduação. Obrigado por me lembrar sempre da minha capacidade, por me ajudar a manter o foco e por ser esse ponto de apoio que torna a vida e a jornada acadêmica muito mais leve e possível. Toda admiração, respeito e carinho por você meu irmão!

A minha querida orientadora, Diana Braga, sem a senhora, nada disso seria possível. Seu acompanhamento desde o início da graduação sempre foram além de um cuidado de professora, sempre teve o carinho de uma mãe, fazendo juz ao pseudônimo "Mãe-Diana", cada conselho, conversa e sermões foram fundamentais. Obrigado!

Aos professores Paulo Armando e Paulyne Matthews Jucá, pela contribuição inestimável na construção deste trabalho. Suas orientações, sugestões e disponibilidade foram essenciais para o amadurecimento das ideias aqui apresentadas, enriquecendo o processo com conhecimento, sensibilidade e rigor acadêmico.

À UFC, em especial à PRAE, por ter me possibilitado, nos momentos mais difíceis, continuar com meus estudos, oferecendo suporte, auxílio e incentivo quando mais precisei.

RESUMO

A gestão ambiental em municípios de pequeno e médio porte no Brasil enfrenta desafios estruturais, frequentemente agravados pela ausência de canais de comunicação eficazes entre a população e o poder público. Nesse contexto, foi desenvolvido o aplicativo ***QuixAlert!***no município de Quixadá-CE, como uma solução digital para facilitar denúncias e solicitações de serviços ambientais. No entanto, sua implementação inicial com uma arquitetura monolítica impôs sérias limitações quanto à manutenção, evolução e escalabilidade da aplicação. Este trabalho apresenta o processo de engenharia de software voltado à evolução arquitetural do *QuixAlert!*, com o objetivo de transformá-lo em uma plataforma mais robusta, manutenível e adaptável. A abordagem metodológica adotada caracteriza-se como um projeto de desenvolvimento tecnológico aplicado, composto por três etapas: (1) diagnóstico técnico e funcional da versão original; (2) refatoração sistemática para redução da dívida técnica; e (3) migração para uma arquitetura modular baseada em Android Nativo (Kotlin), com a inclusão de novos módulos. Como principal resultado, espera-se uma nova versão do *QuixAlert!* com estrutura modular, maior segurança, facilidade de manutenção e novas funcionalidades. A principal contribuição deste trabalho é oferecer um estudo de caso detalhado sobre a modernização de um software cívico, demonstrando como práticas de refatoração e modularização impactam positivamente na qualidade do produto e viabilizam sua escalabilidade.

Palavras-chave: Engenharia de software; Arquitetura de software; Refatoração; Modularização; Aplicações móveis; Software cívico

ABSTRACT

Environmental management in small and medium-sized municipalities in Brazil faces structural challenges, often worsened by the lack of effective communication channels between the population and public authorities. In this context, the ***QuixAlert!*** application was developed in the municipality of Quixadá-CE as a digital solution to facilitate environmental service requests and citizen reports. However, its initial implementation with a monolithic architecture presented serious limitations in terms of maintenance, evolution, and scalability. This work presents the software engineering process aimed at the architectural evolution of *QuixAlert!*, with the goal of transforming it into a more robust, maintainable, and adaptable platform. The adopted methodology is characterized as an applied technological development project, comprising three stages: (1) technical and functional diagnosis of the original version; (2) systematic refactoring to reduce technical debt; and (3) migration to a modular architecture based on native Android (Kotlin), with the addition of new modules. As a main result, a new version of *QuixAlert!* is expected, with a modular structure, improved security, easier maintenance, and new features. The main contribution of this work is to present a detailed case study on the modernization of a civic software application, demonstrating how refactoring and modularization practices positively impact product quality and enable its strategic scalability.

Keywords: Software engineering; Software architecture; Refactoring; Modularization; Mobile applications; Civic tech.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AIA	Avaliação de Impacto Ambiental
AMMA	Autarquia Municipal do Meio Ambiente
APP	Aplicativo
CEMADEN	Centro Nacional de Monitoramento e Alertas de Desastres Naturais
CPU	Unidade Central de Processamento
DETER	Detecção de Desmatamento em Tempo Real
DRY	*Don't Repeat Yourself* (Não se Repita)
ESC	Escritório de Soluções Criativas
IBAMA	Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis
INPE	Instituto Nacional de Pesquisas Espaciais
IoT	*Internet of Things* (Internet das Coisas)
ODS	Objetivos de Desenvolvimento Sustentável
ONGS	Organizações Não Governamentais
PRODES	Projeto de Monitoramento do Desmatamento na Amazônia Legal por Satélite
RAM	Memória de Acesso Aleatório
SGA	Sistema de Gestão Ambiental
SIG	Sistema de Informações Geográficas
SINAFLOR	Sistema Nacional de Controle da Origem dos Produtos Florestais
SRP	*Single Responsibility Principle* (Princípio da Responsabilidade Única)
SUS	System Usability Scale (Escala de usabilidade do sistema)
TCC	Trabalho de Conclusção de Curso
TICs	Tecnologias da Informação e Comunicação

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, o crescimento desordenado dos centros urbanos, aliado à limitação de recursos públicos e à ausência de planejamento ambiental adequado, tem intensificado os desafios encontrados por cidades de pequeno e médio porte no Brasil (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2020; Maricato, 2011). Problemas como o descarte irregular de resíduos, queimadas em áreas urbanas e periurbanas, poluição sonora, alagamentos e ocupações irregulares de áreas verdes são cada vez mais recorrentes e impactam diretamente a qualidade de vida da população(Acselrad, 2004). De acordo com Braga *et al.* (2024), o processo de urbanização acelerada sem diretrizes sustentáveis tem agravado desequilíbrios socioambientais nos municípios, exigindo soluções integradas e políticas públicas que considerem a preservação dos recursos naturais como parte essencial do planejamento urbano.

Cordeiro e Ruela (2024) analisam os desafios da governança pública local em cidades de pequeno porte e destacam que a gestão pública, especialmente em municípios de pequeno e médio porte, enfrenta sérias limitações para responder de forma ágil e eficiente às demandas ambientais e urbanas cotidianas. Essas administrações frequentemente operam com recursos financeiros escassos, déficit de pessoal técnico qualificado e ausência de sistemas informatizados, o que compromete a capacidade de fiscalização e resposta a ocorrências ambientais. Em muitas dessas cidades, as denúncias ainda são feitas de maneira informal, por meio de ligações telefônicas, mensagens em redes sociais ou atendimentos presenciais, o que gera dificuldades no rastreamento de casos, na priorização de ações e na transparência do atendimento à população. Essas fragilidades estruturais são características comuns de prefeituras de pequeno porte no Brasil.

Nesse contexto, surge a importância de soluções digitais voltadas à gestão urbana e ambiental, especialmente aquelas que permitem a participação ativa da população na identificação e denúncia de ocorrências. Tais ferramentas fortalecem a governança colaborativa e contribuem para a construção de cidades mais sustentáveis, resilientes e inclusivas, conforme previsto nos Objetivos de Desenvolvimento Sustentável (*ODS*) (ORGANIZAÇÃO DAS NAÇÕES UNIDAS, 2015), em especial o ODS 11 — Cidades e Comunidades Sustentáveis.

Nesse cenário, a tecnologia e os sistemas de informação assumem um papel estratégico na gestão pública, ao fortalecer a participação ativa da população, que tem se mostrado cada vez mais essencial para a promoção de uma gestão ambiental eficiente e democrática, especialmente em contextos urbanos onde os recursos públicos são limitados e a fiscalização, muitas

vezes, insuficiente. Ao permitir que as cidades atuem como agentes de monitoramento e denúncia de irregularidades ambientais, cria-se um ambiente de corresponsabilidade, no qual a sociedade civil colabora diretamente com o poder público na proteção dos espaços urbanos e naturais. Segundo Leal *et al.* (2024), o envolvimento popular é indispensável para fortalecer políticas públicas ambientais, sendo a mobilização comunitária um fator que contribui significativamente para o reconhecimento, a visibilidade e a resolução de problemas ambientais cotidianos.

Em cidades de pequeno e médio porte, a participação cidadã pode ser decisiva para a identificação de problemas que passariam despercebidos pela gestão local(Garcia; Alves, 2020). Ao vivenciarem o cotidiano das comunidades, os moradores se tornam observadores privilegiados e podem oferecer informações valiosas de forma imediata. Entretanto, para que essa participação seja eficaz, é necessário que existam canais acessíveis, seguros e organizados para o envio dessas informações, bem como mecanismos de retorno e acompanhamento por parte das autoridades (Costa *et al.*, 2016). A falta de sistemas estruturados de denúncia e de retorno ao cidadão pode enfraquecer a confiança nas instituições e desestimular a participação, tornando essenciais ferramentas digitais que promovam transparência, interação e engajamento ambiental Giaretta *et al.* (2020); Silva *et al.* (2022).

Essas iniciativas de participação cidadã alinham-se também aos princípios de governança colaborativa, ampliando a capacidade do Estado de responder às demandas sociais por meio de soluções integradas. Além disso, fortalecem valores como pertencimento, cidadania e sustentabilidade, gerando impactos positivos tanto no meio ambiente quanto no tecido social das cidades (Branco *et al.*, 2023).

Apesar do crescente reconhecimento da importância da participação cidadã na identificação e denúncia de problemas ambientais, a eficácia dessa colaboração depende diretamente das condições locais de estrutura e comunicação (Liu *et al.*, 2023). Em cidades de pequeno e médio porte, onde há limitações tanto técnicas quanto operacionais, o Garcia e Alves (2020), afirma que a ausência de ferramentas adequadas pode enfraquecer esse elo entre população e poder público, comprometendo o potencial transformador da ação coletiva,

É nesse cenário que se insere a realidade de Quixadá, município localizado no interior do Ceará, que compartilha muitos dos desafios enfrentados por outras cidades brasileiras com perfil socioeconômico semelhante. Com um território extenso e características urbanas e rurais, Quixadá enfrenta dificuldades na gestão e no monitoramento ambiental, agravadas por restrições orçamentárias, escassez de profissionais especializados e baixa aderência da modernização dos

processos internos. Problemas como descarte inadequado de lixo, queimadas, poluição sonora, ocupações irregulares e maus-tratos a animais são frequentemente relatados, mas nem sempre tratados com a celeridade necessária, seja por falta de denúncias formalizadas ou por limitações na capacidade de resposta do poder público. Alidado a isso, o fluxo de informações entre cidadãos e a autarquia ambiental local — a Autarquia Municipal do Meio Ambiente (*AMMA*) — ocorria majoritariamente de forma informal, dificultando o registro, o acompanhamento e a tomada de decisões baseada em dados concretos.

A partir desse conjunto de necessidades práticas, foi desenvolvido o *QuixAlert!*, uma proposta tecnológica voltada para suprir essas lacunas. Em parceria com a *AMMA*, o sistema foi idealizado para criar um ambiente digital de denúncia, informação e acompanhamento, com foco na simplificação da comunicação entre a população e a gestão ambiental local.

A primeira versão funcional do *QuixAlert!* foi implementada e validada, entregando os módulos essenciais de denúncias, adoção de animais e notícias. No entanto, o diagnóstico técnico realizado no início deste trabalho identificou que a implementação inicial com uma arquitetura monolítica concentrava todas as funcionalidades em um único módulo, resultando em um alto grau de acoplamento entre as camadas. Essa estrutura levou ao aumento da Dívida Técnica, manifestada por *Code Smells* excessivos e baixa coesão, o que impôs sérias limitações na manutenibilidade, evolução e escalabilidade da aplicação.

Para que o sistema pudesse crescer de forma sustentável, suportando novas funcionalidades e sendo replicável em outros contextos, tornou-se imperativo focar na qualidade de sua estrutura interna.

No ciclo de vida de um software como o *QuixAlert!*, a manutenção contínua é crucial para garantir sua longevidade e capacidade de adaptação a novas demandas. Nesse contexto, este trabalho aborda a evolução da aplicação, tratando-a como uma ação de manutenção preventiva e evolutiva (Sommerville, 2020).

A intervenção foi realizada em duas frentes técnicas principais. Primeiramente, através da refatoração, buscou-se aprimorar a qualidade interna do código-fonte para torná-lo mais legível, organizado e de fácil manutenção, gerenciando a "dívida técnica" acumulada e facilitando a introdução de novas funcionalidades.

De forma paralela, foi executado um processo de modularização, uma etapa estratégica para reorganizar a arquitetura do sistema em componentes funcionais independentes. Essa abordagem arquitetural, conforme apontam Bass *et al.* (2021), não só simplifica a evolução

sustentável do software, mas também viabiliza o objetivo de longo prazo de tornar o *QuixAlert!* uma plataforma escalável, permitindo que outros municípios adotem apenas os módulos que atendem às suas necessidades específicas — como denúncias, adoção de animais ou notícias.

1.1 Estrutura do trabalho

No capítulo 2 desse trabalho está apresentado a fundamentação teórica, abordando os principais conceitos sobre gestão ambiental, participação cidadã ativa, sistemas de informação na gestão pública, softwares cívicos, refatoração, manutenção e processo de modularização de softwares.

No capítulo 3 está apresentado trabalhos semelhantes que abrangem os três aspectos apresentados neste trabalho: softwares cívicos ambientais, participação ativa dos cidadãos, refatoração e modularização de softwares.

No capítulo 4 apresenta a metodologia e o desenvolvimento do projeto, detalhando a metodologia utilizada, passando pelo diagnóstico da versão inicial do *QuixAlert!*, todo o processo de intervenção (refatoração, modularização e implementação de novas funcionalidades) e o planejamento da validação final.

O Capítulo 5 apresenta e discutirá os resultados obtidos, principalmente do processo de refatoração, modularização, adição de novas funcionalidades e avaliação de usabilidade.

O Capítulo 6 trás uma síntese dos resultados e os próximos passos do projeto.

1.2 Objetivos do Trabalho

1.2.1 Objetivo Geral

O objetivo deste trabalho é evoluir a aplicação QuixAlert! por meio dos processos de refatoração e modularização, com o intuito de melhorar sua qualidade interna, ampliar suas funcionalidades com base em necessidades reais da gestão ambiental urbana, e torná-la escalável e adaptável para aplicação em diferentes municípios e organizações.

1.2.2 Objetivos Específicos

1. Analisar a arquitetura e as funcionalidades do *QuixAlert!*, englobando a identificação de limitações e oportunidades técnicas de melhoria.

2. Realizar a refatoração do código do *QuixAlert!* com foco na melhoria da qualidade interna, legibilidade, manutenibilidade e a preparação para a modularização.
3. Expandir o conjunto de funcionalidades do *QuixAlert!*, tendo o foco em adicionar novas funcionalidades com base em requisitos e demandas, mostrando o impacto na capacidade do sistema.
4. Modularizar a aplicação QuixAlert!, destacando os benefícios dessa aboradagem modular.
5. Validar a percepção dos *stakeholders* e usuários finais quanto à usabilidade e impacto das modificações realizadas no *QuixAlert!*.
6. Documentar o processo e os resultados obtidos por todas as etapas da evolução e modularização do *QuixAlert!*, bem como o resultado da avaliação de usabilidade.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo reúne os fundamentos teóricos que sustentam o desenvolvimento e a evolução do *QuixAlert!*. São discutidos os desafios da gestão ambiental urbana em cidades de pequeno e médio porte, a importância da participação cidadã e da governança colaborativa, bem como o papel dos sistemas de informação na gestão pública e o uso de softwares cívicos. Além disso, são apresentados os conceitos técnicos de engenharia de software, com ênfase em refatoração, manutenção evolutiva e modularização, que embasam as melhorias aplicadas ao sistema para torná-lo mais eficiente, escalável e adaptável a diferentes contextos municipais.

2.1 Gestão ambiental urbana e os principais desafios

A gestão ambiental pode ser compreendida como o conjunto de ações, políticas e estratégias voltadas para o uso racional dos recursos naturais e para a mitigação dos impactos ambientais causados pelas atividades humanas (Medeiros, 2023). Complementando essa visão, Dias (2015), define como um processo contínuo de planejamento, implementação, monitoramento e avaliação de práticas que visam integrar o desenvolvimento socioeconômico à preservação ambiental. A gestão ambiental abrange desde ações governamentais até iniciativas privadas e comunitárias, sendo fundamental para o alcance de um modelo de desenvolvimento sustentável.

No contexto urbano, o conceito de gestão ambiental se amplia e adquire novas dimensões. A gestão ambiental urbana envolve a formulação e aplicação de políticas públicas voltadas ao controle da poluição, ao manejo adequado dos resíduos sólidos, à conservação de áreas verdes, ao uso racional da água e à garantia de qualidade de vida para a população (Silva *et al.*, 2020). Entretanto, a gestão ambiental no contexto urbano enfrenta uma série de desafios complexos, muitos dos quais são derivados do rápido crescimento urbano e das pressões socioeconômicas. De acordo com Miranda e Decesaro (2018), a urbanização acelerada tem gerado um desequilíbrio entre a demanda por espaço e recursos e a capacidade das cidades de absorver esse crescimento de forma sustentável. Esses desafios ambientais são particularmente exacerbados em cidades de pequeno e médio porte, como Quixadá, onde a capacidade institucional limitada, a escassez de recursos técnicos e financeiros e a carência de infraestrutura para monitoramento e fiscalização ambiental são evidentes (Braga *et al.*, 2024).

Um dos principais desafios é o descarte inadequado de resíduos sólidos. Em muitas cidades, o tratamento de lixo é ineficiente, levando à proliferação de lixões a céu aberto, conta-

minação de corpos d'água e emissão de gases de efeito estufa. Pereira *et al.* (2024), a falta de políticas públicas adequadas para a gestão de resíduos e a escassez de sistemas de coleta seletiva contribuem para a degradação ambiental, afetando a saúde pública e a qualidade do ambiente urbano.

A ineficiência no manejo desses resíduos não se limita à contaminação gradual, mas também se manifesta em eventos catastróficos (CETESB – Companhia Ambiental do Estado de São Paulo, 2007). A ausência de uma resposta e manejo correto dos resíduos sólidos pode ser um fator desencadeante de desastres ambientais, como inundações urbanas agravadas pelo entupimento de bueiros e corpos d'água, deslizamentos de terra em áreas de descarte irregular, ou incêndios em lixões que liberam poluentes tóxicos na atmosfera (Borgo, 2007). Tais desastres resultam em perdas financeiras significativas, danos à infraestrutura e, tragicamente, em perdas de vidas, com impactos desproporcionais em comunidades vulneráveis, especialmente em países em desenvolvimento. A gestão de riscos e desastres, portanto, precisa ser vista como uma atividade de responsabilidade compartilhada entre governos, sociedade civil e corporações, exigindo planejamento e colaboração para mitigar essas consequências. Para operacionalizar essa gestão, especialmente em cidades que enfrentam os desafios mencionados, é fundamental a aplicação de ferramentas específicas.

Para Seiffert (2011), a gestão ambiental se organiza através de ferramentas essenciais que contribuem para sua eficiência e implementação. A política ambiental, por exemplo, serve como o documento base de toda a gestão, definindo os princípios que nortearão as práticas da instituição. Seu objetivo é assegurar que a atividade principal esteja sempre em consonância com o meio ambiente e a legislação vigente. Já o planejamento ambiental dedica-se ao estudo e à antecipação de ações e projetos, sejam eles estruturais ou não, com o intuito de harmonizar as necessidades sociais e governamentais com a proteção dos recursos naturais.

Dentre as diversas ferramentas que aprimoram a gestão ambiental, destacam-se a Avaliação de Impacto Ambiental (*AIA*) e os Sistema de Gestão Ambiental (*SGA*), como a norma ISO 14001, que se tornaram pilares para a análise preventiva de impactos e a padronização de processos dentro de organizações e projetos (Dias, 2017). O licenciamento ambiental, por sua vez, atua como um instrumento regulatório crucial, garantindo que as atividades econômicas estejam em conformidade com as exigências legais e ambientais conforme Seixas e Saccaro Junior. Historicamente, essas ferramentas foram desenvolvidas para fornecer uma estrutura robusta para a tomada de decisões e a implementação de ações. No entanto, a complexidade

crescente dos problemas ambientais urbanos e a necessidade de respostas mais ágeis e baseadas em dados impulsionaram a busca por soluções inovadoras, marcando a entrada da tecnologia no campo da gestão ambiental.

Embora as ferramentas clássicas da gestão ambiental — como a política, o planejamento, a *AIA* e os *SGA* — sejam indispensáveis e estabeleçam a base conceitual e regulatória, sua eficácia plena, especialmente em contextos urbanos dinâmicos e com recursos limitados como Quixadá, pode ser desafiada. A manualidade na coleta de dados, a dificuldade em integrar informações de diferentes fontes e a lentidão na resposta a emergências ambientais representam obstáculos significativos (Barbieri, 2023). O monitoramento e a fiscalização, por exemplo, muitas vezes dependem de visitas presenciais e registros em papel, o que limita a abrangência e a frequência da atuação. Essa lacuna entre a demanda por uma gestão mais ágil e a capacidade de resposta dos métodos convencionais ressalta a necessidade de buscar aprimoramento contínuo.

Nesse cenário, a tecnologia emerge como um vetor transformador, oferecendo soluções para superar essas barreiras. A capacidade de processar e analisar grandes volumes de dados, automatizar tarefas e conectar diferentes atores em tempo real tem revolucionado a maneira como a gestão ambiental é concebida e executada. É nesse ponto que as *TICs* se posicionam como um componente estratégico e indispensável (Silva; Bruno, 2023).

2.2 A Importância das *TICs* na Gestão Ambiental

As *TICs* oferecem um conjunto robusto de ferramentas e abordagens que potencializam a eficácia das políticas e ações da gestão ambiental. Elas promovem a integração de dados, a otimização de processos e, crucialmente, aprimoram a tomada de decisão baseada em evidências. A aplicação das *TICs* na gestão ambiental não se restringe apenas ao monitoramento e à coleta de dados, mas abrange desde o planejamento estratégico até a participação cidadã ativa, conforme será detalhado a seguir.

A transição para uma gestão ambiental mais eficiente e adaptada aos desafios contemporâneos está intrinsecamente ligada à adoção das *TICs*. Em um cenário onde a velocidade das mudanças ambientais e a complexidade dos problemas urbanos exigem respostas ágeis e informadas, as *TICs* oferecem a infraestrutura necessária para coletar, processar e analisar vastas quantidades de dados, algo inviável com métodos tradicionais (Rosa *et al.*, 2023). Essa capacidade de lidar com Big Data e extrair insights relevantes é crucial para identificar padrões de degradação, prever cenários e avaliar o impacto de intervenções.

Além da capacidade analítica, as *TICs* revolucionam a conectividade e a comunicação. Elas permitem que diferentes níveis de governo, instituições de pesquisa, setor privado e a própria sociedade civil se conectem e colaborem de maneira mais eficaz. Conforme Borgo (2007), "infraestrutura de rede de telecomunicações (...) pode-se induzir que tal complexo sócio-tecnológico tornou-se o mecanismo de suporte para observar e investigar o ambiente planetário". Essa interconexão fomenta a transparência e a agilidade na troca de informações, essenciais para a coordenação de esforços em ações complexas de gestão ambiental, como o manejo de resíduos sólidos ou a resposta a emergências. Em cidades de pequeno e médio porte como Quixadá, onde recursos e capacidade institucional podem ser limitados (Cordeiro; Ruela, 2024), a eficiência e a integração proporcionadas pelas *TICs* tornam-se ainda mais valiosas, permitindo otimizar o uso dos escassos recursos e maximizar o impacto das políticas ambientais.

As *TICs*, portanto, não são apenas ferramentas auxiliares; elas se configuram como um componente estratégico fundamental que reestrutura a maneira como a gestão ambiental é concebida e executada, transformando-a em um processo mais inteligente, preditivo e colaborativo. Esse papel central se manifesta em diversas frentes, desde o monitoramento e planejamento técnico até a promoção da participação ativa da população. Como destacam Sampaio *et al.* (2025), “as *TICs* podem ser integradas nas políticas públicas para promover a participação social na gestão do patrimônio cultural, considerando as territorialidades e os direitos humanos”, reforçando a importância dessas tecnologias na construção de modelos mais inclusivos e eficazes de governança ambiental.Esse papel central se manifesta em diversas frentes, desde o monitoramento e planejamento técnico até a promoção da participação ativa da população

2.2.1 TICs como Ferramentas de Monitoramento, Análise e Planejamento

A capacidade transformadora das *TICs* na gestão ambiental se manifesta primeiramente na sua habilidade de otimizar o monitoramento e a análise de dados. Softwares especializados e plataformas digitais permitem a coleta, o armazenamento e o processamento de grandes volumes de informações em tempo real, superando as limitações dos métodos manuais. Conforme Borgo (2007), os *SGA*, por exemplo, são ferramentas essenciais que possibilitam o mapeamento detalhado de recursos naturais, áreas de risco ambiental, focos de poluição e infraestruturas (como aterros sanitários e pontos de coleta de resíduos), oferecendo uma visão espacial integrada para o planejamento e a gestão. Já segundo Servidoni *et al.* (2019), com o avanço dos Sistema de Informações Geográficas (*SIG*), as análises espaciais se tornaram

mais rápidas, menos custosas e mais eficientes, permitindo que a distribuição de informações geográficas seja mais segura e vantajosa, sendo amplamente utilizada por diversas instituições e pelo poder público. Em contextos urbanos, os SIGs auxiliam na identificação de padrões de degradação, na otimização de rotas de coleta e na alocação mais eficiente de recursos para intervenções.

Além dos *SIG*, outras tecnologias digitais, como sensores inteligentes e sistemas de *Internet of Things* (Internet das Coisas) (*IoT*), podem ser empregadas para coletar dados contínuos sobre qualidade da água, do ar, níveis de ruído e volumes de resíduos, fornecendo informações precisas para a tomada de decisão (Parmar *et al.*, 2017). A análise desses dados, muitas vezes utilizando algoritmos de inteligência artificial e machine learning, permite prever tendências, identificar fontes de problemas e avaliar a eficácia das ações implementadas ((Zhong *et al.*, 2021), tornando a gestão mais proativa e menos reativa. Para cidades como Quixadá, onde os recursos são limitados, a eficiência proporcionada por essas tecnologias é fundamental para maximizar o impacto das iniciativas ambientais.

2.2.2 TICs e Participação Cidadã na Gestão Ambiental

Mais do que apenas ferramentas técnicas de monitoramento e análise, as *TICs* assumem um papel vital na reconfiguração da governança ambiental, impulsionando a democratização da gestão e fomentando modelos de governança colaborativa. A disseminação de informações ambientais de forma acessível e transparente, por meio de portais governamentais interativos, bases de dados abertas e plataformas digitais dedicadas, empodera a população. Esse empoderamento se manifesta na capacidade dos cidadãos de não apenas acompanhar o progresso de projetos e políticas públicas, mas também de acessar dados brutos, compreender as decisões tomadas e, consequentemente, participar de forma mais informada e crítica nos processos decisórios (COMITÊ GESTOR DA INTERNET NO BRASIL, 2019). A transparência promovida pelas *TICs* é, portanto, um pilar para a construção de confiança entre o poder público e a sociedade, legitimando as ações governamentais e fortalecendo o controle social sobre a gestão dos recursos naturais e do ambiente urbano (BRASIL. MINISTÉRIO DA CIÊNCIA, TECNOLOGIA e INOVAÇÕES,).

Essa nova dinâmica de interação transcende a consulta passiva de informações. As *TICs* abrem caminho para múltiplos níveis de engajamento cívico, que podem variar desde o consumo de informações e a participação em consultas públicas online, até formas mais ativas

de colaboração, como o crowdsourcing de dados ambientais e a coprodução de soluções para problemas locais. (Ramos, 2019);(CETIC.br, 2022). A capacidade de conectar múltiplos atores – cidadãos, ONGs, pesquisadores, setor privado e poder público – em redes de discussão e ação coordenadas é uma das grandes contribuições das *TICs* para uma gestão ambiental mais policêntrica e adaptativa, conforme apontado por Reis e Werneck (2017)

Nesse contexto, o impacto mais transformador das TICs na participação cidadã reside, possivelmente, no desenvolvimento e popularização de ferramentas digitais participativas, como os Aplicativo (*APP*) e as plataformas cívicas online. Essas soluções tecnológicas convertem o cidadão comum, munido de um smartphone, em um agente ativo e distribuído no território, capaz de monitorar e contribuir para a melhoria ambiental de sua cidade. Aplicativos dedicados permitem que a população realize denúncias de irregularidades ambientais – como descarte irregular de lixo, focos de poluição sonora, vazamentos de água ou esgoto, e ocorrências de queimadas – de forma ágil, intuitiva e, crucialmente, georreferenciada. O envio de fotos, vídeos e descrições detalhadas enriquece a denúncia, fornecendo evidências concretas aos órgãos competentes. Essa funcionalidade não apenas acelera a capacidade de resposta das entidades responsáveis, mas também cria um vasto "exército"de observadores atentos, atuando como sensores em tempo real que complementam a fiscalização oficial, muitas vezes limitada por recursos humanos e materiais (Reis; Werneck, 2017).

Além da fundamental função de canal de denúncias, esses softwares cívicos evoluíram para se tornarem plataformas multifuncionais de engajamento. Eles podem atuar como canais robustos para uma comunicação bidirecional efetiva, facilitando o diálogo contínuo entre cidadãos e gestores públicos, permitindo não só o envio de demandas, mas também o acompanhamento transparente do status de resolução e o feedback sobre as ações tomadas. Adicionalmente, desempenham um papel cada vez mais relevante na educação ambiental, indo além da simples disseminação de informações estáticas. Podem incorporar módulos interativos, gamificação, dicas personalizadas sobre práticas sustentáveis, informações sobre coleta seletiva, calendário de eventos ambientais locais e alertas sobre riscos. Essa abordagem mais dinâmica e engajadora tem maior potencial para promover mudanças de comportamento e aumentar a conscientização ecológica.

Outras funcionalidades relevantes incluem a coleta de feedback e sugestões da população, transformando a plataforma em um espaço para a inteligência coletiva contribuir com ideias inovadoras para melhorias urbanas e ambientais. Incentivam também o mapeamento

colaborativo, onde os cidadãos são encorajados a identificar, registrar e compartilhar informações sobre pontos de interesse ambiental positivo (como nascentes, árvores notáveis, áreas verdes preservadas, hortas comunitárias) ou negativo (áreas degradadas, pontos viciados de descarte de lixo), enriquecendo a base de conhecimento local e fomentando um senso de pertencimento e cuidado com o espaço. (Guilherme, 2022)

Contudo, para que essas plataformas participativas atinjam seu pleno potencial, alguns fatores são cruciais. A usabilidade da ferramenta, a garantia de feedback por parte dos órgãos gestores, a transparência no uso dos dados coletados e a capacidade de integração com os processos internos da administração pública são determinantes para a adesão e o uso contínuo pela população. Igualmente importante é considerar os desafios da inclusão digital, buscando estratégias para que diferentes perfis de cidadãos possam ter acesso e utilizar essas tecnologias, evitando o aprofundamento de desigualdades preexistentes (Ferreira *et al.*, 2022).

Nessa perspectiva, a implementação de softwares cívicos e aplicativos como o *QuixAlert!* em municípios como Quixadá representa um avanço estratégico e promissor. Ao conectar diretamente a população com a gestão ambiental local, por meio de uma interface acessível e focada nas necessidades da comunidade, promove-se não apenas uma maior eficiência na identificação e resolução de problemas ambientais cotidianos, mas também o fortalecimento da consciência ambiental coletiva. Fomenta-se, assim, a construção de uma governança mais participativa, dialógica e responsiva, onde a responsabilidade pela sustentabilidade urbana e a qualidade de vida é efetivamente compartilhada por todos os atores sociais (Souza; Almeida, 2024).

2.2.3 Exemplos de Aplicações de TICs em Gestão Ambiental no Brasil

A aplicação das TICs na gestão ambiental brasileira tem se manifestado em diversas iniciativas, abrangendo desde sistemas robustos de monitoramento em larga escala, operados por instituições governamentais, até aplicativos focados na participação cidadã e na ciência cidadã, desenvolvidos por universidades, Organizações Não Governamentais (*ONGS*) e startups. Esses exemplos práticos ilustram o potencial transformador das TICs na coleta de dados, na fiscalização, no engajamento da população e na tomada de decisões mais ágeis e informadas.

No âmbito governamental, o Brasil possui exemplos notórios de uso de *TICs* para monitoramento e controle ambiental. O IInstituto Nacional de Pesquisas Espaciais (*INPE*) é pioneiro com sistemas como o Projeto de Monitoramento do Desmatamento na Amazônia

Legal por Satélite (*PRODES*) e o Detecção de Desmatamento em Tempo Real (*DETER*). Essas plataformas utilizam imagens de satélite e geoprocessamento para monitorar a cobertura vegetal, gerar alertas de desmatamento e subsidiar ações de fiscalização e políticas de conservação em biomas críticos como a Amazônia e o Cerrado (INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS, 2025). Outro exemplo significativo é o Sistema Nacional de Controle da Origem dos Produtos Florestais (*SINAFLOR*), coordenado pelo Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (*IBAMA*), que integra o controle da origem da madeira, do carvão e de outros produtos e subprodutos florestais, desde a autorização de exploração até o transporte e armazenamento, visando combater o desmatamento ilegal e promover a sustentabilidade no setor florestal (INSTITUTO BRASILEIRO DO MEIO AMBIENTE E DOS RECURSOS NATURAIS RENOVÁVEIS, 2025). Centro Nacional de Monitoramento e Alertas de Desastres Naturais (*CEMADEN*) utiliza intensivamente *TICs* para monitorar condições geo-hidrometeorológicas e emitir alertas antecipados de desastres como inundações e deslizamentos de terra, essenciais para a gestão de riscos (CENTRO NACIONAL DE MONITORAMENTO E ALERTAS DE DESASTRES NATURAIS, 2025).

No campo da participação cidadã e da ciência cidadã, diversas aplicações têm surgido para engajar a população no cuidado com o meio ambiente. Plataformas como o Colab.re, embora de propósito mais amplo para a zeladoria urbana, são frequentemente utilizadas por cidadãos para reportar problemas ambientais, como descarte irregular de resíduos, terrenos baldios necessitando de limpeza, ou falhas na coleta seletiva, direcionando essas demandas às prefeituras (Colab, 2025). Iniciativas de ciência cidadã também se destacam, como o aplicativo Urubu Mobile, vinculado à Rede Brasileira de Monitoramento de Atropelamentos de Fauna Silvestre (Sistema Urubu), que permite a qualquer cidadão registrar atropelamentos de animais em rodovias, gerando dados valiosos para a pesquisa e para a proposição de medidas mitigadoras (SISTEMA DE INFORMAÇÕES SOBRE A BIODIVERSIDADE BRASILEIRA, 2025). Projetos desenvolvidos por universidades e ONGs também frequentemente resultam em aplicativos para monitoramento participativo da qualidade da água, da biodiversidade local, ou para denúncias específicas de crimes ambientais em unidades de conservação (UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO, 2025).

Muitas secretarias estaduais e municipais de meio ambiente também têm desenvolvido seus próprios canais digitais, incluindo aplicativos ou sistemas online para denúncias ambientais, consulta de processos de licenciamento ou acesso a informações sobre a qualidade

ambiental. Essas ferramentas buscam aproximar a gestão ambiental do cidadão, tornando os serviços mais acessíveis e a comunicação mais fluida (SUPERINTENDÊNCIA ESTADUAL DO MEIO AMBIENTE (CE), 2019).

A análise desses exemplos evidencia uma tendência crescente de incorporação das *TICs* na gestão ambiental brasileira, tanto por iniciativas públicas quanto por projetos da sociedade civil. Essas ferramentas têm se mostrado fundamentais para ampliar a capacidade de monitoramento, otimizar os processos de fiscalização, promover a transparência e, sobretudo, envolver a população como corresponsável pela proteção do meio ambiente. Nesse cenário de inovação e busca por soluções mais eficazes e participativas, o ***QuixAlert!*** se apresenta como uma proposta alinhada às demandas específicas de municípios de pequeno e médio porte, como Quixadá. Para posicionar a aplicação em relação a outras iniciativas similares, a Tabela 1 a seguir sintetiza comparativamente os principais sistemas voltados à gestão urbana e ambiental, destacando escopo, funcionalidades, participação cidadã e diferenciais técnicos.

Tabela 1 – Comparativo entre o *QuixAlert!* e outras ferramentas digitais para gestão urbana e ambiental.

Sistema	Funcionalidades Principais	Diferenciais Técnicos
QuixAlert! Gestão ambiental local com foco em cidades pequenas	Denúncias ambientais, adoção de animais, notificações, solicitação de documentos, notícias	Arquitetura modular, personalizável e escalável
Colab.re Zeladoria urbana e comunicação com prefeituras	Denúncias de problemas urbanos (buracos, lixo, iluminação)	Integração com prefeituras, uso simplificado
Urubu Mobile Monitoramento de atropelamento de fauna silvestre	Registro de atropelamentos com foto e geolocalização	Base de dados colaborativa e foco em conservação da fauna
CEMADEN Monitoramento e alertas de desastres naturais	Monitoramento pluviométrico e de deslizamentos, alertas em tempo real	Uso de sensores geotécnicos e modelagem meteorológica
Plataforma Verde SP Gestão ambiental em centros urbanos grandes	Consulta de licenças ambientais, denúncias, mapa de poluição	Integração com dados públicos e mapeamento geoespacial

Fonte: Elaborado pelo autor (2025).

A Tabela 1 posiciona o *QuixAlert!* em relação a outras iniciativas, ilustrando seu escopo e funcionalidades no panorama das *TICs* para gestão ambiental no Brasil. A existência e a diversidade dessas ferramentas ressaltam a importância da tecnologia nesse domínio. No entanto, para que um sistema de software como o *QuixAlert!*, desenvolvido para atender a um contexto dinâmico e com potencial de evolução e replicação, possa efetivamente cumprir seus objetivos a longo prazo, não basta apenas que ele apresente as funcionalidades desejadas externamente. É crucial que sua estrutura interna, a qualidade do seu código-fonte e sua arquitetura sejam robustas, flexíveis e passíveis de manutenção eficiente. Questões como a escalabilidade da solução e a facilidade de introdução de novas funcionalidades ou adaptações futuras dependem diretamente desses aspectos técnicos internos.

Nesse sentido, práticas consolidadas de Engenharia de Software, como a refatoração e a modularização, são indispensáveis para garantir a saúde técnica e a longevidade de projetos de software, incluindo o *QuixAlert!*. Este capítulo, que até aqui explorou o domínio da aplicação e o papel das TICs na gestão ambiental, passa agora a detalhar os fundamentos teóricos dessas práticas de engenharia. Elas são centrais para alcançar os objetivos de melhoria da qualidade interna, escalabilidade e adaptabilidade propostos para o *QuixAlert!* neste trabalho. A primeira dessas práticas a ser aprofundada é a refatoração de software.

2.3 Refatoração do *QuixAlert!*

A refatoração de software é uma técnica disciplinada que visa reestruturar um código existente, alterando sua organização interna sem modificar seu comportamento externo visível. Conforme definido por Tavares et al., trata-se de um processo de melhoria contínua do design do código após ele ter sido escrito, com o objetivo principal de torná-lo mais compreensível, fácil de manter e, consequentemente, menos propenso a erros. Fowler (2018), uma referência na área, complementa que a refatoração é essencial para manter a qualidade do design do software ao longo do tempo, especialmente em projetos que sofrem alterações e acréscimos constantes, ajudando a controlar a complexidade e a facilitar a evolução do sistema.

O principal motivador para a refatoração é a constatação de que o código, embora funcional, pode se deteriorar com o passar do tempo e com as sucessivas modificações, acumulando o que se convencionou chamar de "dívida técnica". Esta dívida, geralmente, representa o custo implícito de retrabalho causado por se optar por uma solução fácil e rápida no presente, em vez de usar uma abordagem de melhor qualidade que levaria mais tempo (Melo *et al.*, 2020).

Se não gerenciada, a dívida técnica pode comprometer severamente a capacidade de evolução do software, tornando cada nova alteração mais custosa, arriscada e demorada (Rios *et al.*, 2021).

2.3.1 Benefícios da Refatoração

Os benefícios de incorporar a refatoração como uma prática contínua no desenvolvimento de software são múltiplos e impactam diretamente a qualidade do produto final, a produtividade da equipe e a saúde do projeto a longo prazo. Fowler (2018, p. 3-6)() elenca diversos desses proveitos, que vão além da simples estética do código, permeando aspectos cruciais da engenharia de software:

Primeiramente, a refatoração melhora o design do software de forma substancial. Com o tempo, e sob a pressão de cronogramas apertados ou requisitos que mudam rapidamente, é comum que o design inicial de um sistema se degrade, resultando em código acoplado, pouco coeso e difícil de modificar (Santos *et al.*, 2016). A refatoração permite que os desenvolvedores revisitem e aprimorem continuamente a arquitetura e a organização do código, eliminando duplicações, simplificando estruturas condicionais complexas, aumentando a coesão dentro das classes e módulos, e reduzindo o acoplamento entre eles. Um design mais limpo, flexível e robusto não é apenas um ideal técnico, mas um facilitador direto da capacidade do software de se adaptar a novas necessidades de negócio e evoluir com menor atrito. Isso se traduz em menor custo de propriedade a longo prazo, pois as alterações futuras demandam menos esforço e apresentam menor risco (Fowler, 2019).

Consequentemente, um software com design aprimorado torna-se mais fácil de entender. Código claro, com nomes significativos para variáveis, métodos e classes, métodos curtos com responsabilidades únicas e classes bem definidas, é intrinsecamente mais legível e autoexplicativo. Essa clareza é vital não apenas para facilitar a integração de novos membros à equipe de desenvolvimento, reduzindo sua curva de aprendizado, mas também para o desenvolvedor original que retorna ao código após um período, ou para qualquer colega que precise interagir com aquele trecho do sistema (Alawad *et al.*, 2019). A redução da carga cognitiva necessária para compreender e trabalhar no sistema minimiza a probabilidade de erros de interpretação e facilita a colaboração eficaz entre os membros da equipe (Fowler, 2018)(p. 4). Um código compreensível é um pré-requisito para uma manutenção eficiente e para a transferência de conhecimento dentro da equipe.

Outro benefício significativo é que o próprio processo de refatoração ajuda a encon-

trar bugs. Ao analisar criticamente o código existente com o objetivo de identificar pontos de melhoria estrutural, os desenvolvedores frequentemente descobrem defeitos latentes, condições de borda não tratadas ou comportamentos inesperados que poderiam passar despercebidos em uma revisão superficial ou durante os testes funcionais (Ferreira *et al.*, 2025). A simplificação da lógica e a clarificação da estrutura do código podem tornar as causas raízes dos problemas muito mais evidentes, permitindo sua correção antes que impactem os usuários finais. Portanto, a refatoração atua também como uma forma de depuração proativa.

Paralelamente, embora a refatoração consuma um certo tempo e esforço no curto prazo, ela ajuda a programar mais rápido no futuro (Fowler, 2018, p. 5) Este é talvez um dos benefícios mais importantes do ponto de vista da produtividade. Um código de alta qualidade, bem estruturado, modularizado e fácil de entender permite que novas funcionalidades sejam implementadas com maior agilidade, confiança e segurança. A equipe gasta menos tempo decifrando complexidades desnecessárias, lutando contra efeitos colaterais inesperados de suas alterações, ou corrigindo bugs introduzidos por modificações em código frágil e emaranhado. Com uma base de código saudável, os desenvolvedores podem se concentrar em agregar valor ao produto, em vez de constantemente combater a entropia do software.

Por outro ponto de vista, como argumenta Sommerville (2020), a refatoração é uma forma essencial de manutenção preventiva. Ao invés de adotar uma postura reativa, esperando que os problemas de design se agravem a ponto de exigir grandes e custosas reescritas (que muitas vezes falham ou extrapolam prazos e orçamentos), a refatoração contínua e incremental atua como uma profilaxia. Ela previne a erosão gradual da qualidade do código, da arquitetura e da moral da equipe, garantindo que o sistema permaneça saudável, resiliente e evoluível ao longo de todo o seu ciclo de vida. Esse cuidado constante com a estrutura interna do software é um investimento estratégico que se paga ao reduzir os custos de manutenção corretiva e adaptativa, e ao aumentar significativamente a longevidade e o valor do ativo de software (Sommerville, 2020).

2.3.2 Identificando Code Smells

A decisão de quando e onde aplicar técnicas de refatoração é frequentemente guiada pela capacidade de identificar os chamados *"maus cheiros no código"* (*code smells*). Este termo, popularizado por Kent Beck e amplamente discutido por Fowler (2018)(p. 59), refere-se a certos padrões ou características no código que, embora não sejam necessariamente *bugs* que impeçam

a execução do programa, são fortes indicadores de problemas de design subjacentes. Esses "sintomas" sugerem que o código pode ser difícil de entender, modificar, testar ou manter e que, se não tratados, tendem a piorar com o tempo, levando a uma maior complexidade e propensão a erros. Reconhecer esses *"cheiros"* é uma habilidade crucial para o desenvolvedor que busca manter a qualidade e a saúde do código. Fowler (2018, cap. 3) apresenta um catálogo extenso e detalhado, e alguns dos mais comuns e impactantes incluem:

Código Duplicado (*Duplicated Code*)

Este é um dos *smells* mais prevalentes e problemáticos. Ocorre quando blocos idênticos ou muito similares de código são encontrados em diferentes partes do sistema, seja dentro da mesma classe, entre classes diferentes ou até mesmo em módulos distintos. Fowler (2018)(p. 63) o considera um dos piores, pois viola frontalmente o princípio *Don't Repeat Yourself* (Não se Repita) (*DRY*) (*Don't Repeat Yourself – Não se Repita*).

A principal consequência da duplicação é o aumento exponencial do esforço de manutenção: qualquer alteração lógica ou correção de *bug* em uma instância do código duplicado precisa ser manualmente replicada em todas as outras. Este processo é tedioso, altamente propenso a falhas humanas (esquecer uma das cópias, por exemplo) e pode levar a inconsistências e comportamentos divergentes no software. Além disso, o código duplicado infla desnecessariamente a base de código, tornando-a mais difícil de navegar, compreender e testar.

Refatorações como *"Extrair Método"*, *"Extrair Superclasse"*, *"Template Method"* ou a introdução de classes utilitárias são frequentemente usadas para centralizar a lógica repetida e eliminar a duplicação.

Método Longo (*Long Method*)

Métodos ou funções que crescem excessivamente, acumulando muitas linhas de código e, frequentemente, múltiplas responsabilidades, tornam-se um pesadelo para a manutenção. Quanto mais longo um método, mais difícil é para o leitor compreender seu propósito, seu fluxo de controle, suas variáveis locais e seus efeitos colaterais (Fowler, 2018, p. 65).

Métodos longos são difíceis de testar de forma isolada e completa, e qualquer alteração neles carrega um risco maior de introduzir novos problemas. A principal refatoração para lidar com este *smell* é a *"Extrair Método"*, que consiste em decompor o método longo em

vários métodos menores, mais coesos, com nomes autoexplicativos e responsabilidades únicas. A regra geral é que um método deve fazer apenas uma coisa e fazê-la bem.

Classe Grande (*Large Class / God Class*)

Similar ao método longo, uma classe grande, também conhecida como *"Classe Deus"* (*God Class*), é aquela que tenta fazer ou conhecer demais. Ela centraliza um número excessivo de responsabilidades, atributos e métodos, muitas vezes interagindo com uma grande quantidade de outras classes.

Tais classes são complexas de entender e manter, tendem a ter baixo nível de coesão (seus membros não estão fortemente relacionados em prol de um propósito único) e alto acoplamento com outras partes do sistema. Elas violam o *Single Responsibility Principle* (Princípio da Responsabilidade Única) (*SRP*), que preconiza que uma classe deve ter apenas um motivo para mudar.

Refatorações como *"Extrair Classe"*, *"Extrair Subclasse"* ou *"Extrair Interface"* podem ajudar a distribuir as responsabilidades de forma mais equilibrada, criando classes menores, mais focadas e mais fáceis de gerenciar.

Lista Longa de Parâmetros (*Long Parameter List*)

Métodos que requerem uma longa lista de parâmetros para serem chamados podem ser um sinal de que o método está tentando fazer muitas coisas diferentes, ou que alguns desses parâmetros poderiam ser agrupados em um objeto (através da refatoração *"Introduzir Objeto de Parâmetro"* ou *"Preservar Objeto Inteiro"*), ou ainda que o método talvez esteja na classe errada e precise de tantos dados de fora porque não tem acesso direto a eles.

Listas longas de parâmetros dificultam a leitura e o uso do método, tornam a assinatura do método mais instável a mudanças e podem levar a erros na ordem ou no tipo dos argumentos passados (Fowler, 2018, p. 73)

Comentários Excessivos ou Desnecessários (*Comments*)

Embora comentários sejam uma ferramenta importante para explicar decisões de design complexas, algoritmos não triviais ou partes do código que têm um comportamento sutil e não óbvio, um excesso deles, especialmente aqueles que simplesmente parafraseiam o que o

código já diz claramente (*"comentários cosméticos"*) ou que são usados como uma *muleta* para justificar um código confuso e mal escrito, pode ser um *smell* (Fowler, 2018, p. 87)

O ideal é que o código seja o mais autoexplicativo possível, através de bons nomes para variáveis, métodos e classes, e de estruturas claras. Comentários que se tornam obsoletos porque o código foi alterado mas eles não foram atualizados podem enganar o leitor e ser mais prejudiciais do que a ausência de comentários.

A refatoração aqui pode envolver tornar o código mais claro para que o comentário se torne desnecessário, ou garantir que os comentários existentes sejam precisos e valiosos.

Outros *Code Smells* Relevantes

Outros *code smells* frequentemente citados incluem:

- ***Feature Envy***: Quando um método de uma classe parece mais interessado nos dados e métodos de outra classe do que nos da sua própria.
- ***Data Clumps***: Grupos de variáveis que frequentemente aparecem juntas em diferentes partes do código e poderiam ser encapsuladas em um objeto.
- ***Primitive Obsession***: Uso excessivo de tipos de dados primitivos (como inteiros, *strings*, booleanos) para representar conceitos do domínio que poderiam ser mais bem modelados por pequenos objetos.
- ***Switch Statements***: Especialmente quando baseadas no tipo de um objeto, podendo indicar uma oportunidade para usar polimorfismo.
- ***Message Chains***: Longas sequências de chamadas de métodos (ex: `objeto.getOutraCoisa().getMa` que aumentam o acoplamento.

A sensibilidade para identificar esses e outros *"maus cheiros"* é desenvolvida com a experiência e o estudo, e é fundamental para a prática eficaz da refatoração.

2.3.3 Técnicas de Refatoração

Uma vez que um "mau cheiro"é identificado, o desenvolvedor dispõe de um vasto e bem documentado catálogo de técnicas de refatoração. Essas técnicas são, em essência, transformações estruturais no código, geralmente pequenas e mecânicas, que visam melhorar sua organização interna sem alterar seu comportamento externo observável. A disciplina na aplicação dessas técnicas, seguindo os passos recomendados, é crucial para minimizar o risco de

introduzir erros. Fowler (2018) em sua obra *Refactoring* apresenta um extenso compêndio dessas "receitas"de refatoração, cada uma com suas motivações, uma descrição detalhada da mecânica de aplicação (passos) e exemplos práticos. Algumas das mais fundamentais e frequentemente utilizadas no dia a dia do desenvolvimento incluem:

Extrair Método (Extract Method): Esta é, possivelmente, uma das refatorações mais comuns e poderosas. Consiste em identificar um fragmento de código dentro de um método maior que possui uma lógica coesa e pode ser agrupado. Esse fragmento é então movido para um novo método independente, que recebe um nome autoexplicativo que descreva claramente seu propósito. O código original no método de origem é substituído por uma chamada ao novo método recém-criado. Os principais benefícios são a melhoria drástica da legibilidade do método original (que se torna menor e mais fácil de entender), a redução da complexidade, a eliminação de duplicação (se o mesmo fragmento aparecer em outros lugares) e a promoção da reutilização do código encapsulado no novo método (Fowler, 2018, p. 106). A mecânica envolve identificar variáveis locais usadas no fragmento, verificar se são modificadas, e passá-las como parâmetros ao novo método ou retornar valores modificados, se necessário.

Mover Método (Move Method) e Mover Campo (Move Field): Estas técnicas são aplicadas para melhorar a distribuição de responsabilidades entre as classes, aumentando a coesão e reduzindo o acoplamento. "Mover Método"é usado quando um método em uma classe parece interagir mais intensamente com os dados ou métodos de outra classe do que com os da sua própria (um sintoma de Feature Envy). O método é então movido para essa outra classe mais apropriada. "Mover Campo"é similar, mas aplicado a atributos (campos) que fazem mais sentido semanticamente ou são mais utilizados por outra classe. Ao mover esses membros, a classe de origem torna-se mais enxuta e focada, e a classe de destino ganha uma responsabilidade que lhe é mais pertinente (Fowler, 2018, p. 136, p. 140).

Renomear (Rename Variable, Method, Class, etc.): A clareza e a expressividade dos nomes dos elementos do código são absolutamente fundamentais para a sua compreensão e manutenibilidade. Esta refatoração, embora pareça trivial, tem um impacto profundo. Consiste em escolher nomes mais significativos, precisos, consistentes e que revelem a intenção do elemento. Nomes bem escolhidos podem eliminar a necessidade de comentários explicativos e tornar a lógica do programa muito mais óbvia para quem lê o código (Fowler, 2018, p. 119).

Ferramentas de desenvolvimento modernas (IDEs) geralmente oferecem suporte automatizado para renomeações seguras, atualizando todas as referências ao elemento renomeado.

Substituir Número Mágico por Constante Simbólica (Replace Magic Number with Symbolic Constant): "Números mágicos"são valores literais numéricos (ou strings) que aparecem diretamente no código sem uma explicação clara do seu significado ou propósito (ex: `if (status == 2)`). Substituí-los por constantes nomeadas (ex: `private static final int STATUSPROCESSADO = 2; if (status == STATUSPROCESSADO)`) melhora drasticamente a legibilidade, pois o nome da constante documenta o significado do valor. Além disso, facilita futuras alterações, pois o valor só precisaria ser modificado na declaração da constante, em vez de em múltiplos locais no código onde o número mágico poderia estar espalhado (Fowler, 2018, p. 166).

Extrair Classe (Extract Class): Esta refatoração é aplicada quando se percebe que uma única classe está acumulando muitas responsabilidades distintas ou gerenciando conjuntos de dados e comportamentos que poderiam ser agrupados em um novo conceito coeso. Consiste em criar uma nova classe e mover os campos e métodos relevantes da classe original (que está "grande demais"ou pouco coesa) para esta nova classe. Uma ligação (associação ou agregação) é então estabelecida entre a classe original e a nova. Isso promove melhor coesão em ambas as classes, ajuda a seguir o Princípio da Responsabilidade Única (SRP) e pode simplificar significativamente a compreensão e a manutenção de cada uma delas (Fowler, 2018, p. 144).

Encapsular Campo (Encapsulate Field): Envolve tornar um campo (atributo) público de uma classe em privado e fornecer métodos de acesso públicos (geralmente getters para leitura e, se necessário, setters para modificação) para ele. Esta é uma prática fundamental do encapsulamento em programação orientada a objetos. Ao fazer isso, a classe ganha mais controle sobre seus dados internos e como eles são acessados ou modificados, podendo adicionar lógica de validação ou outros comportamentos nos métodos de acesso. Além disso, facilita futuras alterações na representação interna do campo (ex: mudar o tipo de dado ou a forma como é calculado) sem impactar os clientes da classe, desde que a interface dos métodos de acesso seja mantida (Fowler, 2018, p. 162).

Introduzir Variável Explicativa (Introduce Explaining Variable): Em expressões complexas ou longas, pode ser útil quebrar a expressão em partes menores, atribuindo cada parte a uma nova variável local com um nome que explique seu significado. A expressão original é

então reescrita usando essas variáveis. Isso torna a lógica da expressão muito mais fácil de seguir e entender, funcionando como uma forma de documentação interna (Fowler, 2018, p. 115).

A aplicação bem-sucedida e segura dessas e de outras inúmeras técnicas de refatoração (como "Substituir Condicional por Polimorfismo", "Extrair Superclasse", "Pull Up/Push Down Method/Field", etc.) exige não apenas o conhecimento da mecânica de cada uma, mas também a disciplina de aplicá-las de forma incremental. Como enfatizado repetidamente por Fowler (2018), a confiança para realizar essas transformações é imensamente aumentada pela existência de uma suíte de testes automatizados. Os testes atuam como uma rede de segurança, garantindo que, após cada pequena transformação, o comportamento observável do sistema permaneça inalterado, permitindo que refatorações maiores e mais complexas sejam construídas a partir de uma série de passos pequenos, controlados e seguros.

2.3.4 Refatoração e o Projeto QuixAlert!

Os conceitos de refatoração de software apresentados nesta seção são diretamente aplicáveis e fundamentais para os objetivos do projeto ***QuixAlert!***. Conforme detalhado na introdução deste trabalho, a aplicação *QuixAlert!*, após seu ciclo inicial de desenvolvimento e uso, acumulou pontos que indicavam a necessidade de melhorias internas para garantir sua manutenibilidade, legibilidade e, crucialmente, para facilitar tanto a introdução de novas funcionalidades quanto o processo de modularização visando a escalabilidade.

O diagnóstico técnico inicial do *QuixAlert!* revelou a presença massiva dos *code smells* mais comuns, como o Método Longo e o Código Duplicado, que são precisamente os problemas combatidos pelas técnicas de Extrair Método e Extrair Classe (Seção 2.3.3). A refatoração foi, portanto, a ação de manutenção preventiva (Sommerville, 2020) essencial para reduzir a dívida técnica acumulada (Melo *et al.*, 2020; Rios *et al.*, 2021), preparando o sistema para a próxima fase arquitetural. O foco em clean code e no SRP (*Single Responsibility Principle*) garantiu que a base de código estivesse apta a suportar o modelo modular e o desenvolvimento de novas features complexas como o Chat em Tempo Real, minimizando o risco de regressões.

As decisões estratégicas, o processo metodológico e os resultados específicos da refatoração aplicada ao *QuixAlert!* serão detalhados e discutidos no Capítulo 4. A presente fundamentação teórica, portanto, serve de alicerce para justificar, orientar e avaliar as atividades de engenharia de software realizadas.

2.4 Modularização

A modularização é um princípio fundamental da engenharia de software que consiste na decomposição de um sistema em componentes menores, independentes e intercambiáveis, conhecidos como módulos. O objetivo é organizar o software de forma que cada módulo encapsule uma responsabilidade ou funcionalidade específica, interagindo com os demais através de interfaces bem definidas e estáveis. Esta abordagem é um pilar para a construção de sistemas complexos, manuteníveis e escaláveis.

A migração para a modularização foi a resposta arquitetural direta ao alto acoplamento e à baixa coesão observados na primeira versão monolítica do *QuixAlert!*. Ao decompor o software em módulos funcionais (como o futuro `:feature_chat` e `:feature_adocao`), a nova arquitetura aplica o princípio de baixo acoplamento (McConnell, 2004) para resolver o problema de impacto cruzado entre as funcionalidades, garantindo que a evolução do módulo de denúncias não afete a estabilidade dos serviços de adoção.

2.4.1 Princípios Essenciais: Coesão e Acoplamento

O sucesso de uma arquitetura modular é governado por dois princípios de design inter-relacionados: **alta coesão** e **baixo acoplamento**.

Alta Coesão: Refere-se ao grau em que os elementos dentro de um mesmo módulo pertencem uns aos outros e trabalham juntos para um propósito único e bem definido. Um módulo coeso agrupa responsabilidades que são logicamente relacionadas. Por exemplo, em um módulo de *autenticação*, todas as classes, funções e telas devem estar relacionadas exclusivamente ao processo de login, cadastro e gerenciamento de sessão do usuário. A alta coesão torna os módulos mais fáceis de entender e manter (Sommerville, 2020).

Baixo Acoplamento: Mede o grau de interdependência entre os módulos. Em uma arquitetura ideal, os módulos devem saber o mínimo possível uns sobre os outros. As interações devem ocorrer através de interfaces públicas e estáveis, e não por meio do acesso direto aos detalhes de implementação de outros módulos. O baixo acoplamento é crucial, pois permite que um módulo seja modificado ou substituído com impacto mínimo ou nulo sobre os outros componentes do sistema, aumentando a resiliência a mudanças (McConnell, 2004).

2.4.2 Benefícios de uma Arquitetura Modular

A adoção de uma arquitetura modular traz uma série de benefícios que impactam todo o ciclo de vida do software, desde o desenvolvimento até a manutenção e evolução:

- **Manutenibilidade e Compreensibilidade:** Módulos menores e focados são mais fáceis de entender, depurar e manter. As alterações ficam contidas dentro das fronteiras de um módulo, reduzindo o risco de introduzir efeitos colaterais inesperados. Este princípio, conhecido como ocultação da informação, é fundamental para a engenharia de software robusta (Parnas, 1972; Martin, 2017).
- **Desenvolvimento Paralelo:** A decomposição do sistema em módulos independentes permite que diferentes desenvolvedores ou equipes trabalhem em paralelo em funcionalidades distintas, com menor chance de conflitos de código e sobrecarga de comunicação (Brooks, 1995).
- **Reutilização de Código:** Funcionalidades ou componentes comuns a várias partes da aplicação (como serviços de rede, componentes de UI ou lógica de acesso a dados) podem ser encapsulados em módulos e reutilizados, evitando duplicação de código. Padrões de projeto são um exemplo de soluções modulares reutilizáveis (Gamma *et al.*, 1994).
- **Testabilidade:** A modularidade facilita a escrita e execução de testes. Cada módulo pode ser testado isoladamente (testes unitários), garantindo seu correto funcionamento antes de ser integrado ao sistema completo, sendo um pilar para o desenvolvimento de código limpo e a modernização de sistemas legados (Martin, 2008; Feathers, 2004).

2.4.3 Modularização no Contexto de Aplicações Android

No desenvolvimento de aplicações nativas para Android com Kotlin, a modularização é uma prática recomendada e suportada pelo próprio ambiente de desenvolvimento. Ela é tipicamente implementada através da criação de múltiplos *Android Library Modules* dentro de um mesmo projeto no Android Studio. Essa abordagem permite uma separação física e lógica do código, seguindo a seguinte estrutura:

- **Módulo de Aplicação (`:app`):** Módulo principal que serve como ponto de entrada da aplicação. Integra os demais módulos e geralmente contém pouca ou nenhuma lógica de negócio.
- **Módulos de Funcionalidade (`:feature_*`):** Cada um encapsula uma funcionalidade

completa e vertical do aplicativo, como `:feature_denuncias` ou `:feature_adocao`. Idealmente, um módulo de funcionalidade não deve depender diretamente de outro.

– **Módulos de Biblioteca (**`:core`**,** `:data`**, etc.):** Contêm código compartilhado e reutilizável, como modelos de dados, clientes de rede, acesso a banco de dados e componentes visuais comuns.

Além de organizar o projeto, essa estrutura pode levar a melhorias no tempo de compilação, pois o sistema de *build* do Gradle pode recompilar apenas os módulos modificados.

2.4.4 Modularização como Estratégia de Negócio e Escalabilidade

A modularização transcende os benefícios puramente técnicos e se torna uma poderosa habilitadora da estratégia de negócio e produto. A independência dos módulos de funcionalidade permite que a aplicação seja montada e distribuída em diferentes configurações.

Essa capacidade de "ligar"ou "desligar"módulos conforme a necessidade é o que permite a criação de um produto flexível e comercialmente escalável. É possível, por exemplo, oferecer uma versão básica do *QuixAlert!* contendo apenas o módulo de *Denúncias* e, posteriormente, oferecer os módulos de *Adoção de Animais* ou *Educação Ambiental* como pacotes adicionais.

Conforme destacado por Bass *et al.* (2021) (2021), arquiteturas modulares promovem a flexibilidade necessária para adaptação contínua. Essa estratégia não apenas atende a diferentes necessidades e orçamentos de potenciais clientes, mas também define um caminho claro para a evolução e sustentabilidade do produto a longo prazo, transformando o *QuixAlert!* em uma plataforma customizável — e não apenas em um aplicativo monolítico.

3 TRABALHOS RELACIONADOS

Nesta seção, serão apresentados trabalhos relacionados que abordam temáticas alinhadas ao escopo deste projeto. Para melhor organização e aprofundamento, a análise será dividida em três subseções: a primeira dedicada a softwares cívicos voltados à gestão ambiental urbana e à participação cidadã; a segunda abordando pesquisas sobre refatoração de software; e a terceira tratando da modularização e escalabilidade de sistemas. Essa estrutura visa facilitar a compreensão das contribuições existentes em cada área e destacar como o *QuixAlert!* se insere nesse contexto.

3.1 Softwares Cívicos para Gestão e Participação Ambiental

A literatura acadêmica nacional apresenta diversas propostas de softwares cívicos que utilizam a tecnologia para promover a participação cidadã em questões urbanas e ambientais. A análise dessas iniciativas é fundamental para compreender o estado da arte, identificar padrões de funcionalidades e, principalmente, contextualizar a proposta do *QuixAlert!*. Nesta seção, são analisados três trabalhos que, embora com focos distintos, compartilham o objetivo de engajar cidadãos na coleta de informações e na comunicação com o poder público.

O trabalho de Dourado *et al.* (2018) apresenta o aplicativo "Meu Bairro", uma ferramenta de apoio à zeladoria urbana colaborativa. O principal objetivo do sistema é funcionar como um canal de comunicação direto entre os cidadãos e os órgãos públicos municipais. Através do aplicativo, os usuários podem reportar uma gama variada de problemas urbanos, como buracos em vias, acúmulo de lixo, problemas na iluminação pública e poda de árvores. O sistema permite o envio de fotos e a localização do problema, encaminhando a demanda para a secretaria responsável. A proposta do "Meu Bairro"se assemelha à do *QuixAlert!* em seu princípio fundamental de ser uma ponte digital para a comunicação de ocorrências entre a população e a gestão municipal. No entanto, o "Meu Bairro"possui um escopo de zeladoria urbana genérico e amplo. O QuixAlert!, por outro lado, se diferencia por ter um foco temático estritamente ambiental, alinhado às competências de uma autarquia específica (a *AMMA*, em Quixadá). Além disso, a proposta do *QuixAlert!* transcende o modelo de simples relato de problemas, ao incorporar módulos de serviço como a adoção de animais, notícias e solicitações de documentos, caracterizando-se como uma plataforma de gestão e interação ambiental mais completa e especializada para as necessidades de um município.

Com um foco ambiental mais específico, Alencar *et al.* (2020) desenvolveram o "Gota.Urb", um aplicativo para o mapeamento colaborativo de alagamentos urbanos. A ferramenta permite que os cidadãos registrem pontos de alagamento em um mapa, informando dados como o nível da água, a duração do evento e o envio de fotos. O objetivo é criar uma base de dados pública e histórica sobre áreas de risco, que possa subsidiar o planejamento urbano por parte do poder público e auxiliar a população a evitar locais vulneráveis. A similaridade com o *QuixAlert!* reside no uso da colaboração cidadã para gerar dados georreferenciados sobre um problema ambiental. Contudo, a principal diferença está na especialização e no propósito da ferramenta. O "Gota.Urb"é uma plataforma de mapeamento de um único problema (alagamentos) e seu resultado primário é a geração de um mapa de dados. O *QuixAlert!*, em contraste, foi concebido para ser uma ferramenta de gestão multi-problema, capaz de lidar com diversas categorias de ocorrências ambientais (descarte de resíduos, queimadas, poluição sonora, etc.). Mais importante, enquanto o "Gota.Urb"foca na criação de uma base de dados para consulta, o *QuixAlert!* é projetado para ser um canal de ação direta, onde cada denúncia gera um protocolo e inicia um fluxo de atendimento dentro do órgão ambiental responsável, com foco na resolução da ocorrência.

Uma terceira abordagem, também altamente especializada, é apresentada por Queiroz *et al.* (2023) com o aplicativo "Guardiões da Chapada". Este sistema foi desenvolvido como uma ferramenta colaborativa para o combate a incêndios na Chapada Diamantina, um problema ambiental específico de uma localidade particular. A plataforma permite que usuários, como brigadistas, voluntários e gestores do ICMBio, reportem focos de incêndio, visualizem as ocorrências em um mapa e monitorem seu status, centralizando a comunicação e agilizando a resposta a emergências. Assim como o *QuixAlert!*, o "Guardiões da Chapada"é uma solução de monitoramento ambiental colaborativo. A distinção fundamental, entretanto, está no público-alvo e na generalidade da aplicação. O "Guardiões da Chapada"é direcionado a um grupo de usuários com certo nível de especialização (brigadistas, gestores) e focado em um único e complexo problema. O *QuixAlert!*, por sua vez, é projetado para o cidadão comum de um município e estruturado para ser uma plataforma versátil e replicável, capaz de gerenciar o amplo espectro de denúncias e serviços ambientais do cotidiano urbano, caracterizando sua contribuição como uma solução de governança ambiental local mais generalista e adaptável.

3.2 Estudo de Caso Sobre Refatoração de Software

Um estudo de caso relevante é apresentado por Wibowo *et al.* (2022), que descrevem o processo de refatoração do aplicativo móvel para agricultura "Dutataniku". A principal motivação para o trabalho foi a necessidade de melhorar o desempenho da aplicação, que apresentava problemas de alto consumo de recursos computacionais, como memória Memória de Acesso Aleatório (*RAM*) e uso da Unidade Central de Processamento (*CPU*), impactando negativamente a experiência do usuário. O objetivo era, portanto, otimizar a performance do aplicativo através de um processo de refatoração direcionado.

A metodologia adotada por Wibowo *et al.* (2022) foi sistemática. Primeiramente, eles utilizaram ferramentas de análise estática de código para identificar a presença de *code smells* na base de código do "Dutataniku". Com base nos "maus cheiros"encontrados, a equipe aplicou técnicas de refatoração específicas para corrigir esses problemas estruturais que estavam impactando a performance. A avaliação dos resultados foi realizada de forma empírica e quantitativa, medindo e comparando o consumo de *CPU* e memória *RAM* antes e depois do processo de refatoração. Os autores relataram uma melhoria significativa no desempenho, validando a eficácia da refatoração baseada na identificação de *code smells*para otimização de aplicativos móveis.

Contudo, a principal diferença e a contribuição específica do trabalho com o *QuixAlert!* residem na motivação e no escopo dos objetivos da refatoração. Enquanto o caso do "Dutataniku"teve como foco primário e quase exclusivo a melhoria de métricas de desempenho de tempo de execução (*CPU* e *RAM*), a refatoração do *QuixAlert!* foi guiada por objetivos mais amplos e estratégicos. O foco deste Trabalho de Conclusão de Curso (*TCC*) não é apenas a performance, mas primariamente a melhoria de atributos de qualidade internos de longo prazo, como a manutenibilidade, a legibilidade e, crucialmente, a preparação da arquitetura para a modularização e escalabilidade. O esforço no *QuixAlert!* visa garantir que o sistema seja não apenas eficiente, mas também fácil de evoluir, adaptar e replicar em outros municípios, um objetivo arquitetural que não é o foco central do estudo de caso de Wibowo *et al.* (2022)

3.3 Abordagens de Modularização e Escalabilidade de Software

A busca por arquiteturas de software que promovam modularidade e escalabilidade é um tema central no desenvolvimento de aplicações modernas. Nesta seção, analisamos trabalhos

que propõem ou relatam a implementação de tais arquiteturas, a fim de contextualizar e comparar com a estratégia de modularização adotada para o *QuixAlert!*.

Um estudo de caso diretamente alinhado aos objetivos deste *TCC* é o de Silva e Costa (2024), que propuseram e desenvolveram uma "Arquitetura Modular e Escalável para Apliativos Móveis no Escritório de Soluções Criativas (*ESC*)". A principal motivação do trabalho foi a observação de que, nos projetos do *ESC*, a arquitetura de software nem sempre era implementada de forma consistente, seja por falta de conhecimento ou por se ignorar aspectos importantes do processo de construção. O objetivo era, portanto, criar uma arquitetura de referência para padronizar e maximizar a qualidade do desenvolvimento de aplicativos no escritório.

A solução proposta por Silva e Costa (2024) consiste em uma arquitetura baseada no framework Flutter que fornece um esqueleto organizacional de pastas e arquivos, um sistema de rotas e a implementação de módulos independentes. Um dos pilares da arquitetura é o uso da biblioteca Fluttermodular , que permite o carregamento e descarregamento dinâmico dos módulos conforme a navegação do usuário. Isso otimiza o desempenho e o uso de recursos, pois o aplicativo carrega apenas as funcionalidades que estão em uso. Os autores destacam que essa abordagem modular facilita a manutenção, a testabilidade, a reutilização de componentes e a colaboração entre equipes, que podem trabalhar em módulos distintos de forma paralela e isolada.

O trabalho de Silva e Costa (2024) apresenta notáveis similaridades com os objetivos arquiteturais do *QuixAlert!*. Ambos os projetos reconhecem a modularidade como um princípio essencial para o desenvolvimento de sistemas flexíveis, escaláveis e de fácil manutenção. A abordagem de dividir o sistema em módulos independentes que representam funcionalidades específicas, como demonstrado na arquitetura do *ESC* com AuthModule e HomeModule, é conceitualmente idêntica à estratégia de modularizar o *QuixAlert!* em componentes como "Denúncias", "Adoção de Animais"e "Notícias". Ambos os trabalhos visam, em última instância, aumentar a qualidade e a eficiência do processo de desenvolvimento.

Apesar das semelhanças conceituais, existem diferenças cruciais no contexto e no processo que destacam a contribuição única deste *TCC*. O trabalho de Silva e Costa (2024) foca na criação de uma arquitetura de referência, um template "do zero", para ser utilizada em futuros projetos a serem desenvolvidos no *ESC*. O desafio do *QuixAlert!*, por outro lado, é o de refatorar e modularizar uma aplicação existente, com uma base de código já estabelecida e, possivelmente, com dívida técnica acumulada. Este *TCC*, portanto, não se limita a propor uma arquitetura limpa,

mas detalha o processo de transição de uma estrutura potencialmente menos organizada para uma arquitetura modular, um desafio de engenharia de software distinto e complexo.

Adicionalmente, enquanto a arquitetura proposta para o *ESC* é um modelo genérico para diversas aplicações, a modularização do *QuixAlert!* é guiada pelas necessidades específicas do domínio de gestão ambiental urbana, visando não apenas a qualidade técnica, mas também a adaptabilidade funcional para diferentes municípios. Portanto, este trabalho com o *QuixAlert!* contribui ao apresentar um caso prático de evolução arquitetural de um software cívico em operação, aplicando os princípios de refatoração para alcançar a modularidade.

3.4 Quadro Comparativo

Para consolidar a análise realizada nas seções anteriores, o Quadro 1 a seguir apresenta uma síntese comparativa dos trabalhos relacionados. A estrutura do quadro foi projetada para destacar o objeto de estudo, o eixo principal de cada trabalho, seus objetivos, e, crucialmente, sua relevância e seus diferenciais em relação ao projeto *QuixAlert!*, evidenciando a contribuição única deste trabalho.

Quadro 1 – Síntese Comparativa dos Trabalhos Relacionados

Trabalho (Autor, Ano)	**Principal Contribuição / Resultado**	**Relevância e Diferencial para o *QuixAlert!***
Dourado et al. (2018)	Plataforma para relato de problemas urbanos gerais.	**Relevância:** Modelo de interação cidadão-gestor. **Diferencial:** Foco temático ambiental e módulos de serviço.
Alencar et al. (2020)	Mapa de risco de alagamentos gerado por cidadãos.	**Relevância:** Mapeamento ambiental colaborativo. **Diferencial:** Foco na ação e resolução, não apenas na geração de dados.
Queiroz et al. (2023)	Ferramenta colaborativa para público e problema específicos.	**Relevância:** Monitoramento ambiental focado. **Diferencial:** Voltado ao cidadão comum e para múltiplos problemas do cotidiano.
Wibowo et al. (2022)	Validação de que refatoração melhora a performance.	**Relevância:** Estudo de caso de refatoração móvel. **Diferencial:** Foco em manutenibilidade e preparação para modularização, não só performance.
Silva e Costa (2024)	Proposta de uma arquitetura modular "do zero"em Flutter.	**Relevância:** Validação da abordagem de modularidade. **Diferencial:** Foco no processo de evoluir uma aplicação existente.

Fonte: Elaborado pelo autor (2025).

Em suma, a análise dos trabalhos relacionados, consolidada no quadro anterior, permite concluir que o presente trabalho ocupa um espaço de contribuição relevante. Ao integrar os domínios de software cívico ambiental com as práticas de engenharia de software de refatoração e modularização, este trabalho não apenas propõe uma solução tecnológica, mas detalha o processo de evolução arquitetural de uma aplicação existente para um contexto específico – o de municípios de pequeno e médio porte. Essa abordagem pragmática, focada na transição de um sistema em operação para uma arquitetura mais sustentável e escalável, constitui a principal lacuna identificada e que se busca preencher com o desenvolvimento do *QuixAlert!*.

4 METODOLOGIA E DESENVOLVIMENTO DO PROJETO

Este capítulo detalha a abordagem metodológica e a execução prática de todo o ciclo de evolução da aplicação *QuixAlert!*. O percurso metodológico abrange desde a análise diagnóstica da versão inicial do aplicativo, passando pelas etapas de intervenção de engenharia de software – compreendendo a refatoração, a implementação de novas funcionalidades e a modularização da arquitetura – até o delineamento do processo de validação final da versão aprimorada. O objetivo é apresentar de forma clara e sistemática o caminho percorrido para alcançar os objetivos de melhoria da qualidade interna, escalabilidade e inovação funcional do *QuixAlert!*.

4.1 Abordagem Metodológica

O presente trabalho é caracterizado como um projeto de desenvolvimento tecnológico aplicado, com foco na área de Engenharia de Software. A abordagem consiste na evolução de um produto de software existente (o aplicativo *QuixAlert!*) através de um processo estruturado de análise, refatoração, modularização e validação, visando a melhoria de seus atributos de qualidade internos e a expansão de suas funcionalidades.

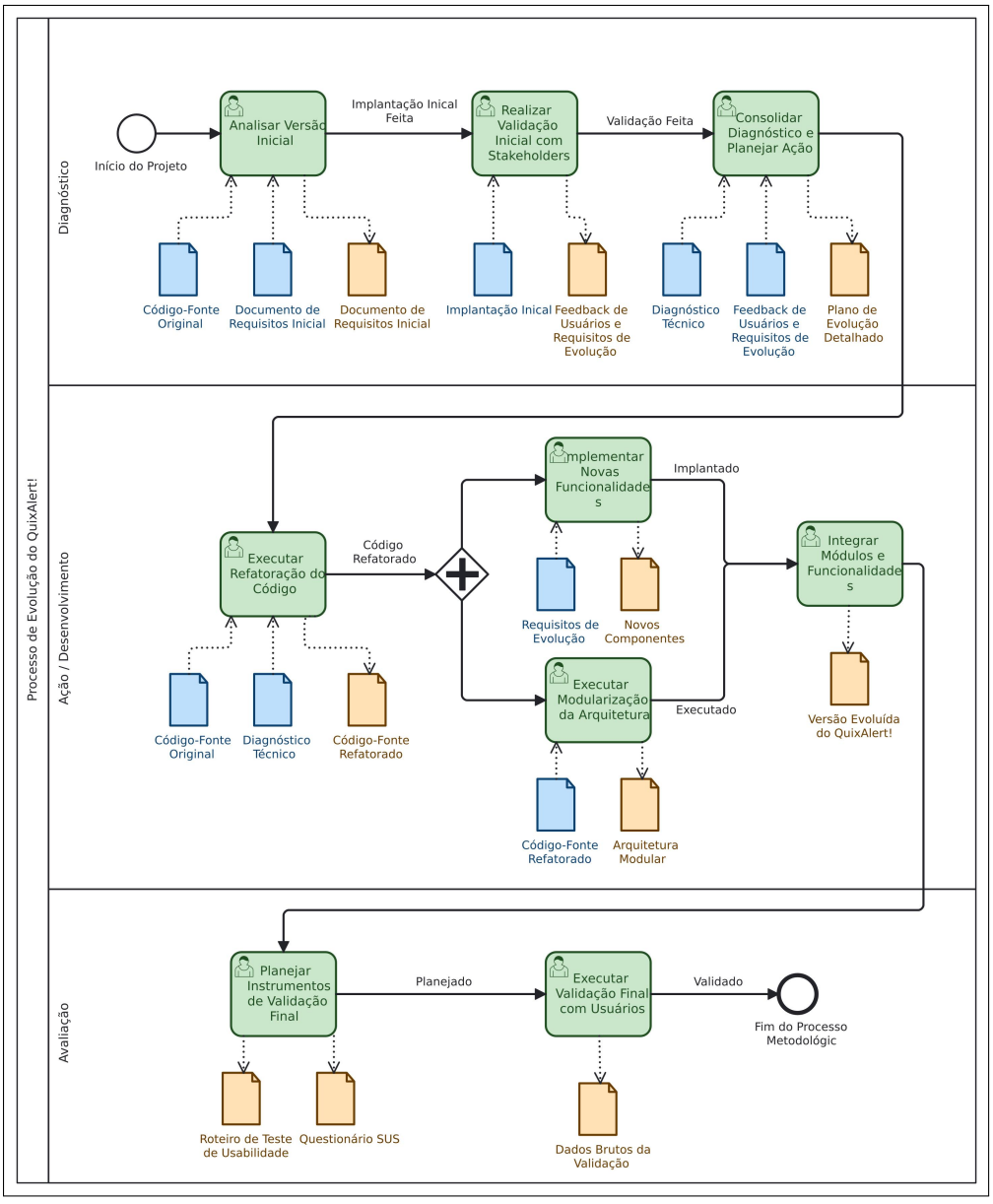
A aplicação da desta metodologia neste trabalho foi estruturada em um ciclo de **planejamento**, **ação** e **avaliação**, que guiará todas as etapas do projeto. Esse ciclo, ilustrado visualmente no fluxograma da metodologia (Figura 1), desdobra-se nas seguintes fases:

- **Fase de Diagnóstico (Observação):** A etapa inicial consistiu em uma imersão profunda na situação-problema. Isso envolveu a análise técnica da versão inicial do *QuixAlert!* para identificar débitos técnicos e *code smells*, e, simultaneamente, a realização de uma validação com os *stakeholders* (usuários e cliente) para compreender suas percepções e levantar novos requisitos.
- **Fase de Planejamento da Ação:** Com base nos problemas e oportunidades identificados no diagnóstico, foi elaborado um plano de ação detalhado. Este plano traduziu as necessidades técnicas (ex: “reduzir complexidade da classe `X`”) e funcionais (ex: “criar filtro para animais”) em um *backlog* de tarefas concretas para o ciclo de desenvolvimento, definindo as metas para as etapas de refatoração, implementação de novas funcionalidades e modularização.
- **Fase de Ação (Intervenção):** Esta é a etapa central do trabalho, onde o plano é executado.

As intervenções no software serão realizadas em três frentes principais: (a) a refatoração do código-fonte para melhorar a qualidade interna; (b) o desenvolvimento das novas funcionalidades solicitadas pelos usuários; e (c) a transformação da arquitetura para um modelo modular.

– **Fase de Avaliação (Reflexão):** A etapa final do ciclo consiste em avaliar os resultados da intervenção. Para isso, foi planejada uma nova rodada de validação com os usuários para medir o impacto das mudanças na usabilidade e na percepção de valor do aplicativo.

Figura 1 – Representação gráfica da metodologia.



Fonte: Elaborado pelo autor (2025).

4.2 Fase 1: Diagnóstico da Versão Inicial do *QuixAlert!*

A primeira fase da metodologia consiste em um diagnóstico aprofundado da aplicação *QuixAlert!* em seu estado original. Esta etapa foi fundamental para compreender o ponto de partida do projeto, suas funcionalidades, sua base técnica e, principalmente, para identificar as oportunidades de melhoria que guiaram todo o trabalho de evolução subsequente. O diagnóstico foi estruturado em três frentes: a análise do contexto e objetivos do projeto, a descrição de suas funcionalidades, e uma validação inicial com os *stakeholders*.

4.2.1 Contexto do Problema e Motivação do Projeto

O projeto *QuixAlert!* foi concebido como uma resposta direta a desafios socioambientais observados no município de Quixadá. O "Domínio do Projeto", conforme descrito no documento de requisitos (Apêndice 8), destaca o descarte inadequado de resíduos sólidos como uma problemática central, cujas consequências incluem a obstrução de canais de drenagem, resultando em alagamentos, além da poluição do solo e da água.

A plataforma foi, portanto, idealizada como uma solução tecnológica e colaborativa para capacitar os cidadãos a se tornarem agentes ativos na fiscalização ambiental, permitindo o relato e a documentação desses problemas.

Adicionalmente, o escopo do projeto foi expandido para abranger outra questão de grande relevância local: o bem-estar de animais de rua. O sistema se propôs a oferecer ferramentas para combater o abandono de animais e promover a adoção responsável, conectando animais em busca de um lar com cidadãos interessados.

Além do aplicativo **QuixAlert!**, o sistema conta com uma plataforma web dedicada à gestão dos dados enviados pelos usuários. Essa plataforma, utilizada exclusivamente pela equipe de colaboradores da AMMA (Autarquia Municipal de Meio Ambiente), foi projetada para apoiar o tratamento, acompanhamento e resolução das ocorrências reportadas. Por meio dela, os colaboradores podem visualizar denúncias geolocalizadas, acompanhar o status dos casos, registrar ações realizadas e gerar relatórios que auxiliam na tomada de decisões estratégicas.

Embora esta plataforma seja parte fundamental do ecossistema do *QuixAlert!*, sua análise detalhada não é o foco deste trabalho. Ainda assim, sua existência e integração com o aplicativo são essenciais para garantir a efetividade do processo de fiscalização e resposta.

Essa estrutura integrada — composta pelo aplicativo voltado à participação cidadã e

pela plataforma interna de gestão — fortalece o sistema de fiscalização ambiental e proteção animal, promovendo um ciclo virtuoso de colaboração entre a sociedade civil e o poder público.

4.2.2 Cliente e Objetivos do Sistema

O cliente e principal parceiro institucional do projeto é a **Autarquia Municipal de Meio Ambiente de Quixadá *AMMA***, órgão governamental responsável pela gestão, fiscalização, licenciamento e educação ambiental no município. A *AMMA* desempenha um papel crucial na proteção da fauna e flora locais, incluindo o desenvolvimento de programas de castração e resgate de animais de rua. Esta parceria foi fundamental para alinhar as funcionalidades do *QuixAlert!* com as necessidades reais de gestão da autarquia.

Com base neste contexto, os objetivos centrais da versão inicial do *QuixAlert!* foram definidos como:

- Desenvolver uma plataforma intuitiva para denúncias de problemas ambientais;
- Facilitar a interação eficiente entre os cidadãos e as autoridades competentes (*AMMA*);
- Promover a conscientização ambiental através de recursos informativos;
- Fornecer um canal confiável de notícias sobre questões ambientais em Quixadá.

4.2.3 Funcionalidades da Versão Inicial

A versão inicial do *QuixAlert!*, possui um conjunto de funcionalidades essenciais, detalhadas na seção “Escopo do Sistema” do documento de requisitos. Essas funcionalidades podem ser agrupadas nos seguintes módulos principais:

- **Módulo de Gestão de Denúncias:** O cerne do sistema, permitindo aos usuários (como a persona **Pedro, o Defensor Ambiental**) reportar problemas ambientais diversos e casos de maus-tratos a animais. A funcionalidade permitia o envio de texto, imagens e vídeos para detalhar a ocorrência. O sistema prever um fluxo onde as denúncias seriam encaminhadas às autoridades e validadas por um especialista (persona **Carolina, a Verificadora Cautelosa**), com o usuário podendo acompanhar o status da sua denúncia, uma imagem do sistema demonstrado na figura 2.
- **Módulo de Adoção:** Focado na causa animal, este módulo oferece recursos para facilitar o processo de adoção. Usuários (como a persona **Luiza, a Amante dos Animais**) poderm visualizar uma lista de animais disponíveis, com fotos e informações, e iniciar um pedido de adoção diretamente pelo aplicativo (Figura 3).

Figura 2 – Tela de Denúncia do *QuixAlert!*

Fonte: Elaborado pelo autor (2025).

Figura 3 – Tela de Adoções do *QuixAlert!*

Fonte: Elaborado pelo autor (2025).

– **Módulo Informativo e de Engajamento:** Visando à conscientização, o aplicativo possuí uma seção de notícias para manter os usuários (como a persona **Roberto, o Observador Informado**) atualizados sobre questões ambientais locais. Também prever recursos de compartilhamento de informações entre usuários para incentivar o engajamento comunitário (Figura 4).

Figura 4 – Tela de Notícias do *QuixAlert!*

Fonte: Elaborado pelo autor (2025).

– **Módulo de Solicitações de Documentos:** Uma funcionalidade mais burocrática, mas importante para a interação com a AMMA, que permite ao usuário iniciar solicitações de documentos como o licenciamento ambiental.

Figura 5 – Tela Solicitação de Documentos do *QuixAlert!*

Fonte: Elaborado pelo autor (2025).

4.2.4 Arquitetura e Diagnóstico Técnico Inicial

A análise técnica da versão inicial do *QuixAlert!* foi uma etapa crucial do diagnóstico para identificar as características da base de código e as oportunidades de melhoria estrutural.

4.2.4.1 Tecnologias e Arquitetura

versão original da aplicação foi desenvolvida de forma nativa para a plataforma Android, utilizando a linguagem *Kotlin* no ambiente de desenvolvimento Android Studio. A infraestrutura de *backend*, incluindo banco de dados e armazenamento de arquivos, foi suportada pelos serviços da plataforma Firebase do Google.

Arquiteturalmente, o sistema foi implementado como uma aplicação monolítica, concentrando todo o código-fonte em um único módulo (`:app`). Internamente, o projeto seguia um padrão em camadas, com organização de pacotes que separava responsabilidades entre *View*, *ViewModel*, *Repository*, *Service* e *Model*. Embora essa estrutura em camadas represente uma boa prática de design, a natureza monolítica da aplicação resultava em alto acoplamento entre funcionalidades. Por exemplo, alterações no fluxo de denúncias poderiam impactar diretamente o fluxo de adoção, devido à ausência de uma separação efetiva entre os domínios.

Para avaliar objetivamente os impactos dessa arquitetura e subsidiar decisões de refatoração, foi utilizada a ferramenta de análise estática de código ***SonarQube***. O SonarQube é uma plataforma de código aberto amplamente adotada pela indústria para inspeção contínua da qualidade do código-fonte. Ele permite identificar e quantificar problemas através de um conjunto de métricas automatizadas, fornecendo um panorama claro da saúde do projeto.

A análise do código realizada pelo SonarQube revelou diversos indicadores relevantes, como **bugs**, **vulnerabilidades**, **pontos sensíveis à segurança (security hotspots)** e **code smells**. Cada uma dessas categorias oferece uma perspectiva distinta: bugs apontam para falhas lógicas que podem gerar erros em tempo de execução; vulnerabilidades indicam brechas que podem ser exploradas por atacantes; security hotspots sinalizam trechos sensíveis que requerem revisão manual; e os *code smells* expõem problemas de design que tornam o código mais difícil de manter e evoluir.

O uso do *SonarQube*, portanto, proporcionou uma base objetiva e rastreável para diagnosticar os pontos frágeis da aplicação original e orientar o processo de refatoração com foco em modularização, desacoplamento e melhoria da manutenibilidade.

4.2.5 Diagnóstico com Stakeholders

Paralelamente à análise técnica, a fase de diagnóstico incluiu um levantamento abrangente com os *stakeholders* do projeto para avaliar a versão inicial do *QuixAlert!* e elicitar requisitos para sua evolução. Foi utlizado uma abordagem que visa coletar dados da usabilidade da ferramenta. Essa abordagem foi fundamental para garantir que as melhorias na aplicação estivessem alinhadas às necessidades reais dos usuários e do cliente.

4.2.5.1 Planejamento e Execução do diagnóstico

O diagnóstico foi planejado para coletar dados de três grupos distintos que compõem a população de interesse do *QuixAlert!*:

- **Especialistas da Área Ambiental:** Incluindo membros da AMMA e estudantes de Engenharia Ambiental, para obter uma perspectiva técnica sobre o domínio do problema.
- **Especialistas em Desenvolvimento de Software:** Para obter feedback sobre a usabilidade e a viabilidade técnica das funcionalidades.
- **Usuários Finais:** Cidadãos de Quixadá, para compreender as necessidades e expectativas do público-alvo principal.

Para a coleta de dados, foram utilizados dois instrumentos complementares, conforme detalhado no documento de requisitos do projeto:

- **Questionários Online:** Aplicados via Google Forms para um grupo de 15 participantes, permitindo a coleta de dados quantitativos e qualitativos de forma escalável.
- **Entrevistas Semiestruturadas:** Realizadas com *stakeholders*-chave, como a presidente da AMMA e especialistas da área, para aprofundar a compreensão sobre as necessidades de domínio e os desafios da gestão local.

Além desses instrumentos, o diagnóstico foi complementado por testes de usabilidade na versão inicial do aplicativo, envolvendo representantes dos diferentes perfis de *stakeholders* para observar a interação direta com a aplicação.

4.3 Fase 2: Ação e Desenvolvimento da Nova Versão

Com base no diagnóstico completo, a fase de ação está sendo executada em três frentes de trabalho paralelas e interdependentes: refatoração, modularização e implementação de novas funcionalidades.

4.3.1 O Processo de Refatoração

O primeiro passo da intervenção consiste na refatoração do código-fonte, orientada pelos *code smells* identificados na análise técnica, com o apoio da ferramenta SonarQube. O objetivo central dessa etapa é aprimorar a qualidade interna do software, tornando o código mais legível, modular e de fácil manutenção.

Um exemplo prático foi a refatoração de um método extenso responsável por criar, validar e enviar uma denúncia. Esse método apresentava mais de 50 linhas e foi classificado como um "Método Longo", um dos smells mais comuns. Utilizando a técnica Extract Method (Fowler, 2018), a lógica foi decomposta em métodos menores e mais coesos.

Antes: Um único método `enviarDenunciaCompleta()`, com mais de 50 linhas de código. **Depois:** O método original foi dividido em métodos auxiliares como `validarCamposDenuncia()`, `formatarDadosParaEnvio()` e `submeterDenunciaAoServidor()`, proporcionando maior legibilidade e facilitando a realização de testes unitários.

Esse processo será replicado de forma sistemática em toda a base de código, preparando o sistema para futuras evoluções arquiteturais. As atividades de refatoração estão sendo conduzidas na IDE Android Studio, com o uso da linguagem Kotlin.

4.3.2 O Processo de Modularização

Com o objetivo estratégico de tornar o *QuixAlert!* uma plataforma flexível e adaptável, a transição da arquitetura monolítica para uma arquitetura modular será um passo crucial. No desenvolvimento nativo Android, a modularização é alcançada por meio da criação de *Módulos de Biblioteca* (*Android Library Modules*) dentro do projeto no Android Studio.

A aplicação será decomposta de forma que cada funcionalidade principal se tornasse um módulo independente. Serão criados módulos como:

- `:core`: Módulo base contendo código compartilhado, como modelos de dados, classes de rede e componentes de UI comuns.
- `:feature_denuncias`: Módulo responsável por todo o fluxo de criação e visualização de denúncias.
- `:feature_adocao`: Módulo contendo as telas e a lógica para a adoção de animais.
- `:feature_noticias`: Módulo para a exibição de notícias e conteúdo informativo.

O módulo principal, `:app`, que antualmente concetra todo o código da aplicação, passará a ter como responsabilidade integrar os módulos de funcionalidade (*feature modules*) e gerenciar a navegação entre eles, utilizando o *Android Navigation Component*.

Dessa forma, cada módulo pode ser desenvolvido, testado e mantido de forma isolada. Mais importante ainda, essa estrutura permite que, em futuras implementações em outros municípios, um módulo como o `:feature_adocao` possa ser facilmente "desligado" ou não incluído na versão final do aplicativo, sem comprometer o funcionamento do restante do sistema.

4.3.3 Implementação das Novas Funcionalidades

As novas funcionalidades, levantadas na validação inicial, serão desenvolvida já dentro da nova arquitetura modular. Algumas das funções levantadas são:

- O filtro de animais e a galeria de imagens que serão implementadas no módulo `Adocoes`.
- A opção de doação será adicionada como um novo componente, também ligado ao módulo `Adocoes`, com sua própria tela e fluxo.
- A função de pesquisa global será implementada como um componente de UI compartilhado, acessível a partir da barra de navegação principal.

4.3.4 Fase 3: Planejamento da Avaliação da Versão Evoluída

A etapa final do ciclo da metodologia consiste em avaliar a eficácia das intervenções realizadas. Para isso, foi planejada uma avaliação em duas frentes: uma avaliação técnica da qualidade interna do software e uma validação com os usuários para medir a percepção de usabilidade e valor.

4.3.4.1 Avaliação Técnica da Qualidade do Código

Para medir o impacto do processo de refatoração descrito na Seção 4.3.1, foi planejada uma segunda análise estática do código-fonte utilizando a plataforma *SonarQube*. O objetivo desta etapa é comparar as métricas de qualidade (número de *code smells*, dívida técnica, duplicação de código, entre outros) coletadas após a refatoração com a linha de base estabelecida na fase de diagnóstico (ver Seção 4.2.4).

Essa comparação quantitativa serve como evidência objetiva para avaliar o sucesso

das melhorias na qualidade interna do código. Os resultados detalhados desta análise comparativa estão apresentados como resultados no Capítulo **??**.

4.3.4.2 Validação com Usuários

Para avaliar a percepção dos usuários sobre a aplicação evoluída, será conduzida uma nova rodada de validação com participantes de perfis semelhantes aos da etapa inicial (cidadãos, especialistas e cliente). O objetivo central é medir o impacto das modificações realizadas na usabilidade e na percepção de valor do aplicativo. A metodologia desta validação compreende os seguintes passos:

4.3.4.2.1 Teste de Usabilidade Guiado

O teste de usabilidade será conduzido para avaliar a eficácia e a eficiência da interação dos usuários com as funcionalidades do *QuixAlert!*, tanto as já existentes quanto as novas.

Roteiro de Tarefas

Será utilizado um roteiro de tarefas pré-definido, que abrange os fluxos de trabalho críticos e as novas funcionalidades implementadas (como o Módulo de Chat e o Sistema de Filtros de Adoção). O roteiro de tarefas, detalhado no Apêndice 10, foi desenhado para simular um cenário de uso real, medindo a capacidade do usuário em executar tarefas de fiscalização, comunicação e serviços de bem-estar animal.

Protocolo Think-Aloud

Durante as sessões, será aplicado o protocolo *think-aloud* (pensamento em voz alta). Este protocolo estimula os usuários a verbalizarem seus pensamentos, expectativas, dificuldades e sentimentos enquanto interagem com o sistema. A observação direta e o registro desses comentários são vitais para identificar *gaps* de usabilidade..

Aplicação de Questionário *SUS*

Ao final do teste de usabilidade, a percepção subjetiva de usabilidade da aplicação será medida por meio do questionário *SUS*. O *SUS* é um questionário com o objetivo de medir quantitativamente a usabilidade percebida; o questionário está demonstrado no Apêndice 9.

Os resultados serão analisados à luz dos *benchmarks* propostos na literatura para determinar a aceitabilidade e a classificação da usabilidade da versão evoluída do *QuixAlert!*. O questionário foi impresso e entregue diretamente aos usuários durante a sessão de teste.

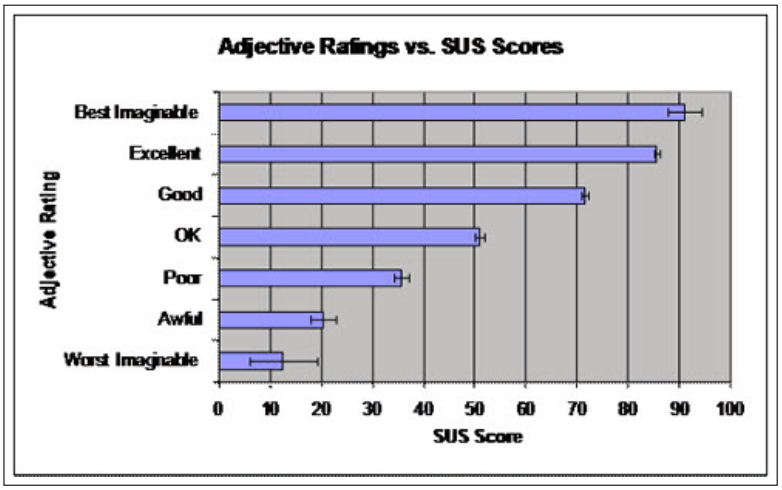
Cálculo da Pontuação SUS

A pontuação total do *SUS* (em uma escala de 0 a 100) é obtida através de uma fórmula específica que considera o tom misto das dez questões (cinco positivas e cinco negativas). O cálculo é realizado em quatro etapas:

1. **Contribuição dos Itens:** Para cada item do questionário, calcula-se a contribuição do escore na escala de 1 a 5:
   - *Itens ímpares* (positivos: 1, 3, 5, 7, 9): a contribuição é obtida subtraindo-se 1 da nota atribuída pelo usuário.
   - *Itens pares* (negativos: 2, 4, 6, 8, 10): a contribuição é obtida subtraindo-se a nota atribuída pelo usuário de 5.
2. **Soma Total:** Calcula-se a soma dos valores de contribuição dos dez itens.
3. **Escore Final:** Multiplica-se o valor total da soma por 2,5, obtendo-se assim a pontuação final do *SUS*, que varia de 0 a 100.

Critério de Classificação Adjetiva: Para interpretar o escore numérico obtido pelo *SUS*, será utilizada a Escala de Classificação Adjetiva proposta (Bangor *et al.*, 2009), que permite categorizar a usabilidade percebida em níveis, conforme descrito no Quadro 2.

Figura 6 – Critérios de Classificação do *SUS*



Fonte: (Bangor *et al.*, 2009).

Conforme ilustrado na Figura 6, que apresenta a distribuição dos escores obtidos no

questionário SUS, é possível observar diferentes níveis de percepção de usabilidade. A partir dessa análise, a classificação detalhada dos resultados é organizada na Quadro 2, permitindo uma interpretação mais precisa do desempenho da aplicação em termos de satisfação do usuário.

Quadro 2 – Escala de Classificação Adjetiva do SUS

Avaliação Adjetiva (Inglês)	**Faixa SUS**	**Tradução sugerida (Português)**
Worst Imaginable	0–12.5	Pior Imaginável
Awful	12.6–25	Horrível
Poor	25.1–38	Ruim
OK	38.1–50	Aceitável
Good	50.1–72.5	Bom
Excellent	72.6–85	Excelente
Best Imaginable	85.1–100	Melhor Imaginável

Fonte: Elaborado pelo autor (2025).

5 RESULTADOS

Este capítulo apresenta os resultados obtidos em cada fase do processo de desenvolvimento evolutivo do *QuixAlert!*, detalhado na metodologia. A exposição inicia-se com a apresentação dos dados da análise diagnóstica da versão inicial, seguida pelos resultados técnicos do processo de refatoração, culminando com a apresentação da nova arquitetura modular e das novas funcionalidades implementadas.

5.1 Resultados da Análise de Qualidade do Código

Conforme planejado na metodologia, para obter uma avaliação objetiva da qualidade interna da aplicação, foi realizada uma análise estática do código-fonte com a plataforma *SonarQube*. Esta análise foi executada em dois momentos: antes e depois do processo de refatoração, permitindo uma comparação direta do impacto das melhorias.

5.1.1 Análise Quantitativa da Qualidade do Código (SonarQube)

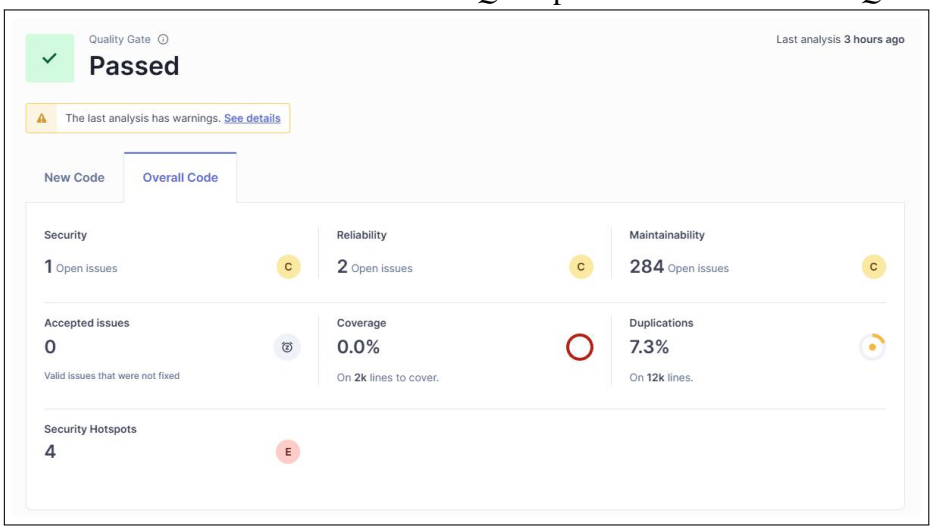
A inspeção manual do código-fonte da versão inicial da aplicação (Silva, 2025), aliada à análise da estrutura monolítica, revelou a existência de dívida técnica e de diversos *code smells*. Para quantificar objetivamente este estado inicial, foi realizada uma análise estática com a ferramenta *SonarQube*, estabelecendo uma linha de base (*baseline*) para métricas de qualidade como o número de *code smells*, o tempo estimado de dívida técnica e o percentual de duplicação de código.

5.1.1.1 Análise da Versão Inicial (Baseline)

A primeira análise foi executada sobre a base de código original do *QuixAlert!* para estabelecer uma linha de base quantitativa. O painel de resultados do *SonarQube* (Figura 3) apresentou uma visão geral da saúde do projeto.

A análise do código, com aproximadamente 11 mil linhas, revelou um conjunto significativo de problemas. Embora o *Quality Gate* tenha sido aprovado com uma nota geral "A" em manutenibilidade (um cálculo que considera a proporção da dívida técnica em relação ao tamanho do código), a análise detalhada dos problemas apontou para questões críticas que justificam o esforço de refatoração. Os principais indicadores foram:

Figura 7 – Painel de Resultados do *SonarQube* para a Versão Inicial do *QuixAlert!*



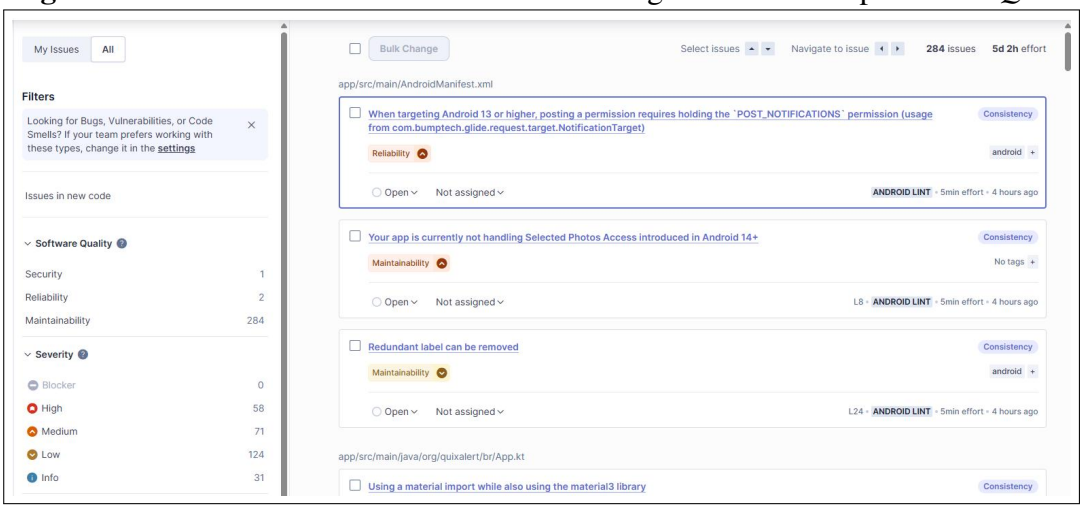
Fonte: Elaborado pelo autor (2025).

- **Manutenibilidade:** Foram identificados **284 *Code Smells***.
- **Confiabilidade:** Foram encontrados **2 bugs potenciais** no código.
- **Segurança:** A análise apontou **1 vulnerabilidade** e, mais criticamente, **4 *Security Hotspots*** com nota "E", indicando pontos que exigem revisão manual urgente por apresentarem riscos de segurança.
- **Duplicação de Código:** O projeto apresentava um índice de **7,3% de duplicação**, o que representa uma quantidade considerável de código repetido que dificulta a manutenção.
- **Cobertura de Testes:** A análise revelou uma cobertura de testes de **0,0%**, indicando a ausência de uma suíte de testes automatizados, o que representa um risco significativo para a evolução segura do software.

Aprofundando a análise dos **284 *code smells*** (Figura 8), o *SonarQube* os classificou por severidade, revelando que **58 deles eram de prioridade alta (críticos)** e **71 de prioridade média**, indicando que uma parcela substancial dos problemas não era trivial e poderia levar a dificuldades significativas de manutenção no futuro.

Esses dados quantitativos serviram como uma confirmação objetiva do diagnóstico qualitativo apresentado na Seção 4.2.4 e forneceram alvos claros para o processo de refatoração. O foco da intervenção foi, portanto, a eliminação sistemática desses *code smells*, com prioridade para aqueles de maior severidade. Para além o diagnóstico técnico evidenciou que, para garantir o cumprimento dos requisitos não funcionais de **Manutenibilidade** e **Escalabilidade** e para viabilizar os objetivos estratégicos do projeto, uma intervenção arquitetural, baseada em refatoração e modularização, era indispensável.

Figura 8 – Detalhamento dos Problemas de Código Identificados pelo *SonarQube*



Fonte: Elaborado pelo autor (2025).

5.1.2 Processo de Refatoração na Prática

A redução da dívida técnica e a melhoria da qualidade interna do código foram alcançadas através de um processo sistemático de refatoração. A abordagem focou em atacar os problemas de maior impacto, guiando-se pela severidade e pela natureza dos *smells* apontados pelo *SonarQube*. A seguir, são detalhados exemplos práticos das principais técnicas aplicadas.

5.1.2.1 Corrigindo Método Longo para Melhorar a Coesão

Um dos *code smells* mais evidentes na versão inicial do sistema era o *Long Method*, caracterizado por funções extensas que acumulavam múltiplas responsabilidades. Para mitigar esse problema, aplicou-se amplamente a técnica *Extract Method*, conforme descrita por Fowler (Fowler, 2018).

No Código-Fonte 1, observa-se um trecho da `DenunciaActivity` antes da refatoração, onde validação, criação do objeto e feedback ao usuário coexistiam em um único bloco monolítico. Já o Código-Fonte 2 apresenta a versão reorganizada, na qual o método original é dividido em funções menores, coesas e semanticamente claras. Com isso, cada rotina passou a expressar um propósito único, facilitando manutenção, testes e evolução futura do código.

Código-fonte 1 – Exemplo de Metodo Longo antes da refatoracao

```
1 private fun enviarDenuncia() {
2     if (binding.titulo.text.isNullOrEmpty()) { /* ...codigo de
          erro... */ return }
```

```
    if (binding.descricao.text.isNullOrEmpty()) { /* ...codigo de erro... */ return }
    ...

    val denuncia = Denuncia(...)

    viewModel.salvarDenunciaNoFirebase(denuncia)
    Toast.makeText(this, "Enviado!", Toast.LENGTH_SHORT).show()
    finish()
}
```

Código-fonte 2 – Refatoração com “Extract Method” aplicada

```
% Dentro de DenunciaActivity.kt (versao refatorada)
private fun enviarDenuncia() {
    if (!validarCamposFormulario()) return
    val novaDenuncia = criarObjetoDenunciaAPartirDaUI()
    viewModel.submeterDenuncia(novaDenuncia)
    exibirFeedbackEFecharTela()
}

% Metodos extraidos com responsabilidades unicas
private fun validarCamposFormulario(): Boolean { /* ... */ }
private fun criarObjetoDenunciaAPartirDaUI(): Denuncia { /* ... */ }
private fun exibirFeedbackEFecharTela() { /* ... */ }
```

5.1.2.2 Centralizando Lógica para Reduzir Código Duplicado

Para combater o alto índice de duplicação, a técnica *Extract Class* foi utilizada para criar classes utilitárias (*Helpers/Utils*) no módulo `:core`. Lógicas que se repetiam em várias partes do aplicativo, como a formatação de datas e a validação de strings, foram movidas para métodos estáticos nessas novas classes. Essa centralização garante consistência e permite que

qualquer futura alteração na lógica seja feita em um único local, facilitando drasticamente a manutenção.

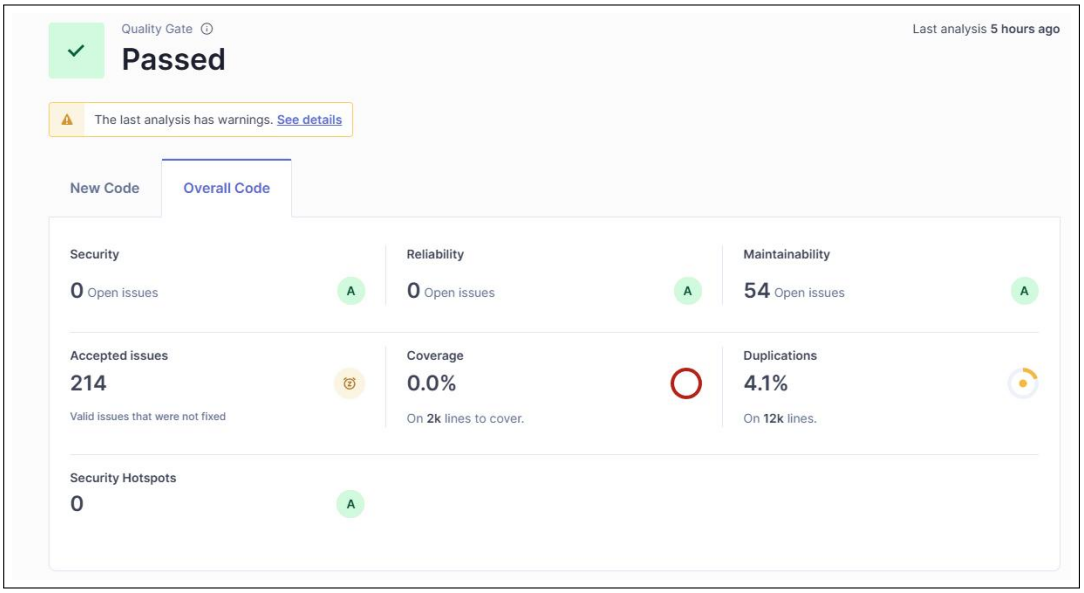
5.1.2.3 Aumentando a Intencionalidade e a Legibilidade

Para atacar os problemas de *Intencionalidade*, foram aplicadas as técnicas *Rename Variable/Method* e *Replace Magic Number with Symbolic Constant*. Nomes pouco descritivos foram substituídos por identificadores que revelam seu propósito, e valores literais no código (por exemplo, `status == 2`) foram substituídos por constantes nomeadas (por exemplo, `status == DenunciaStatus.EM_ANALISE`). Essas mudanças tornaram o código mais autoexplicativo e reduziram a necessidade de comentários redundantes.

5.1.2.4 Análise da Versão Refatorada e Comparativo

Após a execução do processo de refatoração descrito na metodologia (Seção 4.3.1), a nova versão do código-fonte foi novamente submetida à análise do *SonarQube*, utilizando os mesmos critérios da análise inicial. O objetivo foi medir o impacto direto das melhorias implementadas na qualidade interna do código. A Figura 9 apresenta o painel de resultados da versão refatorada.

Figura 9 – Painel de Resultados do *SonarQube* para a Versão Refatorada do *QuixAlert!*



Fonte: Elaborado pelo autor (2025).

Uma comparação direta entre as métricas da versão inicial e da versão refatorada

revela uma melhora substancial em todos os indicadores de qualidade do código. A Tabela 2 consolida esses dados e evidencia o impacto positivo do trabalho de engenharia de software realizado.

Tabela 2 – Comparativo das Métricas de Qualidade Antes e Depois da Refatoração

Métrica	**Valor Inicial**	**Valor Final**	**Variação**
Bugs (Reliability)	2	0	−100%
Vulnerabilidades (Security)	1	0	−100%
Security Hotspots	4	0	−100%
Code Smells (Maintainability)	284	54	−80,9%
Duplicação de Código (%)	7,3%	4,1%	−43,8%
Dívida Técnica	`[124h]`	`[18h]`	85,4%

Fonte: Elaborado pelo autor (2025).

5.1.3 Discussão dos Resultados Técnicos

A comparação das métricas valida o sucesso da refatoração. A esperada redução no número de *Code Smells* — especialmente os de prioridade crítica e média — e na *Dívida Técnica* demonstra que o código está objetivamente mais limpo e manutenível. A diminuição no percentual de duplicação indica que a aplicação de técnicas como *Extract Method* foi eficaz.

Esses resultados evidenciam que o primeiro objetivo da fase de ação — melhorar a qualidade interna do código — foi alcançado, criando uma base sólida para a modularização e para as próximas etapas de validação.

5.2 Resultados do Processo de Modularização

A refatoração (Seção 5.1) serviu como um pré-requisito técnico essencial, estabelecendo as bases estruturais e de organização necessárias para o processo de modularização. Essa etapa permitiu eliminar redundâncias, reduzir o acoplamento entre componentes e aprimorar a legibilidade do código, criando um ambiente mais propício para a evolução contínua do sistema. A modularização, por sua vez, representou uma intervenção arquitetural de maior escala, responsável por segmentar o monólito original em unidades funcionais independentes e coerentes, cada uma com responsabilidades bem definidas e interfaces claras de comunicação. Essa transformação foi fundamental para viabilizar os objetivos estratégicos de escalabilidade, adaptabilidade e manutenção sustentável do *QuixAlert!*, permitindo que futuras expansões e integrações possam

ser realizadas de forma mais ágil e controlada, sem comprometer a estabilidade do sistema como um todo.

5.2.1 Análise da Arquitetura Inicial

O estado inicial do projeto, embora funcional, apresentava desafios arquiteturais significativos, característicos de uma abordagem monolítica. O ponto mais crítico era o arquivo `RenderScreen.kt`, um componente centralizador que acumulava uma quantidade excessiva de responsabilidades, violando o Princípio da Responsabilidade Única (SRP).

Este componente funcionava como um *God Object* e um roteador manual, utilizando uma extensa estrutura `when` para decidir qual tela (`@Composable`) renderizar com base em uma variável de estado (`currentScreen: String`).

As principais desvantagens desta abordagem eram:

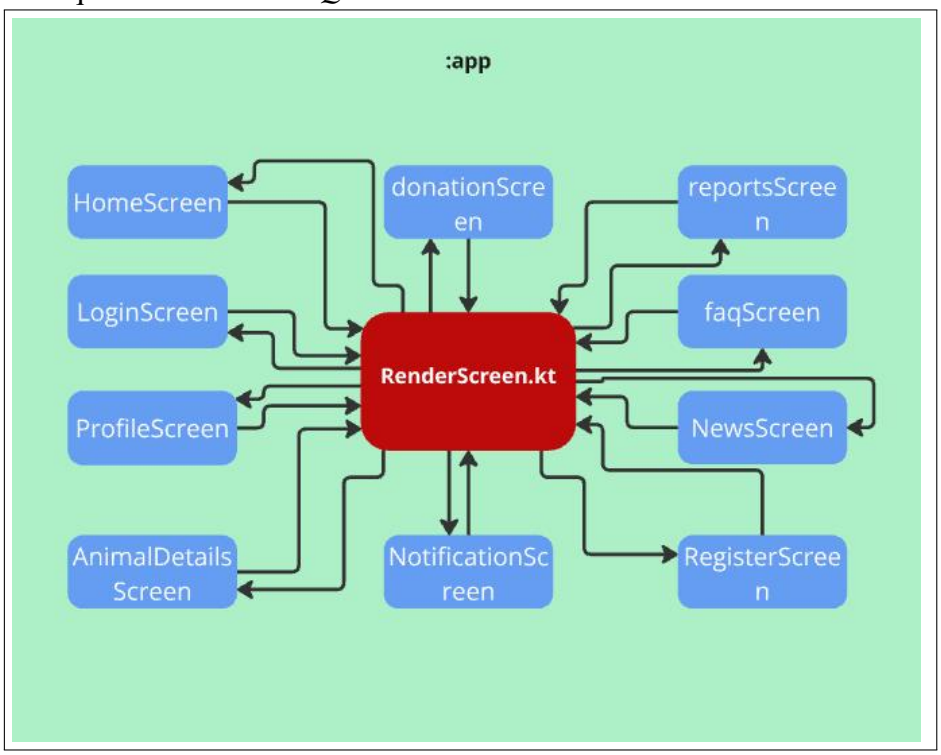
1. **Alto Acoplamento:** Todas as telas possuíam uma dependência direta do `RenderScreen`, e este, por sua vez, precisava conhecer todos os parâmetros de todas as telas, criando uma teia complexa de dependências.
2. **Baixa Coesão:** A lógica de navegação, passagem de estado e renderização de interface estavam misturadas em um único local, dificultando o entendimento e a modificação de fluxos específicos.
3. **Dificuldade de Manutenção e Escalabilidade:** Adicionar ou modificar uma tela exigia alterações em múltiplas partes do `RenderScreen`, aumentando o risco de introdução de *bugs* em funcionalidades não relacionadas.
4. **Complexidade Ciclomática Elevada:** A estrutura `when` com dezenas de casos tornava o código difícil de testar e raciocinar sobre.

5.2.1.1 Transição da Arquitetura Monolítica para Modular

A versão inicial do *QuixAlert!* foi implementada como uma aplicação monolítica, concentrando todo o código-fonte em um único módulo (`:app`). Embora seguisse um padrão em camadas e pacotes (como `domain` e `presentation`), a natureza monolítica resultava em alto acoplamento entre funcionalidades.

A evolução arquitetural teve como objetivo principal romper esse acoplamento, resultando em um modelo modular nativo para Android (Kotlin) e *Jetpack Compose*. A biblioteca *Jetpack Navigation Compose* foi introduzida para gerenciar a navegação de forma declarativa e

Figura 10 – Arquitetura Inicial do *QuixAlert!*



Fonte: Elaborado pelo autor (2025).

desacoplada, promovendo maior clareza e flexibilidade estrutural.

A solução implementada desmantelou a estrutura monolítica em favor de uma arquitetura modular por funcionalidade (*feature modules*), com uma base de camadas compartilhadas (*core modules*). A nova estrutura do projeto foi organizada da seguinte forma:

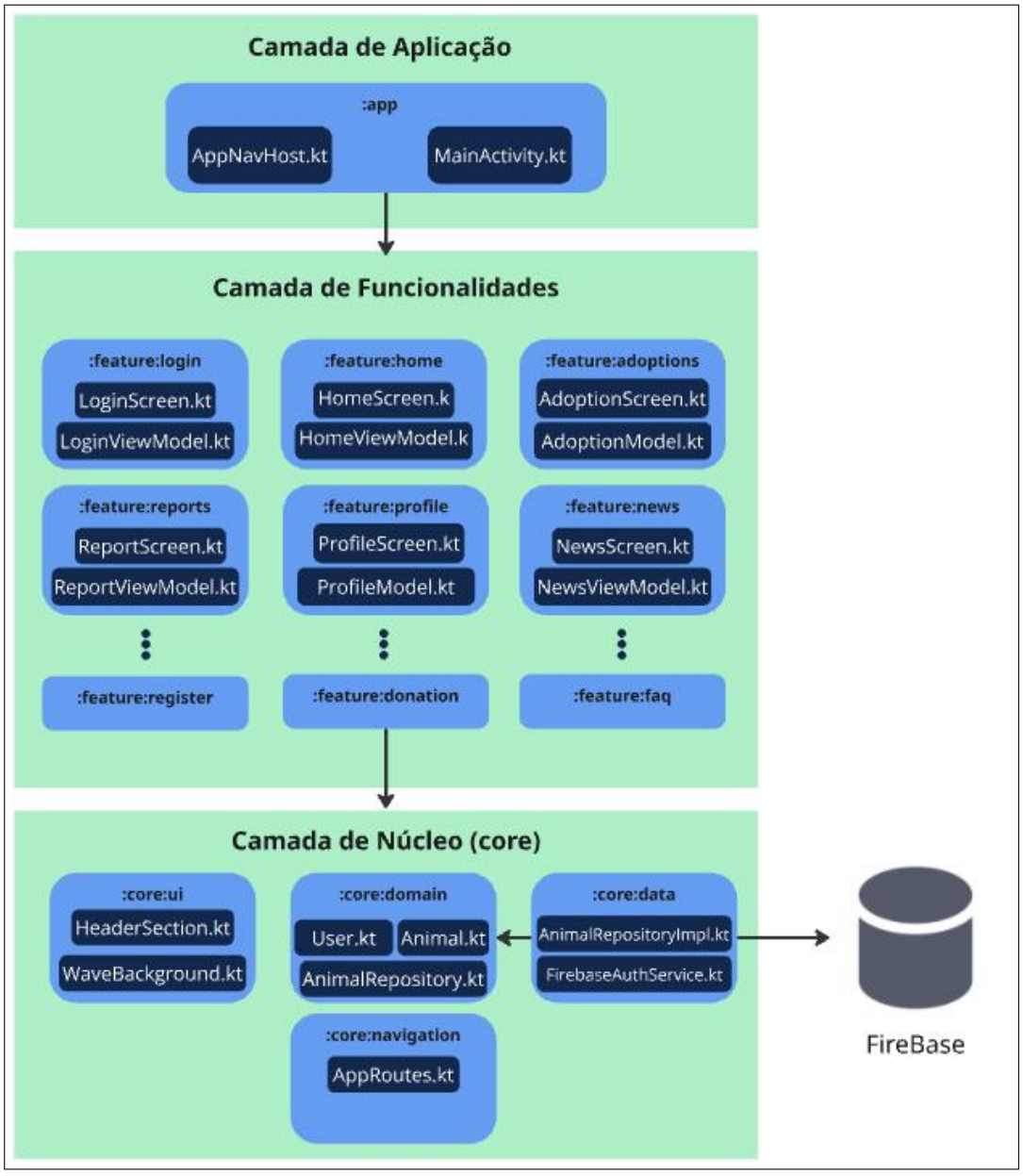
- **Módulo** `:app`**:** Tornou-se o módulo integrador. Sua responsabilidade foi reduzida a hospedar a atividade principal, configurar o tema e, crucialmente, definir o `NavHost`, que funciona como o contêiner para o gráfico de navegação do aplicativo.
- **Módulos de Funcionalidade** (`:feature_*`)**:** Módulos verticais independentes que encapsulam uma funcionalidade completa. Exemplos incluem: `:feature:login`, `:feature:home`, `:feature:profile` e `:feature:notifications`. Cada módulo contém suas próprias telas (`@Composable`), `ViewModels` e lógica de estado, sendo independentes uns dos outros.
- **Módulos de Camada** (`:core_*`)**:** Contêm o código compartilhado e reutilizável em todo o sistema, servindo como base para os módulos de funcionalidade.
  - `:core:ui`**:** Contém componentes de interface reutilizáveis (por exemplo, `HeaderSection`, `WaveBackground`), temas, cores e outros recursos visuais compartilhados.
  - `:core:domain`**:** Define os modelos de dados (por exemplo, `User`, `Animal`) e as interfaces dos repositórios, formando a camada de negócios do aplicativo, sem

dependências diretas do Android.

– `:core:data`**:** Implementa as interfaces do `:core:domain`, gerenciando as fontes de dados (*Firestore*, *Firebase Auth*, APIs REST) e a lógica de acesso a dados.

O fluxo de dependências foi estritamente definido: o módulo `:app` depende dos módulos `:feature`, e os módulos `:feature` dependem dos módulos `:core`. Nenhum módulo `:feature` depende de outro, e os módulos `:core` não possuem conhecimento das funcionalidades, garantindo o baixo acoplamento e a independência entre componentes.A nova arquitetura está representada na Figura 11.

Figura 11 – Arquitura Modular Atual do *QuixAlert!*



Fonte: Elaborado pelo autor (2025).

5.2.2 Resultados e Benefícios da Arquitetura Modular

A transição para a arquitetura modular resultou em melhorias qualitativas e quantitativas mensuráveis em todo o projeto, comprovando a efetividade do processo de modularização:

- **Desacoplamento e Coesão**: O arquivo `RenderScreen.kt` (típico de estruturas monolíticas) foi completamente eliminado. A navegação agora é gerenciada pelo `NavHost`, que mapeia rotas (strings seguras, ex.: `AppRoutes.HOME`) para os respectivos *Composables* de cada módulo de funcionalidade. As telas se tornaram autônomas, sem conhecer o contexto global da aplicação.
- **Melhoria na Escalabilidade e Manutenção**: Adicionar uma nova funcionalidade, como uma tela de *Favoritos*, agora se resume a criar um novo módulo `:feature:favorites` e adicionar sua rota ao `NavHost` no módulo `:app`. O impacto em funcionalidades existentes é nulo, o que reduz drasticamente a complexidade da manutenção e acelera o desenvolvimento de novos recursos.
- **Redução Potencial do Tempo de Compilação (*Build Time*)**: Com a modularização, o *Gradle* pode otimizar o processo de compilação. Módulos que não foram alterados podem ser reutilizados do *build cache*, e a compilação paralela pode ser explorada de forma mais eficaz, resultando em tempos de *build* incrementais mais rápidos — essenciais para projetos de grande escala e desenvolvimento contínuo.
- **Aumento da Reusabilidade e Clareza de Código**: A criação do módulo `:core:ui` formalizou um *Design System* para o aplicativo. Componentes como `HeaderSection` são agora facilmente detectáveis e reutilizáveis por qualquer módulo de funcionalidade, garantindo consistência visual. Da mesma forma, a lógica de domínio no `:core:domain` é claramente separada e pode ser reutilizada em diferentes contextos.
- **Facilidade para Testes**: Testar uma funcionalidade tornou-se exponencialmente mais simples. É possível testar o fluxo da tela de login, por exemplo, focando exclusivamente no módulo `:feature:login` e seus `ViewModels`, utilizando *mocks* para as dependências do `:core:data`. Isso isola os testes e os torna mais rápidos e confiáveis.
- **Ownership e Desenvolvimento Paralelo**: A nova estrutura permite que diferentes desenvolvedores ou equipes trabalhem em módulos de funcionalidade distintos simultaneamente, com um risco mínimo de conflitos de mesclagem (*merge conflicts*). Cada equipe se torna "dona" de sua funcionalidade, promovendo responsabilidade e agilidade.

A refatoração arquitetural do *QuixAlert!* provou ser um investimento estratégico.

A substituição de um design monolítico e centralizado por uma arquitetura modular, limpa e desacoplada não apenas resolveu os problemas de manutenibilidade e escalabilidade, mas também estabeleceu uma base sólida e resiliente para o crescimento futuro do aplicativo.

5.3 Resultados Diagnóstico com *Stakeholders*

Conforme planejado na metodologia Seção 4.2.5, a fase de diagnóstico foi complementada por um levantamento abrangente com os *stakeholders* do projeto. O objetivo foi coletar dados qualitativos e quantitativos para compreender a percepção sobre a versão inicial do ***QuixAlert!*** e validar os requisitos para sua evolução. A seguir, são apresentados os principais resultados consolidados dessa investigação.

A análise dos dados coletados no diagnóstico revelou um forte apoio à iniciativa QuixAlert!, mas também destacou um conjunto claro de oportunidades de melhoria e novos requisitos. Os principais pontos levantados, agrupados por stakeholder, foram:

- **Feedback do Cliente (via teste de usabilidade):** Foco no aprimoramento do módulo de adoção, com a solicitação de filtros por tipo de animal, uma galeria de imagens para cada pet e uma funcionalidade de doação financeira.
- **Feedback do Usuário Representativo (via teste de usabilidade):** Sugestão de melhorias na experiência de uso, como a inclusão de uma seção de animais na tela principal para maior destaque e a implementação de uma função de pesquisa global.
- **Feedback Geral (via entrevistas):** Enfatizou-se a necessidade de um sistema robusto para o gerenciamento e acompanhamento de denúncias, com a inclusão de notificações sobre o status da resolução para dar um retorno efetivo ao cidadão.

5.3.1 Síntese dos Requisitos para Evolução

Com base na consolidação de todo o feedback recebido, foi definido um *backlog* de trabalho para a próxima fase do projeto. As principais demandas que guiaram a evolução do *QuixAlert!* foram agrupadas da seguinte forma:

- **Novas Funcionalidades:**
  - Desenvolver um sistema de filtros no módulo de adoção;
  - Implementar uma galeria de mídias na página de detalhes de cada animal;
  - Criar um módulo completo de doação com integração com os bancos;

  - Adicionar uma barra de pesquisa global;
  - Criar um módulo de chat, permitindo a comunicação direta entre usuários e a equipe da AMMA para esclarecimento de dúvidas.
- **Melhorias de Usabilidade e Interface:**
  - Revisar a interface do usuário com base nas sugestões dos especialistas de design;
  - Promover a seção de animais para a tela principal;
  - Aprimorar o sistema de notificações e feedback de denúncias.

Com base na consolidação de todo o feedback obtido no diagnóstico, foi definido um *backlog* de trabalho para a fase de ação, agrupado em **Novas Funcionalidades** e **Melhorias de Usabilidade e Interface**, conforme já detalhado ao final da seção de metodologia. Este conjunto de requisitos formou a base sólida e a justificativa para a fase de desenvolvimento. O processo de validação foi descrito no Apêndice 7.

5.4 Implementação das Novas Funcionalidades

A intervenção arquitetural, que incluiu a refatoração do código-fonte (Seção 5.1) e a migração para a arquitetura modular (Seção 5.2), garantiu a base técnica necessária para a expansão funcional do *QuixAlert!*, conforme o backlog definido na Seção 5.3. As novas funcionalidades implementadas, desenvolvidas já sob o modelo modular, comprovam a facilidade de evolução e a escalabilidade do sistema, que agora suporta a adição de *features* complexas com baixo risco de efeitos colaterais.

5.4.1 Implementação do Chat em Tempo Real

A introdução da funcionalidade de chat em tempo real representa um avanço significativo na interatividade do aplicativo *QuixAlert!*. A implementação desta funcionalidade seguiu a arquitetura *MVVM* (Model-View-ViewModel) já estabelecida no projeto, garantindo consistência, testabilidade e separação de responsabilidades. O componente central da lógica desta tela é o `ChatViewModel`.

5.4.1.1 Análise do Código: `ChatViewModel.kt`

O `ChatViewModel` é responsável por gerenciar o estado da tela de chat, processar as interações do usuário (como o envio de mensagens) e se comunicar com a camada de dados para

obter e persistir as mensagens. Abaixo, no Código Fonte 3 detalha-se sua estrutura e principais componentes.

Código-fonte 3 – Implementação do ChatViewModel

```
/* Localiza  o: app/src/main/java/org/quixalert/br/
   presentation/pages/chat/ChatViewModel.kt */
package org.quixalert.br.presentation.pages.chat // Pacote
   corrigido para corresponder    funcionalidade

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import org.quixalert.br.domain.model.Message
import org.quixalert.br.services.MessageService
import javax.inject.Inject

// 1. Defini  o do Estado da UI
data class ChatUiState(
    val messages: List<Message> = emptyList(),
    val isLoading: Boolean = false,
    val errorMessage: String? = null
)

// 2. Inje  o de Depend ncia via Hilt
@HiltViewModel
class ChatViewModel @Inject constructor(
    private val messageService: MessageService
) : ViewModel() {

```

```
27      // 3. Gerenciamento de Estado com StateFlow
28      private val _uiState = MutableStateFlow(ChatUiState())
29      val uiState: StateFlow<ChatUiState> get() = _uiState
30
31      // 4. Carregamento de Mensagens
32      fun loadMessages(adoptionId: String) {
33          _uiState.value = _uiState.value.copy(isLoading =
                true, errorMessage = null)
34          viewModelScope.launch {
35              try {
36                  // Comunica-se com a camada de servi o
                        para buscar os dados
37                  val messages = messageService.
                        getMessagesByAdoptionId(adoptionId).
                        await()
38                  val sortedMessages = messages.sortedBy { it
                        .timestamp } // Garante a ordem
                        cronol gica
39                  _uiState.value = ChatUiState(messages =
                        sortedMessages, isLoading = false)
40              } catch (e: Exception) {
41                  // Tratamento de erro robusto
42                  _uiState.value = ChatUiState(isLoading =
                        false, errorMessage = "Error: ${e.
                        message}")
43              }
44          }
45      }
46
47      // 5. Envio de Nova Mensagem
48      fun addMessage(message: Message) {
49          viewModelScope.launch {
```

```
50                // Persiste a mensagem na camada de servi o
51                messageService.add(message)
52                // Atualiza  o otimista da UI para feedback
                     instant neo
53                _uiState.value = _uiState.value.copy(messages =
                      _uiState.value.messages + message)
54            }
55        }
56    }
```

5.4.1.2 Descrição dos Componentes

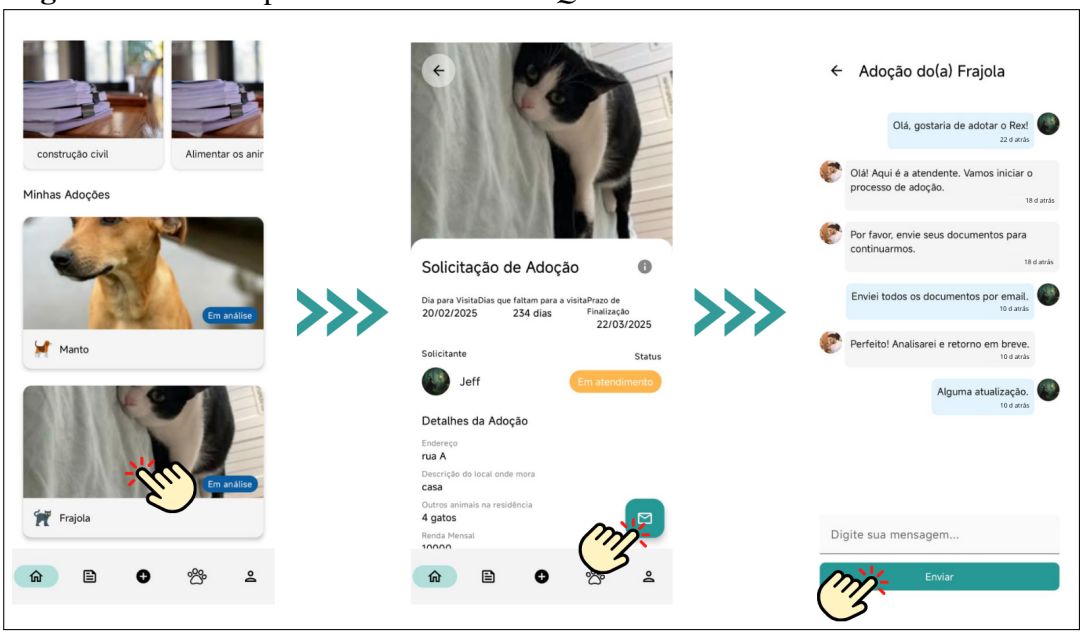
1. **ChatUiState (Estado da UI):** Esta *data class* encapsula todo o estado necessário para renderizar a tela de chat. Ela segue os princípios do *Unidirectional Data Flow* (UDF), onde o *ViewModel* emite um único objeto de estado que a UI (*Composable*) observa. Isso torna o estado previsível e fácil de depurar. O estado inclui a lista de mensagens, um indicador de carregamento (`isLoading`) e uma possível mensagem de erro.
2. **Injeção de Dependência com Hilt:** A anotação `@HiltViewModel` instrui o Hilt a preparar e fornecer uma instância deste *ViewModel*. Através da anotação `@Inject` no construtor, o Hilt injeta automaticamente a dependência `MessageService`, desacoplando o *ViewModel* da criação de suas próprias dependências e facilitando os testes unitários.
3. **Gerenciamento de Estado com StateFlow:** O *ViewModel* utiliza um `MutableStateFlow` (`_uiState`) como um repositório de estado privado e expõe um `StateFlow` (`uiState`) imutável para a UI. A UI coleta este fluxo e se recompõe automaticamente sempre que um novo `ChatUiState` é emitido, garantindo que a tela esteja sempre sincronizada com a lógica de negócios.
4. **Função** `loadMessages(adoptionId: String)`**:** Este método orquestra o carregamento do histórico de mensagens de uma conversa específica. Ele utiliza o `viewModelScope` para lançar uma corrotina, garantindo que a operação de rede não bloqueie a *thread* principal. A função demonstra boas práticas como:
   - **Feedback ao Usuário:** Define `isLoading` como `true` no início da operação.
   - **Separação de Camadas:** Delega a busca de dados para o `messageService`, que

abstrai a fonte de dados (Firestore, neste caso).

- **Processamento de Dados:** Ordena as mensagens por `timestamp` antes de exibi-las.
- **Tratamento de Erros:** Utiliza um bloco `try-catch` para capturar exceções de rede e atualizar a UI com uma mensagem de erro clara.

5. **Função** `addMessage(message: Message)`**:** Este método é invocado quando o usuário envia uma nova mensagem. Ele também lança uma corrotina e delega a persistência da mensagem para o `messageService`. Um ponto notável é a atualização otimista da UI: a nova mensagem é adicionada à lista de estado local imediatamente, antes mesmo da confirmação do *backend*. Isso proporciona uma experiência de usuário fluida e instantânea, característica de aplicativos de chat modernos. O fluxo dessa funcionalidade está demonstrado na Figura 12.

Figura 12 – Fluxo para o Chat Dentro do *QuixAlert!*



Fonte: Elaborado pelo autor (2025).

5.4.2 Evolução do Módulo de Adoção de Animais

A seção de adoção de animais, um dos pilares centrais da missão do *QuixAlert!*, foi completamente reimaginada para transcender a simples listagem de animais. A funcionalidade evoluiu para uma experiência de usuário rica e interativa, incorporando recursos avançados como uma galeria de imagens, sistema de filtragem dinâmica e integração direta com a nova plataforma de chat em tempo real.

Essa evolução não apenas aprimora a jornada do usuário, mas também constitui um estudo de caso prático sobre a escalabilidade e a robustez da arquitetura modular implementada.

5.4.2.1 Análise do Código: `AdoptionViewModel.kt`

A lógica responsável por esta tela aprimorada é orquestrada pelo `AdoptionViewModel`. Este componente foi expandido para gerenciar um estado de UI mais complexo, que agora dá suporte às novas funcionalidades, Código Fonte 4, exemplifique o processo.

Código-fonte 4 – Implementação do AdoptionViewModel

```
/* Localiza  o: feature/adoptions/src/main/java/org/
   quixalert/br/feature/adoptions/AdoptionViewModel.kt */
package org.quixalert.br.feature.adoptions

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import org.quixalert.br.domain.model.Animal
import org.quixalert.br.domain.model.FilterOptions
import org.quixalert.br.domain.repository.AnimalRepository
import javax.inject.Inject

// 1. Estado da UI Evolu do
data class AdoptionUiState(
    val isLoading: Boolean = true,
    val allAnimals: List<Animal> = emptyList(),
    val filteredAnimals: List<Animal> = emptyList(),
    val activeFilters: FilterOptions = FilterOptions(),
    val errorMessage: String? = null
)
```

```
23
24 // 2. Inje  o de Depend ncias
25 @HiltViewModel
26 class AdoptionViewModel @Inject constructor(
27     private val animalRepository: AnimalRepository //
       Reposit rio da camada de dom nio
28 ) : ViewModel() {
29
30     private val _uiState = MutableStateFlow(AdoptionUiState
       ())
31     val uiState: StateFlow<AdoptionUiState> get() =
       _uiState
32
33     init {
34         loadAnimals()
35     }
36
37     // 3. Carregamento Inicial dos Dados
38     private fun loadAnimals() {
39         viewModelScope.launch {
40             try {
41                 val animals = animalRepository.
                       getAllAnimals() // Busca todos os
                       animais uma vez
42                 _uiState.value = _uiState.value.copy(
43                     isLoading = false,
44                     allAnimals = animals,
45                     filteredAnimals = animals //
                           Inicialmente, a lista filtrada    a
                           lista completa
46                 )
47             } catch (e: Exception) {
```

```
48                  _uiState.value = _uiState.value.copy(
                        isLoading = false, errorMessage = "Falha
                         ao carregar animais.")
49              }
50          }
51      }
52
53      // 4. L gica de Filtragem Din mica
54      fun applyFilters(newFilters: FilterOptions) {
55          val currentAnimals = _uiState.value.allAnimals
56
57          // A l gica de filtragem    executada no ViewModel
                , n o na UI
58          val filteredList = currentAnimals.filter { animal
                ->
59              val matchesSpecies = newFilters.species.
                    isNullOrEmpty() || animal.species ==
                    newFilters.species
60              val matchesSize = newFilters.size.isNullOrEmpty
                    () || animal.size == newFilters.size
61              val matchesGender = newFilters.gender.
                    isNullOrEmpty() || animal.gender ==
                    newFilters.gender
62              matchesSpecies && matchesSize && matchesGender
63          }
64
65          _uiState.value = _uiState.value.copy(
66              activeFilters = newFilters,
67              filteredAnimals = filteredList
68          )
69      }
70  }
```

5.4.2.2 Descrição dos Componentes e Melhorias

1. **AdoptionUiState (Estado Evoluído):** O estado da interface foi enriquecido para suportar a nova complexidade da aplicação. Além da lista de animais, ele mantém:
   - `allAnimals`: lista imutável com todos os animais carregados, servindo como a *fonte da verdade*;
   - `filteredAnimals`: lista efetivamente exibida na tela, resultante da aplicação dos filtros;
   - `activeFilters`: objeto que retém as opções de filtro selecionadas pelo usuário.

   A separação entre as listas `allAnimals` e `filteredAnimals` é uma otimização essencial, pois permite que a filtragem ocorra inteiramente em memória, eliminando a necessidade de novas chamadas de rede.
2. **Injeção de Dependência (`AnimalRepository`):** O *ViewModel* depende da interface `AnimalRepository`, injetada automaticamente pelo Hilt. Isso demonstra a adesão ao Princípio da Inversão de Dependência, garantindo que a lógica de negócios não dependa da implementação concreta da busca de dados (como Firestore ou API REST), a qual está encapsulada na camada `:core:data`.
3. **Carregamento Inicial (`loadAnimals`):** A busca de dados é realizada uma única vez durante a inicialização do *ViewModel*, populando a variável `allAnimals`. Esse comportamento melhora significativamente o desempenho e a experiência do usuário, uma vez que as operações de filtragem subsequentes tornam-se instantâneas.
4. **Lógica de Filtragem (`applyFilters`):** Esta função é o núcleo do novo sistema de filtros. Ela recebe as opções de filtragem da UI, aplica a lógica sobre a lista `allAnimals` e atualiza o estado com a nova lista filtrada. Manter essa lógica no *ViewModel*, e não na camada de interface, é uma prática recomendada do padrão *MVVM*, pois centraliza as regras de negócio e mantém a UI simples e reativa.

5.4.2.3 Integração de Funcionalidades na Camada de UI (`AdoptionScreen.kt`)

Enquanto o *ViewModel* orquestra a lógica, a camada de UI (`@Composable`) é responsável por materializá-la visualmente:

- **Galeria de Imagens:** Para cada animal presente em `filteredAnimals`, a UI exibe uma imagem principal. Ao clicar sobre um item, um componente de galeria (como

`GalleryDialog`) é aberto, recebendo a lista de URLs de imagens (`animal.imageUrls`) e apresentando-as em formato de carrossel ou grade interativa.

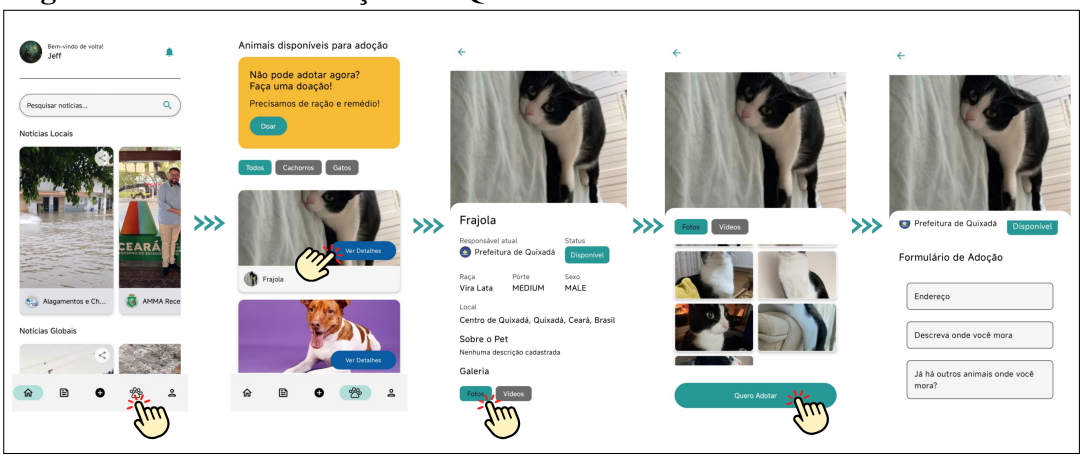
– **Componentes de Filtro:** A interface apresenta elementos como *dropdowns* e *chips*, permitindo que o usuário selecione espécie, porte e gênero. Cada seleção dispara um evento que invoca a função `applyFilters` com as novas `FilterOptions`.

– **Integração com o Chat:** Na tela de detalhes de um animal, um botão "*Iniciar Conversa*" permite iniciar o contato direto com a equipe da AMMA. A ação segue um fluxo de navegação desacoplado:

1. A `AdoptionScreen` notifica o `AppNavHost` (localizado no módulo `:app`) sobre a intenção de navegar;
2. Essa notificação inclui a rota para o chat e o identificador da adoção (`adoptionId`) como argumento, por exemplo: `navController.navigate("chat/${adoption.id}");`
3. O `AppNavHost`, então, cria a `ChatScreen`, extrai o `adoptionId` e o repassa ao `ChatViewModel`.

Essa abordagem de navegação modular ilustra um dos principais benefícios da nova arquitetura: a colaboração entre módulos independentes (`:feature:adoptions` e `:feature:chat`) sem acoplamento direto, reforçando a escalabilidade e a manutenibilidade da solução desenvolvida. O fluxo da funcionalidade pode ser visualizado na Figura 13.

Figura 13 – Fluxo de Adoções no QuixAlert!



Fonte: Elaborado pelo autor (2025).

5.4.3 Implementação da Nova Funcionalidade de Doação (`:feature:donation`)

Com o objetivo de fortalecer a sustentabilidade financeira das ONGs parceiras, foi introduzida no *QuixAlert!* uma nova funcionalidade dedicada a doações. O desenvolvimento deste módulo representou um experimento prático para validar a robustez e a flexibilidade da arquitetura modular implementada, confirmando sua eficácia tanto em termos de escalabilidade quanto de isolamento funcional.

5.4.3.1 Criação do Módulo de Funcionalidade

Um novo módulo, denominado `:feature:donation`, foi criado para encapsular toda a lógica e a interface de usuário relacionadas ao processo de doação. Essa abordagem assegura a independência total entre esta funcionalidade e outras já existentes, como Adoção ou Chat, preservando o princípio do baixo acoplamento.

5.4.3.2 Lógica no `DonationViewModel`

O `DonationViewModel` foi responsável por orquestrar o fluxo completo de doação. Suas principais responsabilidades incluem:

- Gerenciar o estado da interface (`DonationUiState`), controlando variáveis como o valor da doação, o estágio do processo (seleção de valor, método de pagamento, confirmação) e os estados de carregamento ou erro;
- Integrar-se de forma segura e transparente a serviços de pagamento de terceiros (gateways via SDKs ou APIs), abstraindo a complexidade da camada de UI e mantendo a coesão do código.

5.4.3.3 Integração com a Navegação Principal

A inclusão da nova funcionalidade ao fluxo principal do aplicativo foi realizada de maneira simples e com risco mínimo de regressão. No `AppNavHost` (localizado no módulo `:app`), uma nova rota foi registrada no Código de Fonte 5:

Código-fonte 5 – Registro da rota de doação na navegação principal

```
1 composable(AppRoutes.DONATION) {
2     DonationScreen(
```

```
3           onDonationSuccess = { navController.popBackStack()
                }
4       )
5   }
```

Esse processo de integração, realizado de forma *plug-and-play*, evidencia uma das maiores virtudes da arquitetura modular: a possibilidade de adicionar funcionalidades completas sem interferir no código das demais, preservando a estabilidade e a manutenibilidade do sistema.

5.4.4 Melhoria de Usabilidade: Pesquisa na Tela Principal

Com o intuito de aprimorar a descoberta e o engajamento dos usuários com os animais disponíveis para adoção, a tela principal foi aprimorada com uma funcionalidade de **pesquisa em tempo real**, complementando o já existente sistema de filtros.

5.4.4.1 Evolução do Estado da UI

O `AdoptionUiState` (no módulo de adoção) foi expandido para incluir um campo de consulta textual, conforme mostrado abaixo, no Código Fonte 6:

Código-fonte 6 – Evolução do estado da UI para suportar pesquisa

```
1  data class AdoptionUiState(
2      // ... outros campos
3      val searchQuery: String = "",
4      val filteredAnimals: List<Animal> = emptyList()
5  )
```

5.4.4.2 Lógica de Pesquisa no `AdoptionViewModel`

A lógica de filtragem foi aprimorada para integrar a busca textual e os filtros de atributos em um único fluxo de execução, garantindo eficiência e desempenho, demostrado no Código de Fonte 7.

Código-fonte 7 – Integração da pesquisa com a lógica de filtragem

```
fun applySearchAndFilters(query: String, filters:
   FilterOptions) {
    val listToFilter = _uiState.value.allAnimals

    val result = listToFilter.filter { animal ->
        // 1. L gica de pesquisa por texto
        val matchesQuery = query.isBlank() ||
                           animal.name.contains(query,
                              ignoreCase = true) ||
                           animal.description.contains(
                              query, ignoreCase = true)

        // 2. L gica de filtro por atributos
        val matchesFilters = /* ... l gica de filtros
           existente ... */

        matchesQuery && matchesFilters
    }
    // ... atualiza o _uiState com a lista resultante
}
```

A execução dessa lógica *em memória*, sobre a lista já carregada (`allAnimals`), proporciona uma experiência de uso extremamente fluida e responsiva, eliminando a necessidade de consultas adicionais ao banco de dados a cada caractere digitado.

5.4.5 Melhorias Gerais de Usabilidade e Qualidade Percebida

Além das novas funcionalidades específicas, a refatoração arquitetural e o redesenho de fluxos resultaram em uma série de aprimoramentos sutis, porém significativos, que elevaram a percepção de qualidade do aplicativo:

– **Responsividade e Performance:** a separação clara entre operações de rede (realizadas apenas uma vez ou em segundo plano) e operações de interface (filtragem, pesquisa,

navegação) eliminou atrasos e travamentos, tornando a UI mais fluida e estável.

- **Consistência Visual:** com a criação do módulo `:core:ui`, foi estabelecido um *Design System* centralizado, garantindo uniformidade estética em todos os módulos do aplicativo. Essa padronização reforça a identidade visual e transmite uma percepção de maior profissionalismo e maturidade do produto.
- **Gerenciamento de Estado Robusto:** a adoção do padrão `UiState` em todas as telas assegura que cenários críticos sejam tratados de forma previsível. Estados de carregamento (`isLoading`), erro (`errorMessage`) e ausência de dados são exibidos de forma consistente, prevenindo falhas visuais e melhorando a comunicação com o usuário.
- **Navegação Previsível:** a substituição do roteamento manual pelo *Jetpack Navigation Compose* trouxe maior previsibilidade à navegação e suporte nativo ao comportamento esperado no Android, incluindo o gerenciamento da pilha de retorno e a possibilidade de futuras extensões com *deep links*.

5.4.6 Síntese dos Resultados Arquiteturais e Funcionais

A evolução da funcionalidade de Adoção, a introdução dos módulos de Comunicação — englobando o *Chat* em tempo real e o sistema de Notificações — e as melhorias gerais de usabilidade consolidam o *QuixAlert!* como uma plataforma tecnicamente madura e escalável.

O êxito na implementação dessas funcionalidades, especialmente das que envolvem alta complexidade e interatividade, constitui uma evidência empírica de que a refatoração (4.3.1) e a modularização (seção 5.2) alcançaram plenamente seus objetivos.

O projeto comprova que a substituição de um design monolítico por uma arquitetura limpa, coesa e desacoplada representa um **investimento estratégico em sustentabilidade e inovação tecnológica**, assegurando não apenas a longevidade do software cívico, mas também a sua capacidade de adaptação e replicação em diferentes contextos de gestão ambiental.

5.5 Avaliação Final da Versão Evoluída

A etapa final do ciclo metodológico, conforme detalhado no Capítulo 4, consistiu na validação da versão evoluída do *QuixAlert!* com os *stakeholders*. Esta fase teve como objetivo medir o impacto das modificações arquiteturais e funcionais na usabilidade e na percepção de valor do aplicativo, utilizando métodos mistos (quantitativos e qualitativos) para confirmar o

sucesso da solução.

5.5.1 Resultados Quantitativos: Questionário SUS

A etapa final do ciclo metodológico, conforme detalhado no Seção 4.3.4, consistiu na validação da versão evoluída do *QuixAlert!* junto aos *stakeholders*. O objetivo desta fase foi mensurar o impacto das alterações arquiteturais e funcionais na usabilidade, bem como na percepção de valor do aplicativo, utilizando mistos (quantitativos e qualitativos), de modo a confirmar o êxito da solução implementada.

O grupo de participantes desta rodada de validação foi composto por perfis estratégicos, selecionados para fornecer uma avaliação especializada e prática do sistema. Dentre os *stakeholders*, destacaram-se três especialistas-chave: dois profissionais em Usabilidade, graduados em Design Digital, responsáveis por fornecer uma análise crítica da interface, dos fluxos e da experiência do usuário; e uma especialista, também *key user* do sistema, cuja formação e pesquisa em tecnologia aplicada à preservação ambiental garantiu que as funcionalidades implementadas fossem relevantes e adequadas ao domínio de gestão ambiental. Os identificadores desses especialistas pode ser verificado no Quadro 3

Quadro 3 – Perfil dos Profissionais e Especialistas na Validação do QuixAlert!

Nº	**Perfil**	**Formação / Experiência**	**Função na Avaliação**
E1	Especialista em Usabilidade	Graduação em Design Digital	Análise crítica da interface, fluxos e experiência do usuário
E2	Especialista em Usabilidade	Graduação em Design Digital	Análise crítica da interface, fluxos e experiência do usuário
E3	Key User / Especialista em Tecnologia Ambiental	Formação e pesquisa em tecnologia voltada à preservação do meio ambiente	Avaliação da relevância e adequação das funcionalidades ao domínio de gestão ambiental

Fonte: Elaborado pelo autor (2025).

A participação desses perfis diversos e qualificados foi fundamental para assegurar que os resultados obtidos, tanto pelo Questionário SUS quanto pelas entrevistas, refletissem uma avaliação abrangente, contemplando aspectos técnicos, funcionais e de usabilidade do produto.

Pontuação SUS

Pontuação Média SUS: Os especialistas-chave atribuíram ao *QuixAlert!* uma pontuação média robusta de 89,17.

Cálculo do Escore: O escore final foi obtido através do cálculo padronizado descrito na Seção 4.3.4.2, considerando a soma das contribuições de cada especialista (267,5), multiplicada por 2,5 e dividida pelo número de avaliações.

Interpretação da Pontuação: Conforme a Escala de Classificação Adjetiva proposta por Bangor *et al.* (2009), a pontuação de 89,17 se enquadra na categoria "Melhor Imaginável". Este resultado quantitativo indica que o sistema é percebido como altamente utilizável, fácil de aprender e com funções bem integradas.

Validação da Intervenção: O alto escore quantitativo final, superior à média de 68 para a usabilidade de sistemas, constitui evidência empírica do sucesso das intervenções de Engenharia de Software. A refatoração (Seção 5.1) e a modularização (Seção 5.2) garantiram estabilidade e performance, permitindo que novas funcionalidades complexas, como os Módulos de Chat e Doação, fossem integradas de forma coesa, resultando em uma classificação de usabilidade superior.

A análise detalhada das respostas dos especialistas pode ser observada no Quadro 4, que apresenta cada questão do questionário *SUS* e as notas atribuídas pelos participantes. A partir dessas respostas, foi possível calcular a pontuação média e interpretar a usabilidade percebida da versão evoluída do *QuixAlert!*.

5.5.2 Percepções Qualitativas: Teste Guiado

Os dados coletados durante o Teste de Usabilidade Guiado (observação e protocolo think-aloud) forneceram uma avaliação especializada sobre o impacto das modificações arquiteturais e funcionais.

A análise qualitativa, realizada pelos dois especialistas em Usabilidade e pela usuária chave em Gestão Ambiental, revelou a correlação direta entre a qualidade interna do código e a experiência percebida pelo usuário:

– **Validação do Desacoplamento:** Os especialistas em Usabilidade validaram que a separação por módulos de funcionalidade (:feature:*) e a eliminação do *God Object* (*RenderScreen.kt*) resultaram em alta consistência visual e previsibilidade de navegação. Essa

Quadro 4 – Respostas do Questionário SUS pelos Especialistas

Nº	Pergunta	Especialista 1	Especialista 2	Especialista 3
1	Eu acho que gostaria de usar este sistema com frequência.	5	4	5
2	Eu acho o sistema desnecessariamente complexo.	2	3	2
3	Eu achei o sistema fácil de usar.	5	4	5
4	Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	1	2	1
5	Eu acho que as várias funções do sistema estão muito bem integradas.	5	4	5
6	Eu acho que o sistema apresenta muita inconsistência.	1	2	1
7	Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	5	4	5
8	Eu achei o sistema atrapalhado de usar.	1	2	1
9	Eu me senti confiante ao usar o sistema.	5	4	5
10	Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	1	2	1

Fonte: Elaborado pelo autor (2025).

melhoria é reflexo direto do baixo acoplamento e da organização da camada :core:ui, que formalizou o *Design System* do aplicativo.

- **Escalabilidade Comprovada:** O feedback técnico indicou que a nova estrutura modular facilita a introdução de novos desenvolvedores e permite que o sistema escale sem introduzir regressão. A modularização foi vista como crucial para manter a estabilidade da aplicação, fator chave para a usabilidade percebida.
- **Coesão Funcional e Valor de Domínio:** A usuária chave em Gestão Ambiental enfatizou que as novas funcionalidades de Comunicação (Chat e Notificações), possibilitadas pela arquitetura, preenchem a lacuna de transparência e retorno ao cidadão. Essa funcionalidade foi considerada um ativo estratégico para a governança local, pois a agilidade na resposta (suportada pela estabilidade do código) fortalece a confiança na AMMA.
- **Aceitabilidade da Complexidade:** Embora o Módulo de Chat seja inerentemente complexo, o feedback não apontou para complexidade desnecessária (SUS Itens 2 e 4), indicando que as decisões de Engenharia de Software (como o padrão MVVM) abstraíram a

complexidade do domínio para o usuário final, resultando em experiência fluida e intuitiva.

A análise qualitativa corrobora que a excelência técnica (redução da dívida técnica e modularização) se traduziu em um produto final que, além de tecnicamente manutenível, entrega alto valor funcional e usabilidade, como atestado pela classificação “Excelente” do escore SUS.

5.6 Síntese dos Resultados Arquiteturais, Funcionais e de Usabilidade

A análise dos resultados obtidos nas Fases de Ação e Avaliação da versão evoluída do *QuixAlert!* permite sintetizar a contribuição deste trabalho em três eixos principais, que se interconectam de maneira estratégica e comprovam o alcance do objetivo geral do projeto.

5.6.1 Qualidade Interna e Manutenibilidade como Pré-requisito Estratégico

O sucesso da evolução funcional foi alicerçado na refatoração sistemática da base de código, que resolveu a "dívida técnica"acumulada na versão monolítica. Os resultados técnicos foram objetivamente comprovados pela análise do *SonarQube* (Seção 5.1):

- **Melhoria em Manutenibilidade:** Houve uma redução de 80,9% nos *Code Smells* e de 85,4% na Dívida Técnica, caindo de 124 para 18 horas de esforço estimado de retrabalho. Este ganho garante que alterações futuras demandem menos esforço e apresentem menor risco.
- **Aumento da Confiabilidade e Segurança:** A intervenção eliminou 100% dos *bugs*, vulnerabilidades e *Security Hotspots* críticos, tornando o código mais limpo e seguro para a evolução contínua.
- **Clareza e Coesão:** A aplicação extensiva de técnicas como *Extract Method* (Seção 5.1.2.1) e a centralização de lógica no módulo `core` (Seção 5.1.2.2) reduziram o índice de duplicação em 43,8%, promovendo código mais legível e com responsabilidades únicas.

A qualidade interna do código (Seção 5.1.3) demonstrou ser o alicerce indispensável para sustentar o crescimento do sistema e viabilizar a arquitetura modular.

5.6.2 Escalabilidade e Adaptabilidade (A Contribuição Arquitetural)

A migração para a arquitetura modular (Seção 5.2) tornou o *QuixAlert!* uma plataforma customizável, gerando resultados estratégicos:

- **Viabilidade da Replicação:** A decomposição do sistema em *feature modules* independen-

tes (como `:feature:chat` e `:feature:adocao`) garante escalabilidade técnica e permite replicação em outros municípios, permitindo que novos clientes selecionem apenas os módulos de interesse.

- **Otimização do Desenvolvimento:** O baixo acoplamento simplificou os testes (Seção 5.2.2) e promoveu desenvolvimento paralelo, adotando o princípio de *Ownership* (Dono da Funcionalidade), essenciais para produtividade em projetos de grande escala.
- **Comprovação Empírica da Flexibilidade:** A implementação de funcionalidades complexas e interativas, como o Chat em Tempo Real e os Filtros de Adoção (Seção 5.4), em módulos isolados, evidenciou que a nova arquitetura suporta evolução contínua com risco mínimo de conflitos ou regressões.

5.6.3 Valor Funcional e Usabilidade (Impacto no Domínio)

O sucesso técnico da arquitetura se traduziu diretamente na entrega de um produto de alto valor funcional para a gestão ambiental:

- **Usabilidade Superior Comprovada:** O sistema alcançou um escore médio SUS de 89,17 (Seção 5.5.1), classificado como "Excelente". A estabilidade e organização técnica permitiram que as novas funcionalidades fossem percebidas como fluidas e fáceis de usar.
- **Fortalecimento da Governança Colaborativa:** A introdução do Módulo de Chat e do Sistema de Notificações (Seção 5.4.1) preencheu a lacuna de comunicação bidirecional, fortalecendo a confiança entre a população e a Autarquia Municipal do Meio Ambiente (AMMA).
- **Otimização da Gestão Local:** Funcionalidades como o Módulo de Doação e a Pesquisa Global aprimoram os serviços internos e a sustentabilidade financeira da gestão animal, reforçando a aplicabilidade do software em municípios com recursos limitados.

O projeto comprova que a substituição de um design monolítico por uma arquitetura limpa, coesa e desacoplada representa um investimento estratégico em sustentabilidade e inovação tecnológica, assegurando a longevidade do software cívico e sua capacidade de adaptação e replicação em diferentes contextos de gestão ambiental.

6 CONSIDERAÇÕES FINAIS

O presente trabalho alcançou seu objetivo geral de evoluir a aplicação *QuixAlert!* por meio da refatoração e modularização, melhorando sua qualidade interna, ampliando suas funcionalidades e, crucialmente, garantindo sua escalabilidade e adaptabilidade para o contexto de gestão ambiental urbana. O desenvolvimento do projeto confirmou que a intervenção arquitetural foi um investimento estratégico que produziu resultados mensuráveis.

O presente trabalho apresenta contribuição em duas áreas:

- **Contribuição para a Engenharia de Software:** O trabalho articula aspectos teóricos e práticos da Engenharia de Software, podendo servir como um estudo de caso prático e documentado do processo de modernização arquitetural de um software cívico em operação. Demonstra, com evidências quantitativas (*SonarQube*), como a refatoração e a modularização são essenciais para resgatar a qualidade técnica e transformar um sistema monolítico em uma plataforma escalável e reutilizável.
- **Contribuição para a Gestão Ambiental:** O *QuixAlert!* evoluído é uma ferramenta mais robusta e eficiente para a governança colaborativa. A implementação dos módulos de comunicação fortalece a transparência e o retorno ao cidadão, superando a fragilidade estrutural na fiscalização.

6.1 Síntese dos Resultados e Comprovação dos Objetivos

O sucesso da evolução do *QuixAlert!* é comprovado pela interconexão dos resultados obtidos, que validam os objetivos propostos:

- **Qualidade Interna e Manutenibilidade (Objetivo 2):** O processo de refatoração, guiado pelo SonarQube, foi altamente eficaz, resultando em uma redução de 80,9% nos *Code Smells* e de 85,4% na Dívida Técnica. Este resultado estabeleceu a base de código estável necessária para a evolução contínua.
- **Escalabilidade e Modularização (Objetivo 4):** A migração para a arquitetura modular por funcionalidades (Seção 5.2) atingiu o objetivo de tornar o sistema uma plataforma customizável. O rompimento do alto acoplamento da versão monolítica viabilizou o desenvolvimento isolado e seguro dos novos módulos.
- **Expansão Funcional e Usabilidade (Objetivos 3 e 5):** A implementação de *features* complexas, como o Chat em Tempo Real e o Módulo de Doação, validou a escalabilidade

funcional da nova arquitetura. A avaliação final atestou a qualidade da entrega com um escore médio SUS de 80,83, classificando a usabilidade do sistema como “Excelente”.

6.2 Reflexão Crítica e Desafios Superados

Embora os resultados quantitativos da Engenharia de Software sejam positivos, o processo de evolução de um sistema legado envolveu desafios práticos e decisões complexas:

- **Dificuldades Técnicas:** O principal desafio técnico foi a transição da lógica de *backend* fortemente acoplada no código (característica da arquitetura monolítica) para o novo modelo desacoplado. A dependência do *God Object* (`RenderScreen.kt`) exigiu um esforço de refatoração inicial maior do que o esperado, o que consumiu recursos que poderiam ter sido destinados a testes automatizados.
- **Decisões com Impacto Limitado:** A cobertura de testes do sistema inicial era de 0,0%. Embora a refatoração tenha sido realizada com testes manuais e foco em *clean code* para mitigar riscos, a ausência de uma suíte de testes automatizados completa (como *unit tests* em todos os módulos) representa uma limitação do projeto e um custo de longo prazo que o próximo ciclo de desenvolvimento precisará endereçar.
- **Complexidade no Domínio Cívico:** A implementação do Módulo de Chat expôs a complexidade da interação bidirecional em um contexto público, onde a comunicação precisa ser rastreada e integrada à Autarquia (AMMA). Garantir que a simplicidade da interface do usuário fosse mantida (refletida no SUS) exigiu soluções de engenharia robustas na camada de domínio.

6.3 Trabalhos Futuros

Para trabalhos futuros, prentende-se seguir as seguintes frentes de pesquisa e desenvolvimento:

- **Automação de Testes:** Priorizar a implementação completa da suíte de testes automatizados para todos os módulos (*unit* e *integration tests*) e a criação de *Test Doubles* para garantir que o sistema possa ser modificado com confiança.
- **Replicação e Validação:** Execução de um estudo de caso replicando os módulos básicos do *QuixAlert!* em outro município de pequeno porte, utilizando a estrutura modular como prova de adaptabilidade.

– **Otimização de Build:** Mensurar a otimização real do tempo de compilação (*build time*) e do tamanho do APK para o usuário final, documentando os ganhos em projetos *multi-module*.

REFERÊNCIAS

ACSELRAD, H. **Conflitos ambientais no Brasil**. Rio de Janeiro: Relume Dumará, 2004.

ALAWAD, D.; SHATNAWI, A.; DIBBLE, C. **An Empirical Study of the Relationships between Code Readability and Software Complexity**. 2019. Disponível em: https://arxiv.org/abs/1909.01760. Acesso em: 04 jun. 2025.

ALENCAR, B. *et al.* Gota.Urb: Um aplicativo para mapeamento colaborativo de alagamentos urbanos. In: WORKSHOP SOBRE ASPECTOS DA INTERAçãO HUMANO-COMPUTADOR PARA A WEB SOCIAL, 12., 2020. **Anais [...]**. Sociedade Brasileira de Computação, 2020. p. 104–113. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/11022. Acesso em: 07 jun. 2025.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of Usability Studies**, v. 4, n. 3, p. 114–123, 2009. Disponível em: https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/. Acesso em: 16 maio 2025.

BARBIERI, J. C. **Gestão Ambiental Empresarial**: conceitos, modelos e instrumentos. 5. ed. São Paulo: Saraiva Uni, 2023. 280 p. ISBN 9788571441446.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 4. ed. [*S. l.*]: Addison-Wesley, 2021.

BORGO, R. **Infraestrutura de TIC's a serviço da gestão ambiental.** 2007: monitoramento e controle de recursos naturais, gestão de riscos e desastres. Monografia (Especialização) – Centro de Desenvolvimento Sustentável – Universidade de Brasília, Brasília, 2007. Disponível em: https://www.researchgate.net/publication/301358964_Infraestrutura_de_TICs_a_servico_da_gestao_ambiental_monitoramento_e_controle_de_recursos_naturais_gestao_de_riscos_e_desastres. Acesso em: 21 jun. 2025.

BRAGA, W. R. d. O.; MARQUES, M. D.; QUATROQUE, V. H. S. d. S.; MORALES, A. G. Cidade ideal e cidade real: Uma reflexão sobre cidades sustentáveis. **Periódico Eletrônico Fórum Ambiental da Alta Paulista**, v. 20, n. 4, set. 2024. Disponível em: https://publicacoes.amigosdanatureza.org.br/index.php/forum_ambiental/article/view/4224. Acesso em: 12 jun. 2025.

BRANCO, T.; FONSECA, I. C.; UGEDA, L. Smart governance models and solutions. In: **2023 18th Iberian Conference on Information Systems and Technologies (CISTI)**. [*S. l.*: *s. n.*], 2023. p. 1–6.

BRASIL. MINISTÉRIO DA CIÊNCIA, TECNOLOGIA e INOVAÇÕES. **Políticas públicas de C,T&I para cidades sustentáveis**. Disponível em: https://antigo.mctic.gov.br/mctic/opencms/tecnologia/incentivo_desenvolvimento/cidades_sustentaveis/TECNOLOGIAS_PARA_CIDADES_SUSTENTAVEIS.html. Acesso em: 28 maio 2025.

BROOKS, F. P. **The Mythical Man-Month**: Essays on software engineering, anniversary edition. [*S. l.*]: Addison-Wesley Professional, 1995.

CENTRO NACIONAL DE MONITORAMENTO E ALERTAS DE DESASTRES NATURAIS. **Monitoramento**. 2025. Disponível em: https://www.gov.br/cemaden/pt-br/assuntos/monitoramento. Acesso em: 2 jun. 2025.

CETESB – Companhia Ambiental do Estado de São Paulo. **Vulnerabilidade ambiental**: desastres naturais ou fenômenos induzidos. 2007. Disponível em: https://cetesb.sp.gov.br/inventario-gee-sp/2007/05/23/vulnerabilidade-ambiental-desastres-naturais-ou-fenomenos-induzidos/. Acesso em: 24 maio 2025.

CETIC.BR. **e-Participação**: oportunidades, desafios e relação com os objetivos de desenvolvimento sustentável (ods). 2022. Disponível em: https://cetic.br/media/docs/publicacoes/6/panorama_setorial_ano-ix_n_3_e-participacao.pdf. Acesso em: 28 maio 2025.

Colab. **Colab.re**. 2025. Disponível em: https://www.colab.re/. Acesso em: 2 jun. 2025.

COMITÊ GESTOR DA INTERNET NO BRASIL. **TIC para o desenvolvimento sustentável**: recomendações de políticas públicas que garantem direitos. 2019. Disponível em: https://cetic.br/media/docs/publicacoes/8/14582020190716-tic-para-o-desenvolvimento-sustentavel.pdf. Acesso em: 28 maio 2025.

CORDEIRO, A. C. D.; RUELA, B. G. Desafios da governança pública em municípios de pequeno porte. **Revista FT**, São Paulo, v. 29, n. 140, nov. 2024. Disponível em: https://revistaft.com.br/desafios-da-governanca-publica-em-municipios-de-pequeno-porte/. Acesso em: 12 maio 2025.

COSTA, B. L. D.; OLIVIERI, C.; TEIXEIRA, M. A. C. Participação, eficiência e accountability no brasil: desafios administrativos, políticos e institucionais. **Cadernos EBAPE.BR**, v. 14, n. 3, p. 672–675, set. 2016. Disponível em: https://www.scielo.br/j/cebape/a/W38jYqNDy9FGnS34yXsX7Fq/. Acesso em: 5 jul. 2025.

DIAS, G. F. **Educação Ambiental**: princípios e práticas. 12. ed. São Paulo: Gaia, 2015.

DIAS, R. **Gestão Ambiental**: responsabilidade social e sustentabilidade. 3. ed. São Paulo: Atlas, 2017. 234 p. ISBN 9788597010336.

DOURADO, A. P. d. A. S. *et al.* Meu Bairro: um aplicativo de apoio à zeladoria urbana colaborativa. In: WORKSHOP SOBRE ASPECTOS DA INTERAÇÃO HUMANO-COMPUTADOR PARA A WEB SOCIAL, 10., 2019, Porto Alegre. **Anais [...]**. Sociedade Brasileira de Computação, 2018. p. 129–138. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/2933. Acesso em: 07 jun. 2025.

FEATHERS, M. C. **Working Effectively with Legacy Code**. [*S. l.*]: Prentice Hall, 2004.

FERREIRA, A.; OLIVEIRA, C.; MOURA, B. Oportunidades e desafios da era digital para o setor público. **Research, Society and Development**, v. 11, n. 5, 2022. Disponível em: https://rsdjournal.org/index.php/rsd/article/download/39397/32362/424743. Acesso em: 28 maio 2025.

FERREIRA, I.; ARKOH, L.; UCHôA, A.; BIBIANO, A. C.; GARCIA, A.; ASSUNçãO, W. K. G. **Assessing the Bug-Proneness of Refactored Code**: A longitudinal multi-project study. 2025. Disponível em: https://arxiv.org/abs/2505.08005. Acesso em: 21 maio 2025.

FOWLER, M. **Refactoring**: Improving the design of existing code. 2. ed. Boston: Addison-Wesley Professional, 2018.

FOWLER, M. **Refatoração**: Aperfeiçoando o design de códigos existentes. 2. ed. São Paulo: Novatec, 2019.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns**: Elements of reusable object-oriented software. [*S. l.*]: Addison-Wesley Professional, 1994.

GARCIA, T. M.; ALVES, V. P. Pequenas cidades, problemas urbanos e participação social na perspectiva da população local. **Ateliê Geográfico**, v. 14, n. 3, p. 218–237, 2020. Disponível em: https://revistas.ufg.br/atelie/article/view/64370. Acesso em: 07 jun. 2025.

GIARETTA, J. B. Z.; FERNANDES, V.; JR., A. P. Desafios e condicionantes da participação social na gestão ambiental municipal no brasil. **Organizações Sociedade**, v. 27, n. 94, p. 635–656, 2020. Disponível em: https://periodicos.ufba.br/index.php/revistaoes/article/view/11211. Acesso em: 16 maio 2025.

GUILHERME, A. Ciência cidadã como estratégia de escolarização aberta em biodiversidade tornando estudantes atores responsáveis no amazonas. **Diálogo Educacional**, v. 22, n. 72, p. 1802–1823, 2022. Disponível em: https://periodicos.pucpr.br/dialogoeducacional/article/download/30065/26108/69220. Acesso em: 28 maio 2025.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Perfil dos Municípios Brasileiros**: aspectos urbanos e ambientais. 2020. Disponível em: https://www.ibge.gov.br. Acesso em: 5 jul. 2025.

INSTITUTO BRASILEIRO DO MEIO AMBIENTE E DOS RECURSOS NATURAIS RENOVÁVEIS. **Sistema Nacional de Controle da Origem dos Produtos Florestais (Sinaflor)**. 2025. Disponível em: https://www.gov.br/ibama/pt-br/servicos/sistemas/sinaflor. Acesso em: 2 jun. 2025.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. **Uses and Applications**. 2025. Disponível em: https://www.gov.br/inpe/en/programs/amazonia1/uses-and-applications. Acesso em: 2 jun. 2025.

LEAL, L. d. F. S.; VASCONCELOS, B. R. d.; RODRIGUES, H. d. C.; LEAL, L. S. Erosão da governança ambiental e participação popular: a exclusão da sociedade civil nos processos decisórios ambientais. **Contribuciones a las Ciencias Sociales**, v. 17, n. 6, p. e7627, 2024. Disponível em: https://ojs.revistacontribuciones.com/ojs/index.php/clcs/article/view/7627. Acesso em: 17 maio 2025.

LIU, H.; ZHANG BEIBEI", e. J.; LU, W.; PENG, Y.; YUAN, H.; WANG, D. Mechanism and collaborative governance of public participation in urban renewal project. In: INTERNATIONAL SYMPOSIUM ON ADVANCEMENT OF CONSTRUCTION MANAGEMENT AND REAL ESTATE, 27., 2023. **Proceedings [...]**. Singapore: Springer Nature Singapore", 2023. p. 1405–1418. ISBN 978-981-99-3626-7.

MARICATO, E. **O impasse da política urbana no Brasil**. Petrópolis: Vozes, 2011.

MARTIN, R. C. **Clean Code**: A handbook of agile software craftsmanship. [*S. l.*]: Prentice Hall, 2008.

MARTIN, R. C. **Clean Architecture**: A craftsman's guide to software structure and design. Upper Saddle River, NJ, USA: Prentice Hall Press, 2017.

MCCONNELL, S. **Code Complete**: A practical handbook of software construction. [*S. l.*]: Microsoft Press, 2004.

MEDEIROS, R. F. d. Gestão ambiental como instrumento de preservação dos recursos naturais. **Revista Brasileira de Direito e Gestão Pública**, v. 11, n. 2, p. 802–818, jul. 2023. Disponível em: https://www.gvaa.com.br/revista/index.php/RDGP/article/view/9844. Acesso em: 2 jun. 2025.

MELO, A. C. de; FAGUNDES, R.; SANTOS, W. Um guia para identificação e mensuração de dívida técnica de requisitos no desenvolvimento de software. In: CONGRESSO BRASILEIRO DE SOFTWARE: TEORIA E PRÁTICA, 11., 2020. **Anais Estendidos [...]**. Porto Alegre, RS, Brasil: SBC, 2020. p. 8–14. ISSN 0000-0000. Disponível em: https://sol.sbc.org.br/index.php/cbsoft_estendido/article/view/14603. Acesso em: 24 maio 2025.

MIRANDA, D. T.; DECESARO, G. D. Os impactos e as consequências gerados pela urbanização acelerada às águas urbanas. **Revista Técnico-Científica do CREA-PR**, v. 13, p. 02–09, 2018. ISSN 2358-5420. Disponível em: https://revistatecie.crea-pr.org.br/index.php/revista/article/view/404. Acesso em: 07 jun. 2025.

ORGANIZAÇÃO DAS NAÇÕES UNIDAS. **Transformando Nosso Mundo**: A agenda 2030 para o desenvolvimento sustentável. 2015. Disponível em: https://brasil.un.org/pt-br/91863-agenda-2030-para-o-desenvolvimento-sustentavel. Acesso em: 16 maio 2025.

PARMAR, G.; LAKHANI, S.; CHATTOPADHYAY, M. K. An iot based low cost air pollution monitoring system. In: INTERNATIONAL CONFERENCE ON RECENT INNOVATIONS IN SIGNAL PROCESSING AND EMBEDDED SYSTEMS (RISE). **Proceedings [...]**. [*S. l.*], 2017. p. 524–528.

PARNAS, D. L. On the criteria to be used in decomposing systems into modules. **Communications of the ACM**, ACM, v. 15, n. 12, p. 1053–1058, 1972.

PEREIRA, J. H. de O.; SANTOS, E. O. dos; VIANA, B. A. da S. PolÍtica nacional de resÍduos sÓlidos brasileira: Conquistas e desafios. **Revista Homem, Espaço e Tempo**, v. 17, n. 1, p. 06–22, fev. 2024. Disponível em: //rhet.uvanet.br/index.php/rhet/article/view/498. Acesso em: 14 jun. 2025.

QUEIROZ, G. T. d. M. P. C. *et al.* Guardiões da Chapada: Um aplicativo colaborativo para o combate a incêndios na chapada diamantina. In: WORKSHOP SOBRE ASPECTOS DA INTERAçãO HUMANO-COMPUTADOR PARA A WEB SOCIAL. **Anais [...]**. Sociedade Brasileira de Computação, 2023. p. 11–20. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/25104. Acesso em: 07 jun. 2025.

RAMOS, A. P. d. S. A participação cidadã como um dos princípios de governo aberto. **Cadernos Gestão Pública e Cidadania**, v. 24, n. 80, p. 1–21, 2019. Disponível em: https://www.scielo.br/j/cgpc/a/qsfDXzyN9BZNzwXtyrCPb9G/. Acesso em: 28 maio 2025.

REIS, P. R.; WERNECK, A. L. M. Participação cidadã, pseudoparticipação ou apenas retórica? In: ENCONTRO DA ANPAD (ENANPAD). **Anais [...]**. 2017. Disponível em: https://anpad.com.br/uploads/articles/120/approved/69cd21a0e0b7d5f05dc88a0be36950c7.pdf. Acesso em: 28 maio 2025.

RIOS, N.; SPíNOLA, R.; MENDONçA, M. Organização de um conjunto de descobertas experimentais sobre causas e efeitos da dívida técnica através de uma família de surveys globalmente distribuída. In: CONGRESSO BRASILEIRO DE SOFTWARE: TEORIA E PRáTICA, 12., 2020, Joinville. **Anais Estendidos [...]**. Porto Alegre, RS, Brasil: SBC, 2021. p. 80–94. ISSN 0000-0000. Disponível em: https://sol.sbc.org.br/index.php/wcama/article/view/11022. Acesso em: 07 jun. 2025.

ROSA, E. M. S.; BERNALDINO, E. d. S.; BARBA, C. H. d.; SIMãO, B. P. As tecnologias da informação e comunicação (tics) na educação ambiental de estudantes da educação profissional e tecnológica em rondônia. **Revista e-Curriculum**, scielo, v. 21, 00 2023. ISSN 1809-3876. Disponível em: http://educa.fcc.org.br/scielo.php?script=sci_arttext&pid=S1809-38762023000100110&nrm=iso. Acesso em: 02 jun. 2025.

SAMPAIO, G. M.; ASSIS, B. d. P.; SANTOS, J. A. B. d. Integração das tecnologias da informação e comunicação (tics) nas políticas públicas: Promovendo a participação social na gestão do patrimônio cultural e respeitando os direitos humanos nas territorialidades. **Revista Orbis Latina**, v. 15, n. 1, 2025. Disponível em: https://revistas.unila.edu.br/orbis/article/view/5192. Acesso em: 25 maio 2025.

SANTOS, D.; JúNIOR, P. A.; COSTA, H. Uma abordagem para reestruturação de sistemas de software orientados a objetos. In: SIMPOSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 15., 2016, Maceió. **Anais [...]**. Porto Alegre, RS, Brasil: SBC, 2016. p. 286–300. Disponível em: https://sol.sbc.org.br/index.php/sbqs/article/view/15141. Acesso em: 22 maio 2025.

SEIFFERT, M. E. B. **ISO 14001 Sistema de Gestão Ambiental**: implantação objetiva e econômica. 4. ed. São Paulo (SP): Editora Atlas S.A., 2011. 140-147 p.

SEIXAS, L.; SACCARO JUNIOR, N. L. Td 2808 - o licenciamento como instrumento de regulação ambiental : desafios, propostas e perspectivas. **Texto para Discussão**, p. 1–21, 11 2022. Disponível em: https://www.econstor.eu/bitstream/10419/284864/1/TD2808.pdf. Acesso em: 10 maio 2025.

SERVIDONI, L. E.; SILVA, F. M. d.; MACHADO, C. A.; OLIVEIRA, M. C. d.; MENEZES, E. Avaliação de risco a enchentes e inundações por krigagem ordinária em sistemas de informação geográfica. **Caderno de Geografia**, Belo Horizonte, v. 29, n. Especial 1, p. 126–143, 2019. Disponível em: https://periodicos.pucminas.br/geografia/article/view/p.2318-2962.2019v29nespp126/14887. Acesso em: 28 maio 2025.

SILVA, A. D. O. F. D. S. D.; BRUNO, D. R. Produção mais limpa: contribuições da indústria 4.0 para a sustentabilidade — uma análise crítica. **Revista Interface Tecnológica**, v. 20, n. 2, p. 634–644, dez. 2023. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/1798. Acesso em: 07 jun. 2025.

SILVA, F. J. M. d. **QuixAlert! App — Versão 2.0 Inicial**. 2025. Disponível em: GitHub. Acesso em: 17 jun. 2025. Disponível em: https://github.com/QuixAlert/quixalert-app-v2/releases/tag/v2.0-inicial. Acesso em: 07 jun. 2025.

SILVA, M. H. C. d.; LIMA, L. N. F. d.; SILVA, C. S. e.; SILVA, B. V. d.; TAVARES, H. S. d. A.; FALCãO, W. H. d. R.; SOUSA, M. L. P. S.; LIMA, S. C. Resíduos sólidos: o uso da gestão ambiental como ferramenta para o manejo adequado do lixo urbano / solid waste:

the use of environmental management as a tool for the proper management of urban waste. **Brazilian Journal of Development**, v. 6, n. 11, p. 85668–85677, Nov. 2020. Disponível em: https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/19447. Acesso em: 13 jun. 2025.

SILVA, N. G. d.; SILVA, R. N. d.; ANDRADE, H. M. L. d. S.; ANDRADE, L. P. d. As incógnitas e os desafios da inclusão participativa da governança ambiental: uma revisão sistemática de literatura. **Revista Geama**, v. 8, n. 1, p. 14–24, 2022. Disponível em: https://www.journals.ufrpe.br/index.php/geama/article/view/4261. Acesso em: 16 maio 2025.

SILVA, R. A.; COSTA, J. M. da. Arquitetura modular e escalável para desenvolvimento de aplicativos móveis no escritório de soluções criativas. **Facit Business and Technology Journal**, Faculdade FACIT, v. 1, n. 55, p. 1–10, 2024. Disponível em: https://revistas.faculdadefacit.edu.br/index.php/JNT/article/view/3051. Acesso em: 07 jun. 2025.

SISTEMA DE INFORMAÇÕES SOBRE A BIODIVERSIDADE BRASILEIRA. **Ciência cidadã | Urubu**. 2025. Disponível em: https://sibbr.gov.br/cienciacidada/urubu.html. Acesso em: 2 jun. 2025.

SOMMERVILLE, I. **Engenharia de Software**. 10. ed. [*S. l.*]: Pearson Education, 2020.

SUPERINTENDÊNCIA ESTADUAL DO MEIO AMBIENTE (CE). **Semace recebe denúncia de crime ambiental por aplicativo**. 2019. Disponível em: https://www.semace.ce.gov.br/2019/06/17/semace-recebe-denuncia-de-crime-ambiental-por-aplicativo/. Acesso em: 2 jun. 2025.

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO. **FFP lança aplicativo para monitoramento ambiental**. 2025. Disponível em: https://www.uerj.br/agenda/17051/. Acesso em: 2 jun. 2025.

WIBOWO, A.; CHRISMANTO, A. R.; RINI, M. N. A.; CHRISANTYO, L. Mobile application performance improvement with the implementation of code refactor based on code smells identification: Dutataniku agriculture mobile app case study. In: CONFERENCE ON INFORMATICS AND COMPUTING (ICIC), 7., 2022. **Proceedings [...]**. [*S. l.*], 2022. p. 1–7.

ZHONG, S.; ZHANG, K.; BAGHERI, M.; BURKEN, J. G.; GU, A.; LI, B.; MA, X.; MARRONE, B. L.; REN, Z. J.; SCHRIER, J.; SHI, W.; TAN, H.; WANG, T.; WANG, X.; WONG, B. M.; XIAO, X.; YU, X.; ZHU, J. J.; ZHANG, H. Machine learning: New ideas and tools in environmental science and engineering. **Environmental Science Technology**, v. 55, n. 19, p. 12741–12754, 2021. Disponível em: https://pubs.acs.org/doi/full/10.1021/acs.est.1c01339. Acesso em: 28 maio 2025.

7 FORMULÁRIO DE VALIDAÇÃO DO *QUIXALERT!*

Documento que detalha o processo de validação da plataforma de monitoramento ambiental, o *QuixAlert!*.

Plataforma de Monitoramento Ambiental

Formulário de Validação

Av. José de Freitas Queiroz, 5003, Quixadá - CE, 63902-580
04.05.2025

Formulário de Validação com o Cliente (Preenchido pelo pesquisador)

Título do Projeto: Plataforma QuixAlert! – Monitoramento e Gestão Ambiental Urbana

Dados do Cliente: Autarquia Municipal do Meio Ambiente de Quixadá (AMMA)

Data da validação com o cliente: 01 / 02 / 2025

Versão do sistema apresentada: Aplicativo QuixAlert! v1.2

1. Informações sobre a Reunião de Validação

- 1.1. A validação foi realizada com:
  - ( ) Cliente final
  - (X) Representante da área
  - ( ) Outro: _____
- 1.2. Forma de validação:
  - (X) Presencial
  - (X) Videoconferência
  - ( ) Troca de mensagens
  - ( ) Outra: _____
- 1.3. Quantas pessoas participaram da validação?
  - 2 Especialistas
  - 2 Representantes do Cliente
  - 15 Especialistas Ambientais
- 1.4. Duração aproximada das reuniões:
  - Reunião presencial durou 100 minutos
  - Reunião remota durou 120 minutos

---

2. Apresentação do Sistema

2.1. Quais telas ou funcionalidades foram apresentadas?
Foram apresentados os fluxos principais da aplicação, incluindo:

- Tela de Login e Cadastro de usuário;
- Tela principal;
- Tela de feed de notícias;

- Fluxo de criação de denúncia ambiental e de maus-tratos;
- Fluxo de solicitação de documentos ambientais;
- Fluxo de visualização e candidatura para adoção de animais;
- Navegação geral e acesso ao perfil do usuário.

2.2. O cliente interagiu diretamente com o sistema?

- (X) Sim
- ( ) Não
- ( ) Parcialmente

---

3. Feedback Coletado

- 3.1. O cliente compreendeu o funcionamento geral do sistema?
  - (X) Sim
  - ( ) Parcialmente
  - ( ) Não

- 3.2. O cliente sugeriu mudanças?
  - (X) Sim
  - ( ) Não
  - Se sim, descreva as principais:

  Sim. O representante do cliente sugeriu melhorias importantes no módulo de adoção de animais. As principais sugestões foram:

  - Desenvolver um sistema de filtros no módulo de adoção;
  - Implementar uma galeria de mídias na página de detalhes de cada animal;
  - Criar um módulo completo de doação com integração com os bancos;
  - Adicionar uma barra de pesquisa global;
  - Criar um módulo de chat, permitindo a comunicação direta entre usuários e a equipe da AMMA para esclarecimento de dúvidas.

- 3.3. Alguma funcionalidade foi considerada não útil ou desnecessária?
    - ( ) Sim
    - (X) Não
    - Se sim, qual?

- 3.4. Alguma funcionalidade não prevista foi solicitada?
    - (X) Sim
    - ( ) Não
    - Descreva:
- Sim. O cliente solicitou a inclusão de uma nova funcionalidade de doação financeira. Ele destacou que muitos cidadãos desejam ajudar a causa animal, mas não têm condições de adotar. Uma opção de doação permitiria que essas pessoas contribuíssem com recursos para ração, medicamentos e outros cuidados, aumentando o engajamento da comunidade com a AMMA.

---

4. Ações Realizadas Após a Validação

- 4.1. Quais melhorias foram implementadas no sisrema após a reunião?

As sugestões do cliente foram consideradas de alto valor e totalmente alinhadas aos objetivos do projeto. As três principais solicitações (filtros para animais, galeria de mídia e funcionalidade de doação) foram adicionadas ao backlog de requisitos do projeto para serem desenvolvidas na fase de evolução do software, conforme detalhado no Capítulo 4 deste trabalho. Após o desenvolvimento dessas novas funcionalidades, o sistema passará por uma nova validação.

---

5. Processo de Validação

Para validar a versão inicial do QuixAlert! foi escolhida a técnica de teste de usabilidade guiado. Uma técnica muito utilizada e com um bom nível de feedback.

A validação do sistema guiada é um processo que envolve a apresentação e teste de um protótipo ou sistema interativo para obter feedback e avaliar sua eficácia. Nesse tipo de validação, os participantes são orientados por um facilitador a realizar tarefas específicas no aplicativo, enquanto suas interações e comentários são observados e registrados.

Essa abordagem permite que os desenvolvedores e designers obtenham informações valiosas sobre a usabilidade, funcionalidade e experiência do usuário do sistema. Ao realizar testes guiados, é possível identificar pontos fortes, pontos problemáticos e áreas que precisam de melhorias, além de receber sugestões e opiniões dos usuários. A validação do sistema guiada ajuda a garantir que o produto final atenda às expectativas dos usuários e cumpra seus objetivos.

Roteiro

Aqui está uma avaliação guiada do QuixAlert! com uma sequência de passos para que o usuário possa navegar e explorar o sistema:

Passo 1: Tela de boas-vindas (Login/Registro)

- Abra o aplicativo do QuixAlert! em seu dispositivo móvel.
- Na tela de boas-vindas, escolha entre login e registrar para acessar a tela principal do sistema.

Passo 2: Tela principal

- Agora você está na tela principal do QuixAlert!.
- Observe o feed de notícias ambientais e animais para adoção.
- Role verticalmente para ver as informações atualizadas.

Passo 3: Explorando recursos principais

- Na parte inferior da tela, você verá botões de navegação.
- Toque no botão "Animais" para acessar o menu de animais disponíveis para adoção.
- Toque no botão "Notícias" para acessar informações sobre questões ambientais.
- Explore outros botões de navegação, como "Solicitação de Documentação Ambiental" e outros recursos relevantes.

Passo 4: Fazendo uma denúncia

- Toque no botão "Realizar Denúncia".

- Na tela de denúncias, preencha as informações necessárias, como localização exata do problema, descrição detalhada e, se possível, anexe fotos como evidência.
- Após preencher as informações, toque no botão "Enviar" para registrar sua denúncia.

Passo 5: Explorando recursos adicionais
- Volte para a tela principal.
- Ir até ao menu números de telefones para emergência

Passo 6: Opção de Perfil
- Toque no ícone do perfil na barra de navegação ou na foto do usuário.
- Explore a tela e veja as possibilidades.

Passo 7: Notificações
- Observe o ícone de notificações na parte superior da tela.
- Toque no ícone para acessar a lista de notificações recebidas.
- Verifique se há atualizações de denúncias, informações importantes ou outros alertas relevantes.

Essa avaliação guiada permite que o usuário navegue e explore os recursos principais do QuixAlert!, incluindo a tela principal, denúncias, notícias, licenciamento ambiental, perfil e notificações. Os passos fornecidos ajudam a familiarizar o usuário com as funcionalidades do sistema e a obter uma visão geral de como utilizar o aplicativo de forma eficiente.

No caso específico do QuixAlert!, a validação do aplicativo guiada foi conduzida em diferentes etapas, envolvendo o cliente, um usuário representativo e especialistas em design. Essa abordagem permitiu que o pesquisador coletasse feedback abrangente, abordando as perspectivas do cliente, dos usuários finais e de profissionais experientes. O processo de validação guiada contribuiu para aprimorar o sistema, tornando-o mais intuitivo, eficiente e capaz de atender às necessidades e expectativas dos usuários.

Resultados

Validação com o usuário

A validação com a Larissa foi conduzida de forma presencial, onde ela teve acesso ao documento de requisitos do QuixAlert! e pôde testar o aplicativo em

um ambiente controlado. Durante a validação, a Larissa desempenhou o papel de um usuário representativo e teve a oportunidade de interagir com o sistema, realizar tarefas específicas e fornecer feedback sobre sua experiência.

Durante o teste, a Larissa sugeriu duas melhorias importantes. Primeiramente, ela expressou o desejo de ver uma seção dedicada à exibição de animais na tela principal do aplicativo. Essa sugestão foi considerada relevante, pois proporciona uma maneira visualmente atraente e fácil de acessar informações sobre animais disponíveis para adoção, incentiva a conscientização e ajuda a promover a adoção responsável.

Além disso, a Larissa mencionou a importância de ter uma opção de pesquisa no QuixAlert!. Essa funcionalidade permitiria aos usuários procurar informações específicas, como notícias, dicas de sustentabilidade, eventos ambientais, entre outros. A inclusão dessa opção de pesquisa facilita a navegação e o acesso rápido a conteúdos relevantes, tornando a experiência do usuário mais eficiente e personalizada.

As sugestões da Larissa foram cuidadosamente registradas e levadas em consideração pelo pesquisador e desenvolvedor do QuixAlert!. Essas melhorias propostas contribuíram para aprimorar a usabilidade e a funcionalidade do sistema, garantindo que atenda às necessidades dos usuários finais de forma mais abrangente e satisfatória.

Validação com os especialistas

A validação com os especialistas em design, Guilherme Mendes e Iana Oliveira, também foi realizada após a análise do documento de requisitos e a disponibilização da versão do aplicativo do QuixAlert!. Essa validação teve como objetivo obter feedback específico sobre aspectos estéticos e de design, visando aprimorar a interface do aplicativo.

Durante a validação, Guilherme e Iana examinaram cuidadosamente o layout, as cores, a tipografia e os elementos visuais presentes na versão inicial. Eles utilizaram seu conhecimento e experiência em design para avaliar a consistência, a harmonia e a atratividade geral da interface.

Com base em sua avaliação, eles sugeriram alguns ajustes estéticos que poderiam ser feitos para melhorar a experiência visual do usuário. Isso incluiu recomendações de cores mais adequadas à proposta do aplicativo, ajustes de espaçamento entre elementos para melhorar a legibilidade e proporções, e aprimoramentos na escolha de ícones e botões.

Essas sugestões de ajustes estéticos foram consideradas valiosas pelo desenvolvedor QuixAlert!. Eles reconheceram a importância de uma interface visualmente agradável e intuitiva, e trabalharam em colaboração com os especialistas em design para implementar as melhorias propostas. Isso resultou em uma interface mais atraente e coerente, que contribui para a usabilidade e a experiência positiva do usuário ao utilizar o aplicativo.

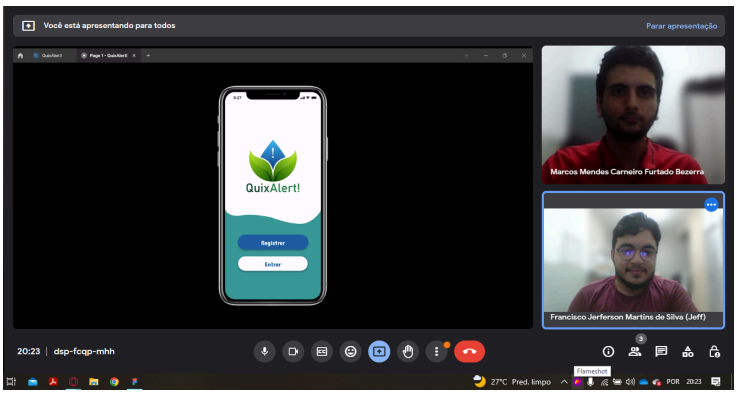
Validação com o cliente

Durante a validação com o cliente Marcos Mendes, ele teve acesso ao documento de requisitos do QuixAlert! e também teve a oportunidade de testar a versão do aplicativo. Sua participação foi fundamental para a identificação de melhorias e ajustes que poderiam ser feitos, com base em sua perspectiva como usuário e também em suas necessidades específicas.

Uma das principais contribuições do Marcos foi a sugestão de inclusão de filtros para facilitar a busca por animais específicos. Ele destacou a importância de ter a opção de filtrar os animais por tipo, como cachorros e gatos, para que os usuários pudessem encontrar mais rapidamente os animais de seu interesse.

Outra sugestão valiosa do Marcos foi a adição de uma galeria de imagens na página de cada animal. Ele apontou que ter acesso a mais fotos do animal, mostrando diferentes ângulos e expressões, poderia ajudar os usuários a terem uma melhor compreensão de sua aparência e personalidade, auxiliando na decisão de adoção.

Além disso, o Marcos sugeriu a inclusão da opção de doação no aplicativo. Ele destacou que nem todos os usuários têm a possibilidade de adotar um animal de estimação, mas ainda assim desejam contribuir para o bem-estar dos animais. A opção de doação permitiria que as pessoas apoiarem financeiramente os animais do abrigo, fornecendo recursos para cuidados médicos, alimentação e outras necessidades.



As sugestões de melhorias feitas pelo Marcos foram muito bem recebidas pelo desenvolvedor do QuixAlert!. Eles entenderam a importância de personalizar a experiência do usuário, facilitar a busca por animais desejados e oferecer opções adicionais para aqueles que desejam ajudar, mas não podem adotar. Assim, as sugestões do Marcos foram implementadas no aplicativo, resultando em uma versão mais abrangente e satisfatória para os usuários.

Todas as contribuições e sugestões feitas pelos validadores foram cuidadosamente analisadas e consideradas durante o processo de validação do QuixAlert!. Valorizamos a participação e o envolvimento de cada validador, pois reconhecemos a importância de incorporar diferentes perspectivas para garantir a qualidade e a efetividade do sistema.

Ao receber o feedback do cliente Marcos Mendes, da usuária Larissa Barbosa e dos especialistas em design Guilherme Mendes e Iana Oliveira, o desenvolvedor avaliou cada sugestão e avaliou sua viabilidade e relevância para o projeto. Todas as contribuições foram consideradas benéficas e alinhadas com os objetivos do QuixAlert! e serão implementadas na versão final .

Entre as melhorias sugeridas, inclui-se a adição de um filtro de cachorros e gatos para facilitar a busca por animais específicos, a inclusão de uma galeria de imagens nas páginas dos animais para proporcionar uma visão mais abrangente, e a disponibilização da opção de doação, permitindo que as pessoas apoiem os animais sem necessariamente adotá-los.

Essas alterações, juntamente com outras sugestões que serão implantadas, resultarão em um sistema aprimorado e mais completo do QuixAlert!. Agradecemos a todos os validadores por sua valiosa contribuição e por ajudarem a moldar um sistema que atende às necessidades e expectativas dos usuários. Seu envolvimento e participação foram fundamentais para o sucesso deste projeto.

8 DODCUMENTO DE REQUISITOS DO *QUIXALERT!*

Documento que detalha todo o escopo da plataforma de monitoramento ambiental, o *QuixAlert!*. O presente documento detalha o processo de elicitação de requisitos e também todo o escopo do projeto de desenvolvimento da primeira versão do QuixAlert!

Plataforma de Monitoramento Ambiental

Documento de Requisitos

Av. José de Freitas Queiroz, 5003, Quixadá - CE, 63902-580
12.03.2025

Personas

Nome: Pedro, o Defensor Ambiental
Idade: 42 anos
Profissão: Engenheiro Civil

Descrição:
Pedro é um engenheiro civil engajado na preservação ambiental. Ele tem uma preocupação especial com a sua cidade, Quixadá, e está determinado a combater os problemas de alagamentos, descarte inadequado de lixo e também a situação dos animais de rua. Pedro acredita que a documentação visual é uma forma poderosa de chamar a atenção para essas questões e pressionar por mudanças.

Motivações:
Pedro é motivado por sua experiência profissional e conhecimento técnico em engenharia civil. Ele entende as consequências negativas que os alagamentos, o descarte indevido de lixo e a falta de cuidados com os animais de rua podem causar na infraestrutura urbana, no meio ambiente e no bem-estar dos animais. Sua motivação é proteger a cidade, promover a conscientização sobre esses problemas e buscar soluções efetivas, incentivando ações por parte das autoridades e envolvimento da comunidade.

Comportamentos:
Pedro está sempre preparado para documentar e denunciar situações de alagamentos, descarte inadequado de lixo e o abandono de animais em Quixadá. Ele utiliza seu smartphone e câmera para capturar imagens e vídeos claros e informativos, destacando os locais afetados e as condições precárias dos animais. Pedro também realiza pesquisas e coleta dados complementares para fornecer uma visão abrangente das situações denunciadas.
Além de enviar as denúncias através do QuixAlert, Pedro utiliza suas habilidades de comunicação para compartilhar as informações com outras organizações ambientais, mídias locais e grupos de proteção animal, ampliando o alcance de suas denúncias e aumentando a conscientização sobre a situação dos animais de rua em Quixadá.

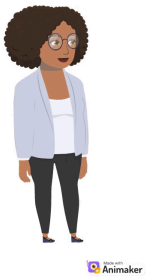
Necessidades:
Como usuário do QuixAlert, Pedro valoriza uma plataforma fácil de usar, com recursos intuitivos que permitem o envio rápido e eficiente de imagens e vídeos. Ele busca uma interação ágil com as autoridades competentes, esperando que suas denúncias sejam tratadas com seriedade e que as medidas corretivas sejam tomadas tanto em relação aos alagamentos e ao descarte de lixo, quanto ao bem-estar dos animais de rua. Pedro também valoriza a colaboração e o compartilhamento de conhecimento com outros usuários interessados na preservação ambiental e na proteção animal.

Objetivos:
O principal objetivo de Pedro é utilizar o QuixAlert como uma ferramenta para criar um impacto positivo em Quixadá. Ele deseja influenciar as autoridades a tomar medidas efetivas para prevenir alagamentos, promover o correto descarte de lixo e melhorar a

situação dos animais de rua na cidade. Além disso, Pedro busca conscientizar a população sobre a importância da preservação ambiental, do respeito aos animais e incentivar a mudança de hábitos e o engajamento da comunidade na resolução desses problemas.

Nome: Carolina, a Verificadora Cautelosa
Idade: 35 anos
Profissão: Gestora Ambiental

Descrição:
Carolina é uma gestora ambiental comprometida em garantir a qualidade e a veracidade das denúncias recebidas no QuixAlert. Ela possui experiência em análise de dados ambientais com um olhar crítico para identificar problemas e soluções relacionadas a alagamentos e descarte inadequado de lixo em Quixadá. Carolina está dedicada a garantir que as denúncias sejam analisadas e validadas com precisão, visando uma resposta adequada.



Motivações:
Carolina é motivada pelo seu amor pela natureza e pelo desejo de proteger o meio ambiente. Ela acredita que sua atuação na validação das denúncias recebidas pelo QuixAlert é fundamental para a eficácia do sistema e para a resolução dos problemas ambientais em Quixadá. Carolina busca contribuir para a conscientização da comunidade e para a tomada de medidas assertivas por parte das autoridades competentes.

Comportamentos:
Carolina é dedicada e minuciosa em seu trabalho. Ela utiliza as ferramentas disponíveis no QuixAlert para acessar as denúncias recebidas, analisando cuidadosamente as informações fornecidas pelos usuários. Ela verifica a autenticidade das imagens e vídeos enviados, cruzando dados com outras fontes e realizando pesquisas complementares, a fim de obter uma visão abrangente das situações denunciadas.
Além disso, Carolina interage com outros usuários do QuixAlert, esclarecendo dúvidas, solicitando informações adicionais e fornecendo feedback sobre as denúncias. Ela se preocupa em manter uma comunicação transparente e construtiva, buscando soluções colaborativas para os problemas ambientais.

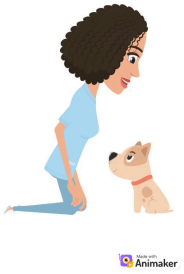
Necessidades:
Como usuária do QuixAlert, Carolina valoriza uma plataforma intuitiva e organizada, que facilite a visualização e a análise das denúncias recebidas. Ela busca recursos que facilitem a verificação das informações, como ferramentas de pesquisa e acesso a dados atualizados sobre a região de Quixadá.
Carolina também necessita de uma comunicação eficiente com os usuários, para poder solicitar informações adicionais e fornecer feedback sobre as denúncias. Ela valoriza a interação colaborativa, pois acredita que isso contribui para uma resposta mais precisa e efetiva aos problemas ambientais.

Objetivos:

O principal objetivo de Carolina é garantir que as denúncias recebidas sejam analisadas e validadas com precisão, visando uma ação adequada por parte das autoridades e a resolução dos problemas ambientais em Quixadá. Ela busca fornecer um feedback claro e construtivo aos usuários, auxiliando-os na compreensão do processo e na importância de suas denúncias.

Nome: Luiza, a Amante dos Animais
Idade: 29 anos
Profissão: Professora de Educação Infantil

Descrição:
Luiza é uma professora de educação infantil apaixonada por animais e comprometida em proporcionar um lar amoroso para um animal de rua. Ela é uma defensora dos direitos dos animais e está determinada a adotar um pet abandonado para oferecer-lhe uma vida cheia de carinho e cuidados. Luiza utiliza o QuixAlert como uma ferramenta para encontrar informações sobre animais disponíveis para adoção em Quixadá, buscando um companheiro leal e amoroso.



Motivações:
Luiza é motivada por sua conexão profunda com os animais e pelo desejo de proporcionar um ambiente seguro e amoroso para um pet em situação de rua. Ela acredita que a adoção é uma forma de salvar vidas e oferecer uma segunda chance aos animais abandonados. Luiza busca encontrar um companheiro que traga alegria e amor para sua vida, ao mesmo tempo em que oferece uma vida melhor a um animal necessitado.

Comportamentos:
Luiza utiliza o QuixAlert para pesquisar informações sobre animais disponíveis para adoção em Quixadá. Ela navega pela seção de animais de rua e adoção, visualizando fotos, descrições e informações sobre os pets em busca de um lar. Luiza procura animais que se encaixem no seu estilo de vida e nas suas preferências, como tamanho, idade e temperamento.
Além disso, Luiza interage com outros usuários do QuixAlert, fazendo perguntas sobre os animais disponíveis para adoção, buscando informações adicionais sobre o processo de adoção responsável e compartilhando sua empolgação e expectativas. Ela valoriza a transparência e a confiabilidade das informações fornecidas pelos usuários e pelas instituições envolvidas na adoção.

Necessidades:
Luiza busca uma plataforma amigável e de fácil navegação no QuixAlert, que permita encontrar rapidamente informações atualizadas sobre animais disponíveis para adoção em Quixadá. Ela valoriza a clareza e a veracidade das descrições e fotos dos animais, bem como informações sobre seu histórico médico, vacinação e esterilização.
Luiza também deseja encontrar recursos educativos no QuixAlert que a auxiliem no processo de adoção responsável, fornecendo orientações sobre cuidados básicos, treinamento e integração do animal ao novo lar. Ela busca uma experiência positiva e

suporte ao longo do processo de adoção, desde a escolha até a adaptação do pet em sua casa.

Objetivos:
O principal objetivo de Luiza é adotar um animal de rua em Quixadá, proporcionando-lhe um lar amoroso e cuidados adequados. Ela deseja encontrar um companheiro fiel e leal, capaz de trazer alegria e amor para sua vida. Luiza está comprometida em seguir os passos necessários para uma adoção responsável, garantindo o bem-estar e a felicidade do animal adotado.

Nome: Roberto, o Observador Informado
Idade: 40 anos
Profissão: Professor de História

Descrição:


Roberto é um professor de história interessado em se manter atualizado sobre os acontecimentos da cidade de Quixadá. Ele é um observador atento e busca informações confiáveis sobre questões relacionadas a alagamentos, descarte de lixo e outros eventos relevantes para a comunidade. Roberto utiliza o QuixAlert como uma fonte confiável de notícias locais, visando compreender os desafios enfrentados pela cidade e sua evolução ao longo do tempo.

Motivações:
Roberto é motivado por sua paixão pela história e pela cidade de Quixadá. Ele acredita que entender os problemas ambientais e sociais enfrentados pela cidade é fundamental para contribuir com a comunidade e promover mudanças positivas. Roberto busca conhecimento e informações confiáveis para embasar suas aulas e ampliar sua compreensão sobre a cidade em que vive.

Comportamentos:
Roberto é um usuário ativo no QuixAlert, dedicando-se principalmente a consumir as notícias e informações publicadas na plataforma. Ele acessa regularmente a seção de notícias, onde busca atualizações sobre eventos relacionados a alagamentos, descarte de lixo e outras questões relevantes para Quixadá.
Além disso, Roberto utiliza a função de pesquisa no QuixAlert para encontrar artigos e relatórios anteriores, buscando entender a evolução dos problemas e as soluções propostas ao longo do tempo. Ele também compartilha algumas das notícias relevantes com seus alunos, utilizando-as como material para discussões em sala de aula.

Necessidades:
Roberto valoriza uma plataforma com uma interface intuitiva e organizada, que permita acessar facilmente as notícias e informações relevantes sobre Quixadá. Ele busca uma experiência de usuário amigável e uma apresentação clara dos artigos, possibilitando uma leitura agradável e eficiente.

Roberto também necessita de informações confiáveis e atualizadas, fornecidas por fontes verificadas e especialistas na área. Ele busca aprofundar seu conhecimento sobre os desafios enfrentados pela cidade e estar ciente das ações em andamento para solucionar esses problemas.

Objetivos:
O principal objetivo de Roberto é se manter informado sobre os acontecimentos em Quixadá, especialmente relacionados a alagamentos, descarte de lixo e outros eventos relevantes. Ele busca compreender as causas e consequências desses problemas, bem como as ações tomadas pela comunidade e pelas autoridades para resolvê-los

Elicitação dos Requisitos

Técnicas de elicitação de requisitos de software são métodos utilizados para identificar, coletar e compreender as necessidades, expectativas e restrições dos stakeholders envolvidos em um projeto de desenvolvimento de software. Essas técnicas são aplicadas para capturar os requisitos do sistema de forma precisa e completa, a fim de garantir que o produto final atenda às necessidades e expectativas dos usuários.

Questionários e entrevistas são técnicas amplamente utilizadas na elicitação de requisitos de software devido às suas vantagens e capacidades de obter informações valiosas dos stakeholders. Aqui está o motivo pelo qual essas técnicas são aplicadas no contexto do QuixAlert:

Questionários: Um questionário é um conjunto de perguntas estruturadas que podem ser respondidas pelos stakeholders de forma assíncrona. É uma técnica eficiente para coletar uma abundância de informações de um grupo amplo de pessoas de forma rápida e econômica. No caso do QuixAlert, um questionário pode ser distribuído para os usuários em potencial, autoridades ambientais, especialistas em meio ambiente e outras partes interessadas para coletar opiniões, sugestões e necessidades em relação às funcionalidades desejadas e à usabilidade do aplicativo.

Entrevistas: As entrevistas são conversas diretas e interativas realizadas entre o analista de requisitos e os stakeholders. Elas permitem uma compreensão mais profunda das necessidades e expectativas dos stakeholders, além de fornecer insights valiosos que podem não ser facilmente obtidos por meio de questionários. No contexto do QuixAlert, as entrevistas podem ser conduzidas com os usuários, autoridades ambientais e especialistas para explorar seus conhecimentos, experiências e perspectivas sobre os problemas ambientais, as funcionalidades desejadas do aplicativo, as preocupações de segurança, entre outros aspectos relevantes.

Ambas as técnicas, questionários e entrevistas, são complementares e podem fornecer uma visão abrangente dos requisitos do QuixAlert, ajudando a identificar as necessidades dos usuários, as expectativas das autoridades e outros fatores que influenciam o design e a funcionalidade do aplicativo. A seguir descrevemos como cada uma das duas técnicas e os seus principais resultados:

Questionário - Perguntas

O questionário foi desenvolvido utilizando a ferramenta Google Formulários, e ele pode ser acessado clicando aqui. O questionário está o mais objetivo possível para tudo que gostaríamos de elicitar. O questionário contava com um Termo de Consentimento Livre e Esclarecido (TCLE), que pode ser acessado clicando aqui.

Seguem as perguntas feitas no formulário:

1. Você considera importante ter um sistema que facilite a denúncia de problemas ambientais, como lixo na rua, poluição e outros problemas relacionados?
   Objetiva:
   (a) Sim
   (b) Não
   (c) Não tenho certeza

2. Quais funcionalidades você gostaria de ver no QuixAlert para ajudar na denúncia e resolução desses problemas?

3. Além das denúncias, você acha importante ter recursos para cuidados de animais de rua, como informações sobre adoção, resgate e apoio veterinário?
   Objetiva:
   (a) Sim, é muito importante
   (b) Talvez, depende da abordagem
   (c) Não é uma prioridade para mim

4. Como você imagina que o QuixAlert poderia facilitar a interação entre os usuários e as autoridades competentes responsáveis pela resolução dos problemas ambientais?

5. Quais são as suas expectativas em relação à interface e usabilidade do QuixAlert? O que você considera importante em termos de facilidade de uso?

6. Além das denúncias, você gostaria que o QuixAlert oferecesse recursos informativos sobre questões ambientais e dicas para práticas sustentáveis?
   Objetiva:
   (a) Sim, seria útil
   (b) Não, prefiro focar apenas nas denúncias
   (c) Estou indiferente

7. Qual é a sua opinião sobre a importância da conscientização ambiental e do engajamento da comunidade na resolução dos problemas ambientais?

8. Você acredita que o QuixAlert pode desempenhar um papel importante na conscientização da população e na promoção de mudanças positivas na cidade?

Objetiva:
(a) Sim, definitivamente
(b) Talvez, depende da adesão da comunidade
(c) Não acredito que tenha um impacto significativo

9. Quais outras sugestões ou necessidades você tem em relação ao QuixAlert e sua funcionalidade?

10. Além das denúncias de problemas ambientais, quais outras informações você gostaria de ter acesso pelo QuixAlert? (ex: notícias, eventos, campanhas, dicas, etc.) (parecida com a 6)

11. Você já utilizou algum aplicativo ou sistema semelhante ao QuixAlert em outras ocasiões?
Objetiva:
(a) Sim, frequentemente
(b) Sim, ocasionalmente
(c) Não, nunca utilizei
(d) Não tenho certeza

12. Qual a frequência com que você enfrenta problemas ambientais no seu dia a dia?
Objetiva:
(a) Diariamente
(b) Semanalmente
(c) Mensalmente
(d) Raramente
(e) Nunca

13. Você já teve experiências anteriores de relatar problemas ambientais? Se sim, como foi essa experiência?

14. Quais são as principais barreiras ou dificuldades que você enfrenta ao relatar um problema ambiental atualmente?

15. Que tipo de informações você acha importante incluir no processo de relato de um problema ambiental? (exemplo: localização, descrição detalhada, fotos)

16. Você gostaria de receber atualizações ou feedback sobre a resolução do problema que você relatou? Por quê?

17. Você acredita que incentivos ou recompensas poderiam encorajar mais pessoas a relatar problemas ambientais? Quais tipos de incentivos você consideraria efetivos?

Questionário - Resultados

Participaram do questionário um total de 40 pessoas, divididas em três (3) diferentes grupos, os grupos foram divididos conforme os conhecimentos de cada um, os grupos são:

- **Especialistas da área ambiental** - contamos com a participação de 10 especialista da área ambiental, entre eles estão os profissionais da AMMA e os estudantes concludentes do curso de Engenharia Ambiental e Sanitária do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) - *campus* Quixadá.
- **Especialistas em desenvolvimento de *software*** - participaram do questionário 10 desenvolvedores com experiência em desenvolvimento de software para o âmbito socioambiental e social. Os desenvolvedores são estudantes da Universidade Federal do Ceará - *campus* Quixadá, atuantes em diversos projetos.
- **Usuários comuns (cidadãos de Quixadá)** - convidamos 20 pessoas que moram em Quixadá para participarem do questionário, conseguimos a participação deles pelo intermédio da AMMA. Então os convidados foram selecionados por eles.

O formulário foi disponibilizado *on-line*, através do Google Formulários e a tabela com todas as respostas obtidas podem ser acessadas clicando aqui.

Vamos listar algumas das principais respostas de diferentes grupos que nos levaram a criar requisitos específicos para o sistema:

Grupo de Especialistas da área ambiental

Pergunta - Quais funcionalidades você gostaria de ver no QuixAlert para ajudar na denúncia e resolução desses problemas?

Respostas -

“Um guia de localização, por exemplo, o local de onde estou denunciando já foi denunciado várias vezes, ai era bom ter um mecanismo pra saber isso”

“Receber notificações sobre o andamento e resolução das denúncias realizadas.”
“Múltiplos canais de denúncia: Ofereça várias opções para que as pessoas possam denunciar problemas, como envio de texto, fotos, vídeos ou até mesmo gravações de áudio. Quanto mais formas de comunicação forem disponibilizadas, maior será a chance de obter informações detalhadas sobre o problema.”

Pergunta - Como você imagina que o QuixAlert poderia facilitar a interação entre os usuários e as autoridades competentes responsáveis pela resolução dos problemas ambientais?

Respostas -

“Notificações em tempo real: O aplicativo pode enviar notificações em tempo real para as autoridades competentes sempre que uma nova denúncia relacionada a problemas

ambientais for feita. Isso permite uma resposta mais rápida e efetiva por parte das autoridades, aumentando as chances de resolução do problema.”

“O QuixAlert pode fornecer um sistema de acompanhamento e relatórios para as autoridades competentes. Isso permite que eles monitorem o status das denúncias, acompanhem o progresso da resolução dos problemas e gerem relatórios detalhados para uma análise mais precisa e eficiente.”

Especialistas em desenvolvimento de *software*

Pergunta - Quais são as suas expectativas em relação à interface e usabilidade do QuixAlert? O que você considera importante em termos de facilidade de uso?

Respostas -

“A estrutura de navegação do QuixAlert deve ser clara e consistente, permitindo que os usuários encontrem facilmente as funcionalidades desejadas. O uso de menus, botões de navegação e ícones familiares ajuda a orientar os usuários e a facilitar o acesso às diferentes partes do aplicativo.”

“É importante fornecer feedback visual adequado para as ações realizadas pelos usuários. Isso inclui indicadores de progresso, animações sutis e mensagens de confirmação para garantir que os usuários se sintam informados sobre o que está acontecendo no aplicativo.”

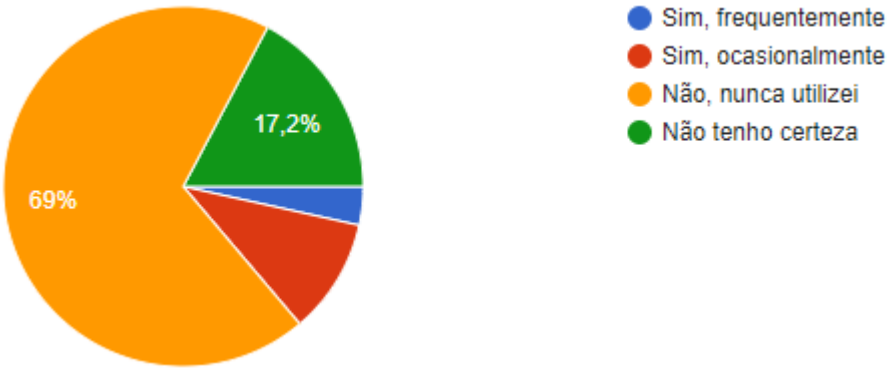
Pergunta - Quais outras sugestões ou necessidades você tem em relação ao QuixAlert e sua funcionalidade?

“Integração com redes sociais: Permitir que os usuários compartilhem suas denúncias e histórias relacionadas a problemas ambientais em suas redes sociais. Isso ajudará a aumentar a conscientização e mobilizar mais pessoas para se envolverem na resolução dos problemas.”

“Implementar elementos de gamificação no QuixAlert para incentivar e recompensar o engajamento dos usuários. Recompensas, níveis, desafios e sistemas de pontos podem estimular a participação ativa e a continuidade no uso do aplicativo.
”

Usuários comuns (cidadãos de Quixadá)

Pergunta - Você já utilizou algum aplicativo ou sistema semelhante ao QuixAlert em outras ocasiões?



Pergunta - Você já teve experiências anteriores de relatar problemas ambientais? Se sim, como foi essa experiência?

Respostas -

“percebi um acúmulo de lixo em uma área verde próxima ao meu bairro.Decidi relatar a situação aos órgãos responsáveis pela limpeza urbana.Embora não tenha recebido um feedback detalhado sobre as medidas tomadas, fiquei satisfeito em saber que a minha denúncia foi atendida e que providências foram tomadas para resolver o problema.”

“Várias vezes precisei relatar problemas em torno da minha casa para órgãos competentes, mas sempre acontece de maneira nada intuitiva.”

Conclusão

Tivemos várias respostas que contribuíram bastante para elicitar requisitos super interessantes. Essas contribuições valiosas dos usuários, especialistas e partes interessadas forneceram *insights* fundamentais para o desenvolvimento do QuixAlert. Com base nessas respostas, pudemos identificar requisitos inovadores e funcionais que atenderão às necessidades da comunidade e auxiliarão na resolução dos problemas ambientais.
As respostas abordaram uma ampla gama de funcionalidades desejadas, incluindo a facilidade de uso da interface, recursos de denúncia de problemas ambientais, integração com autoridades competentes, compartilhamento em redes sociais, acesso a notícias e informações atualizadas, dicas práticas de sustentabilidade e engajamento da comunidade. Essas sugestões refletem a importância da conscientização ambiental, do engajamento da comunidade e da colaboração com as autoridades responsáveis na resolução dos problemas ambientais.

Entrevista - Perguntas

A elicitação de requisitos é uma etapa crucial no processo de desenvolvimento de software, e a entrevista desempenha um papel fundamental nesse contexto. No caso do QuixAlert, um aplicativo voltado para denúncias e conscientização ambiental, a realização de entrevistas com stakeholders relevantes é de extrema importância para identificar e compreender as necessidades e requisitos do sistema.

Durante o processo de elicitação de requisitos do QuixAlert, foram realizadas entrevistas com dois profissionais de destaque na área ambiental: Larissa Barbosa Alves, estudante de Engenharia Ambiental e Sanitária, e Mara Lopes, presidente da Autarquia do Meio Ambiente de Quixadá. Essas entrevistas proporcionaram *insights* valiosos, considerando a *expertise* e experiência desses profissionais.

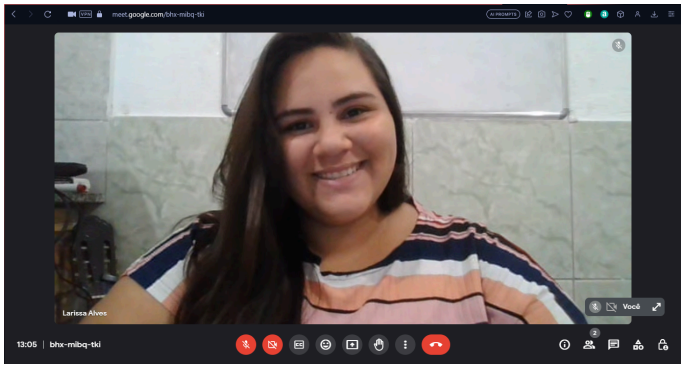


Imagem 1: Entrevista com Larissa Barbosa Alves

Larissa Barbosa Alves, como estudante de Engenharia Ambiental e Sanitária, trouxe uma perspectiva técnica e atualizada sobre os problemas ambientais enfrentados na região. Suas contribuições ajudaram a identificar funcionalidades específicas que poderiam ser implementadas no QuixAlert, considerando as necessidades e expectativas dos usuários.

Por outro lado, a entrevista com Mara Lopes, presidente da Autarquia do Meio Ambiente de Quixadá, permitiu compreender as prioridades, metas e desafios enfrentados pela autarquia em relação à proteção ambiental. Através dessa entrevista, foi possível identificar requisitos que atendessem às demandas da autarquia, como a integração entre o aplicativo e a instituição, a colaboração na resolução dos problemas ambientais e a disseminação de informações relevantes.

Imagem 2: Entrevista com Mara Lopes (AMMA)

A importância dessas entrevistas reside no fato de que elas forneceram um conhecimento aprofundado sobre as necessidades e requisitos específicos do QuixAlert. Ao entrevistar profissionais com *expertise* na área ambiental, foi possível obter *insights* valiosos e direcionados para o desenvolvimento do aplicativo, garantindo que as funcionalidades e características do QuixAlert atendessem às necessidades reais dos usuários e das instituições envolvidas.

Além disso, a entrevista permitiu uma interação direta e um diálogo aberto com os entrevistados, proporcionando a oportunidade de esclarecer dúvidas, explorar sugestões e compreender melhor o contexto em que o QuixAlert seria utilizado. Essa troca de informações contribuiu para um melhor alinhamento entre as expectativas dos stakeholders

e a equipe de desenvolvimento, resultando em um aplicativo mais eficaz e alinhado às necessidades do público-alvo.

O roteiro para a entrevista com a Mara Lopes pode ser acessado clicando aqui. E o roteiro para a entrevista com a Larissa Barbosa Alves pode ser acessado clicando aqui.

Em resumo, as entrevistas realizadas com Larissa Barbosa Alves e Mara Lopes desempenharam um papel crucial na elicitação de requisitos para o QuixAlert. Essa abordagem permitiu aproveitar a *expertise* e experiência desses profissionais para identificar funcionalidades, prioridades e requisitos específicos do aplicativo. Através das entrevistas, foi possível obter informações valiosas e direcionadas, garantindo que o QuixAlert seja uma solução eficaz e alinhada com as demandas do meio ambiente em Quixadá.

Entrevista - Resultados

O resultado das entrevistas foram muito positivas, gravamos todo o processo e os vídeos podem ser acessados clicando aqui. Os vídeos ficaram longos, por tanto, para assisti-los é necessário baixá-los.

Sintetizamos as resposta para uma melhor compreensão da equipe e para facilitar o processo de elicitação de requisitos, seguem as respostas de cada uma das entrevistadas.

Respostas - Mara Lopes (AMMA)

Experiência e Conhecimentos sobre Meio Ambiente:

a) Como presidente da Autarquia do Meio Ambiente de Quixadá, tenho uma vasta experiência na área ambiental, com especialização em gestão de recursos naturais e políticas públicas voltadas para a proteção ambiental.

b) Os principais desafios enfrentados em Quixadá estão relacionados à escassez de água, desmatamento, poluição dos recursos hídricos e degradação do solo. Esses problemas afetam tanto a qualidade de vida da população quanto a preservação dos ecossistemas locais.

c) Nossas principais prioridades são investir em projetos de saneamento básico, promover a educação ambiental nas escolas, implementar ações de reflorestamento e buscar parcerias para a conservação dos recursos naturais.

Conscientização Ambiental e Engajamento da Comunidade:

a) A conscientização ambiental e o engajamento da comunidade são fundamentais para a resolução dos problemas ambientais. É necessário que a população compreenda a importância da proteção ambiental e se envolva ativamente nas ações de preservação.

b) Sim, a autarquia já realizou diversas campanhas de conscientização e engajamento da comunidade. Alguns resultados obtidos incluem o aumento da participação em projetos de reciclagem, o uso mais consciente dos recursos naturais e a redução da poluição nos rios e córregos locais.

c) O QuixAlert pode ser uma ferramenta eficiente para promover a conscientização e o engajamento da comunidade. Através do aplicativo, poderemos divulgar informações sobre questões ambientais, disponibilizar notícias e eventos relacionados, além de fornecer dicas práticas para a população adotar práticas mais sustentáveis em seu cotidiano.

Funcionalidades do QuixAlert:

a) Considero que a funcionalidade mais importante para o QuixAlert seja a possibilidade de denunciar problemas ambientais de forma fácil e rápida. Isso permitirá que a comunidade reporte situações que necessitam de atenção e ação por parte da autarquia.

b) Além das denúncias, seria relevante ter acesso a informações sobre pontos de coleta seletiva, incentivando a prática da reciclagem. Também seria interessante disponibilizar notícias sobre ações ambientais locais, eventos, campanhas e dicas práticas para a preservação do meio ambiente.

c) Em termos de usabilidade, é essencial que o QuixAlert tenha uma interface intuitiva, de fácil navegação e compreensão para que todos os usuários, mesmo aqueles com pouca familiaridade com aplicativos, possam utilizá-lo sem dificuldades. Recursos visuais, como mapas interativos, gráficos e imagens ilustrativas, podem facilitar a compreensão e o engajamento dos usuários.

Integração com a Autarquia do Meio Ambiente:

a) A autarquia está totalmente disponível e disposta a colaborar com o QuixAlert. Podemos contribuir fornecendo informações atualizadas, orientações técnicas, validação das denúncias recebidas, apoio na divulgação de campanhas e eventos ambientais, entre outras formas de parceria.

b) A integração entre o QuixAlert e a autarquia traria benefícios significativos. Poderíamos fornecer suporte técnico, validar a veracidade das denúncias recebidas, auxiliar na definição de ações corretivas e preventivas, além de fornecer feedback contínuo para aprimorar o aplicativo e suas funcionalidades.

Dificuldades e Necessidades Específicas:

a) Uma preocupação importante é garantir que o QuixAlert tenha um sistema eficiente de gerenciamento de denúncias, que permita o registro e a comunicação adequada dos problemas relatados pelos usuários.

b) Nossas necessidades mais urgentes são ter um canal efetivo de comunicação com a comunidade, receber informações precisas e atualizadas sobre os problemas ambientais locais e contar com uma ferramenta que facilite o monitoramento e a resolução dessas questões.

c) A autarquia possui recursos e suporte disponíveis para a implementação e manutenção do aplicativo, seja através de parcerias institucionais, alocação de equipe técnica especializada ou fornecimento de informações e dados relevantes para o funcionamento do QuixAlert.

Respostas - Larissa Barbosa Alves

Pergunta 1:

a) Como estudante de Engenharia Ambiental e Sanitária, minha formação é focada na gestão de recursos hídricos e na busca por soluções sustentáveis para os desafios ambientais.

b) Em Quixadá, enfrentamos alguns desafios ambientais, como a escassez de água, a poluição dos recursos hídricos e a degradação do solo. Esses problemas impactam diretamente a qualidade de vida da população e a saúde dos ecossistemas locais.

c) Considerando as prioridades da região, é fundamental investir em saneamento básico, promover o uso sustentável dos recursos naturais e incentivar a preservação das áreas verdes.

Pergunta 2:

a) A conscientização ambiental e o engajamento da comunidade são pilares fundamentais para a promoção da sustentabilidade e a proteção do meio ambiente. É necessário que as pessoas compreendam a importância da conservação dos recursos naturais e se sintam motivadas a agir de forma responsável.

b) Tive a oportunidade de participar de campanhas de reciclagem e palestras em escolas, onde pude perceber os resultados positivos na conscientização da população. É gratificante ver as pessoas se envolvendo e adotando práticas mais sustentáveis em seu dia a dia.

c) O QuixAlert pode desempenhar um papel importante nesse sentido, fornecendo informações educativas sobre questões ambientais, divulgando eventos e campanhas de conscientização e oferecendo dicas práticas para a preservação do meio ambiente. Através do aplicativo, será possível incentivar a participação ativa da comunidade e promover uma mudança de comportamento em relação à proteção ambiental.

Pergunta 3:

a) No contexto do QuixAlert, algumas funcionalidades são essenciais para atender às necessidades da comunidade. A possibilidade de denunciar problemas ambientais de forma fácil e rápida é uma funcionalidade-chave, permitindo que os usuários relatem situações que precisam de atenção.

b) Além das denúncias, seria interessante oferecer acesso a informações atualizadas sobre pontos de coleta seletiva, promovendo a separação adequada dos resíduos e incentivando a reciclagem. Também seria relevante fornecer notícias sobre ações ambientais locais, eventos e campanhas de conscientização, para manter a comunidade informada e engajada.

c) Em termos de usabilidade, é importante que a interface do QuixAlert seja intuitiva e de fácil uso. Os usuários devem ter acesso rápido às funcionalidades principais, com opções claras e bem organizadas. Além disso, recursos visuais, como mapas interativos e gráficos, podem ajudar a visualizar a situação ambiental da região e facilitar a compreensão dos dados.

Pergunta 4:

a) Uma necessidade essencial para o QuixAlert é que haja um sistema de acompanhamento e resolução das denúncias feitas pelos usuários. É importante que a comunidade se sinta ouvida e veja resultados concretos a partir das denúncias realizadas.

b) Em termos de prioridades, a implementação de um sistema de notificações que alerte os usuários sobre o andamento e a resolução dos problemas relatados seria muito relevante. Dessa forma, a comunidade se manterá informada sobre as ações tomadas e poderá perceber o impacto positivo de suas denúncias.

Requisitos Funcionais

RF1: Cadastro de Usuário

- O sistema deve permitir o cadastro de novos usuários, solicitando informações como nome, e-mail e senha.
- O sistema deve validar os dados fornecidos durante o cadastro para garantir sua integridade e consistência.

RF2: Login de Usuário

- O sistema deve permitir que os usuários façam login utilizando suas credenciais (e-mail e senha) cadastradas.
- O sistema deve autenticar as credenciais fornecidas e permitir o acesso aos recursos do QuixAlert.

RF3: Denúncia de Problemas Ambientais

- O sistema deve fornecer a funcionalidade de denúncia de problemas ambientais, permitindo que os usuários relatem questões relevantes.
- O sistema deve permitir que os usuários insiram informações como localização exata do problema, descrição detalhada e, se possível, anexem fotos como evidência.

RF4: Acesso a Informações sobre Coleta Seletiva
- O sistema deve fornecer informações atualizadas sobre os pontos de coleta seletiva em Quixadá, permitindo que os usuários saibam onde descartar corretamente seus resíduos.

RF5: Acesso a Notícias e Atualidades Ambientais
- O sistema deve fornecer acesso a notícias e informações atualizadas sobre questões ambientais em Quixadá e região.
- O sistema deve permitir que os usuários estejam informados sobre desafios, avanços e iniciativas relacionadas à proteção ambiental.

RF6: Acompanhamento de Denúncias
- O sistema deve permitir que os usuários acompanhem o status e o progresso da resolução das denúncias feitas por eles.
- O sistema deve fornecer atualizações e informações sobre as ações tomadas pelas autoridades competentes em relação às denúncias.

RF7: Denúncia de Maus-Tratos a Animais
- O sistema deve fornecer a funcionalidade de denúncia de casos de maus-tratos a animais, permitindo que os usuários relatem essas situações.
- O sistema deve permitir que os usuários insiram informações relevantes, como localização do ocorrido, descrição detalhada do caso e, se possível, anexem fotos ou vídeos como evidência.

RF8: Acesso a Informações sobre Adoção de Animais
- O sistema deve fornecer informações atualizadas sobre animais disponíveis para adoção em abrigos ou instituições parceiras.
- O sistema deve permitir que os usuários consultem informações sobre os animais, como espécie, idade, temperamento e requisitos de adoção.

RF9: Realizar Pedidos de Adoção
- O sistema deve permitir que os usuários interessados em adotar um animal façam um pedido de adoção através do aplicativo.
- O sistema deve registrar e encaminhar os pedidos de adoção para as instituições responsáveis, facilitando o processo de adoção de animais.

RF10: Acompanhamento do Processo de Adoção

- O sistema deve permitir que os usuários acompanhem o status e o progresso de seus pedidos de adoção.
- O sistema deve fornecer informações sobre as etapas do processo, como avaliação do perfil do adotante, visitas prévias e agendamento da adoção.

RF11: Solicitação de Licenciamento Ambiental

- O sistema deve permitir que os usuários solicitem licenciamento ambiental para suas atividades ou empreendimentos.
- O sistema deve guiar os usuários por um processo claro e intuitivo, coletando as informações necessárias para a solicitação.

RF12: Orientações sobre Licenciamento Ambiental

- O sistema deve fornecer orientações claras e informações atualizadas sobre o processo de licenciamento ambiental.
- O sistema deve explicar os requisitos legais, documentos necessários e procedimentos a serem seguidos para obter o licenciamento.

RF13: Acompanhamento do Status de Licenciamento

- O sistema deve permitir que os usuários acompanhem o status de suas solicitações de licenciamento ambiental.
- O sistema deve fornecer atualizações sobre as etapas do processo, como análise, parecer técnico e emissão do licenciamento.

RF14: Notificações sobre Atualizações do Processo de Licenciamento

- O sistema deve enviar notificações aos usuários sobre atualizações relevantes relacionadas às suas solicitações de licenciamento ambiental.
- O sistema deve informar sobre a conclusão de etapas, solicitação de documentos adicionais ou qualquer outra informação relevante.

RF15: Visualização de Notícias

- O sistema deve permitir que os usuários visualizem as notícias listadas na HomePage de maneira resumida.
- O sistema deve oferecer uma visualização expandida de notícias para acesso completo ao conteúdo e detalhes.

RF16: Visualização de Lista de Animais com Filtros

- O sistema deve permitir que os usuários visualizem uma lista de animais com opções de filtro, como espécie, idade, porte e localização.
- O sistema deve atualizar dinamicamente a lista de animais com base nos filtros selecionados.

RF17: Visualização Detalhada dos Animais

- O sistema deve permitir que os usuários acessem informações detalhadas sobre cada animal, incluindo fotos, histórico médico e requisitos para adoção.
- O sistema deve exibir informações adicionais relevantes, como temperamento e compatibilidade com crianças ou outros animais.

RF18: Fluxo de Adoção de Animais

- O sistema deve guiar os usuários pelo fluxo de adoção, permitindo realizar pedidos de adoção e acompanhar o status até a finalização do processo.
- O sistema deve coletar informações dos usuários para avaliação do perfil de adotante e fornecer as próximas etapas do processo de adoção

RF19: Contatos de Emergência

- O sistema deve listar contatos de emergência para situações relacionadas a problemas ambientais ou proteção animal, permitindo aos usuários uma comunicação rápida em casos urgentes.

RF20: Notificações Gerais e Personalizadas

- O sistema deve enviar notificações gerais sobre alertas e atualizações do aplicativo.
- O sistema deve permitir que os usuários personalizem as notificações de acordo com suas preferências, como tipo de alerta (ex.: adoção, licenciamento, denúncias).

RF21: Edição de Perfil e Preferências

- O sistema deve permitir que os usuários editem seu perfil, atualizando dados pessoais e configurações de preferência, como notificações e interesses.

RF22: Doações

- O sistema deve permitir que os usuários realizem doações financeiras ou de itens específicos, detalhando o processo e listando as necessidades atuais das organizações parceiras.

RF23: Pesquisar no App

- O sistema deve oferecer uma funcionalidade de pesquisa global que permita aos usuários encontrar rapidamente notícias, animais para adoção, ou informações sobre licenciamento e denúncias.

- A pesquisa deve considerar palavras-chave e filtros relevantes para facilitar o acesso rápido às informações.

RF24: Estilização e Configurações do App

- O sistema deve permitir que os usuários personalizem a aparência do app, incluindo temas de cor, layout, e outras configurações visuais para melhor acessibilidade.

RF25: Tela de Ajuda com Vídeos Explicativos

- O sistema deve fornecer uma tela de ajuda com vídeos que orientem os usuários sobre as principais funcionalidades do app, como o fluxo de denúncias, adoção e configurações.

Requisitos Não Funcionais

Requisitos não funcionais são critérios que definem as características e propriedades do sistema, além de seu comportamento funcional. Eles se concentram em aspectos como desempenho, segurança, usabilidade, confiabilidade e outros atributos não diretamente relacionados às funcionalidades específicas do sistema. Esses requisitos abordam a qualidade do sistema e suas restrições, buscando garantir que o sistema atenda às expectativas e necessidades dos usuários em termos de desempenho, segurança, experiência do usuário e outros aspectos importantes para sua efetividade e sucesso. Listamos a seguir os principais requisitos funcionais do QuixAlert!

1. **Desempenho:**
    - O sistema deve ser responsivo e fornecer uma experiência de usuário fluida.
    - O tempo de resposta para a confirmação de denúncias e notificações deve ser rápido.
    - Os mapas e recursos interativos devem carregar rapidamente e sem interrupções.

2. **Segurança:**
    - Todas as informações dos usuários, incluindo denúncias, devem ser protegidas por medidas de segurança adequadas.

- Os dados pessoais e confidenciais devem ser criptografados e armazenados de forma segura.
- As comunicações entre o aplicativo e os servidores devem ser protegidas por criptografia.

3. **Usabilidade:**
   - O aplicativo deve ser intuitivo e fácil de usar, mesmo para usuários com pouca experiência tecnológica.
   - Os formulários de denúncia e outros recursos devem ter um design claro e simples, com instruções claras para os usuários.
   - A navegação no aplicativo deve ser lógica e coesa, facilitando a localização das informações e funcionalidades desejadas.

4. **Confiabilidade:**
   - O sistema deve ser confiável e estar disponível para uso na maior parte do tempo.
   - As denúncias e outras informações enviadas pelos usuários devem ser registradas de forma precisa e consistente.
   - O sistema deve ser capaz de lidar com um grande volume de denúncias e notificações sem falhas ou perda de dados.

5. **Escalabilidade:**
   - O sistema deve ser capaz de lidar com um aumento no número de usuários e na atividade do aplicativo.
   - Os servidores e infraestrutura do sistema devem ser dimensionados adequadamente para suportar um crescimento futuro.

6. **Compatibilidade:**
   - O aplicativo deve ser compatível com diferentes dispositivos, sistemas operacionais e navegadores.
   - O design responsivo deve garantir que o aplicativo funcione bem em dispositivos móveis, tablets e computadores.

7. **Acessibilidade:**
   - O sistema deve ser acessível para pessoas com deficiências visuais, auditivas ou motoras.
   - Deve ser fornecido suporte para recursos de acessibilidade, como leitores de tela e opções de tamanho de fonte ajustável.

8. **Manutenibilidade:**

- O código-fonte e a infraestrutura do sistema devem ser bem documentados para facilitar a manutenção futura.
- Deve ser seguido um padrão de desenvolvimento modular e boas práticas de programação para facilitar a adição de novos recursos e a correção de possíveis problemas.

9 QUESTIONÁRIO *SUS*!

Documento contendo o questionário SUS aplicado nos especialistas para testes de usabilidade.

Questionário SUS – Avaliação da Documentação

Por favor, indique seu nível de concordância com as afirmativas abaixo, utilizando a escala de 1 a 5:

1 – Discordo totalmente
2 – Discordo
3 – Neutro
4 – Concordo
5 – Concordo totalmente

Perguntas:

1. Eu acho que gostaria de consultar esta documentação com frequência.
   □ 1 □ 2 □ 3 □ 4 □ 5

2. Eu acho esta documentação desnecessariamente complexa.
   □ 1 □ 2 □ 3 □ 4 □ 5

3. Eu achei a documentação fácil de usar.
   □ 1 □ 2 □ 3 □ 4 □ 5

4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para entender esta documentação.
   □ 1 □ 2 □ 3 □ 4 □ 5

5. Eu acho que as várias seções desta documentação estão muito bem integradas.
   □ 1 □ 2 □ 3 □ 4 □ 5

6. Eu acho que a documentação apresenta muita inconsistência.
   □ 1 □ 2 □ 3 □ 4 □ 5

7. Eu imagino que as pessoas aprenderão como encontrar o que precisam nesta documentação rapidamente.

□ 1 □ 2 □ 3 □ 4 □ 5

8. Eu achei a documentação atrapalhada/confusa de usar.
   □ 1 □ 2 □ 3 □ 4 □ 5

9. Eu me senti confiante ao consultar a documentação.
   □ 1 □ 2 □ 3 □ 4 □ 5

10. Eu precisei aprender várias coisas novas antes de conseguir encontrar o que procurava na documentação.
    □ 1 □ 2 □ 3 □ 4 □ 5

10 ROTEIRO DE TAREFAS PARA AVALIAÇÃO DE USABILIDADE DA VERSÃO EVOLUÍDA DO *QUIXALERT!*

O presente apêndice apresenta o **Roteiro de Tarefas** utilizado na etapa de validação de usabilidade do *QuixAlert!*. Este documento tem como objetivo orientar a condução dos testes com os participantes, garantindo consistência e reprodutibilidade na coleta de dados.

Roteiro de Tarefas para Avaliação de Usabilidade da Versão Evoluída do QuixAlert!

Objetivo: O presente roteiro tem como finalidade guiar a aplicação do Teste de Usabilidade, visando avaliar a eficiência, a eficácia e a satisfação do usuário na interação com as funcionalidades da versão evoluída do QuixAlert!. O foco está em verificar o desempenho das funcionalidades antigas (como teste de regressão após a modularização) e validar as novas *features* implementadas.

Metodologia: As sessões de teste serão guiadas pelo observador, que aplicará o protocolo *think-aloud* para registrar verbalizações dos participantes, além de registrar as métricas de tempo e a taxa de sucesso/erro de cada tarefa.

Instrução ao Participante: "Você está utilizando a nova versão do aplicativo **QuixAlert!**. Por favor, realize as tarefas a seguir de forma sequencial. Durante a execução, lembre-se de verbalizar seus pensamentos e sentimentos em voz alta para que possamos entender sua experiência (Protocolo *Think-Aloud*)."

I. Fluxo Central e Funções Antigas (Teste de Regressão)

O primeiro bloco de tarefas visa garantir que as funcionalidades centrais do sistema, que existiam na versão monolítica, continuam operando de forma correta e intuitiva após a refatoração e a migração para a nova arquitetura modular.

1. **Tarefa 1 (Registro de Denúncia):** Simule o registro de uma denúncia ambiental completa sobre o **descarte irregular de lixo** em uma área urbana. A tarefa deve incluir a inserção do título, descrição, endereço da ocorrência e o envio de uma foto (simulada).
2. **Tarefa 2 (Consumo de Informação):** Acesse a seção de Notícias e procure por informações ou comunicados recentes da **Autarquia Municipal do Meio Ambiente (AMMA)**, verificando a clareza e a facilidade de navegação entre os itens.

II. Expansão da Cidadania Digital (Novas Funcionalidades Estratégicas)

Este bloco de tarefas avalia o impacto das novas funcionalidades de comunicação em tempo real e *feedback*, que comprovam a escalabilidade da nova arquitetura e o aprimoramento da governança colaborativa.

3. **Tarefa 3 (Comunicação Direta via Chat):** Após simular a denúncia na Tarefa 1, encontre o canal de suporte ou comunicação e **inicie um Chat** com a equipe da AMMA para tirar uma dúvida sobre o **prazo de atendimento** da sua ocorrência.
4. **Tarefa 4 (Acompanhamento e Notificações):** Simule a alteração do *status* da sua denúncia (por exemplo, de "Em Análise" para "**Em Atendimento**") e verifique se o novo **Módulo de Notificações** exibiu o alerta corretamente. A tarefa é considerada bem-sucedida se o usuário compreender a mudança de *status* por meio da notificação.

III. Módulo de Adoção e Sustentabilidade (Novas Features de Valor)

O terceiro bloco foca nas melhorias do Módulo de Adoção, que representam a expansão dos serviços de bem-estar animal e a inclusão de um novo fluxo de sustentabilidade.

5. **Tarefa 5 (Filtros e Detalhes): Aplique o filtro** do módulo de Adoção para visualizar apenas animais de **pequeno porte** e, em seguida, acesse a página de detalhes de um animal para interagir com a **Galeria de Mídias Detalhada**.
6. **Tarefa 6 (Fluxo de Doação):** Localize a nova funcionalidade e **simule o fluxo de Doação Financeira**, escolhendo um valor e prosseguindo até a tela final de pagamento (sem efetuar a transação real).

IV. Avaliação da Usabilidade e Navegabilidade

A tarefa final visa validar a usabilidade e a arquitetura de informação do aplicativo após as intervenções.

8. **Tarefa 7 (Navegação e Descoberta):** Partindo da tela inicial (Home), navegue até o Módulo de Notícias, em seguida vá para a seção de Detalhes de um animal, e retorne à tela principal. O participante deve demonstrar facilidade ao encontrar os diferentes módulos e ao usar os botões de retorno (gestos) do aplicativo.