UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VINICIUS DOMINGUES PEREIRA

BAIRROFORTE: UM APLICATIVO COLABORATIVO PARA MONITORAMENTO DE SEGURANÇA EM TEMPO REAL ENTRE MORADORES E COMERCIANTES

GUARAPUAVA

2025

VINICIUS DOMINGUES PEREIRA

BAIRROFORTE: UM APLICATIVO COLABORATIVO PARA MONITORAMENTO DE SEGURANÇA EM TEMPO REAL ENTRE MORADORES E COMERCIANTES

BairroForte: A collaborative app for real-time security monitoring between residents and merchants

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Roni Fabio Banaszewski

GUARAPUAVA

2025

VINICIUS DOMINGUES PEREIRA

BAIRROFORTE: UM APLICATIVO COLABORATIVO PARA MONITORAMENTO DE SEGURANÇA EM TEMPO REAL ENTRE MORADORES E COMERCIANTES

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 04/dezembro/2025

Prof. Roni Fabio Banaszewski
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

Prof. Emerson André Fedechen
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

Prof. Luciano Ogiboski
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

GUARAPUAVA

2025

AGRADECIMENTOS

Primeiramente, minha gratidão a toda minha família por todo o apoio durante esta jornada. Um agradecimento especial ao meu pai, que sempre me incentivou em todos os momentos e é meu grande exemplo de vida.

Ao meu orientador, Prof. Dr. Roni Fabio Banaszewski, agradeço imensamente pela orientação, paciência e por todo o conhecimento compartilhado, essenciais para a concretização deste trabalho.

À minha namorada, sou grato pelo incentivo constante e por não me deixar desistir durante os desafios encontrados pelo caminho.

Agradeço aos meus amigos, que me apoiaram e tornaram este trajeto mais leve.

Por fim, agradeço a todos que de alguma maneira me incentivaram e deixaram o processo mais divertido.

RESUMO

O presente trabalho aborda o desenvolvimento do BairroForte, um aplicativo de segurança colaborativa projetado para empoderar comunidades urbanas e suburbanas na prevenção de crimes e fortalecimento da sensação de segurança. A justificativa para o projeto está na crescente preocupação com a segurança pública e a falta de ferramentas eficazes para comunicação e ação comunitária em áreas vulneráveis. O objetivo principal do aplicativo é conectar moradores e comerciantes, permitindo o compartilhamento de ocorrências em tempo real, a visualização de câmeras, o envio de alertas e a formação de grupos de bairro para ações preventivas. O projeto inclui a personalização das notificações, permitindo que os usuários definam preferências como categorias de incidentes, raio de alcance e horários. Como resultado, foi desenvolvido um *Minimum Viable Product* (Produto Mínimo Viável) (MVP) funcional que demonstra potencial para gerar maior engajamento comunitário, auxiliar na redução de incidentes e aumentar a confiança entre os vizinhos. O BairroForte destaca-se por oferecer funcionalidades como mapas de calor de criminalidade, controle sobre as configurações de privacidade e um sistema de alertas geolocalizados. Conclui-se que o BairroForte tem potencial para transformar a forma como as comunidades lidam com a segurança, promovendo um ambiente mais conectado, colaborativo e seguro.

Palavras-chave: segurança colaborativa; aplicativo móvel; prevenção de crimes; geolocalização; .

ABSTRACT

This study addresses the development of BairroForte, a collaborative security application designed to empower urban and suburban communities in crime prevention and enhancing the sense of safety.The justification for the project lies in the growing concern over public safety and the lack of effective tools for communication and community action in vulnerable areas. The primary objective of the application was to connect residents and business owners, enabling the sharing of real-time incidents, the visualization of cameras, sending alerts, and forming neighborhood groups for preventive actions. The project implemented notification customization, allowing users to set preferences such as incident categories, coverage radius, and schedules. As a result, a functional MVP was delivered, which demonstrates the potential to generate greater community engagement, assist in reducing incidents, and increase trust among neighbors. BairroForte stands out by offering features like crime heat maps, control over privacy settings, and a geolocated alert system.It is concluded that BairroForte has the potential to transform how communities address security, fostering a more connected, collaborative, and safe environment.

Keywords: collaborative security; mobile app; crime prevention; geolocation; .

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

ACR	*Azure Container Registry* (Registro de Contêineres do Azure)
API	*Application Programming Interface* (Interface de Programação de Aplicações)
APNs	*Apple Push Notification Service* (Serviço de Notificações Push da Apple)
FAQ	*Frequently Asked Questions* (Perguntas Frequentes)
FCM	*Firebase Cloud Messaging* (Serviço de Mensageria em Nuvem do Firebase)
IBGE	Instituto Brasileiro de Geografia e Estatística
MVP	*Minimum Viable Product* (Produto Mínimo Viável)
PNAD	*National Continuous Household Sample Survey* (Pesquisa Nacional por Amostra de Domicílios Contínua)
S.O.S.	*Save Our Souls* (Código Universal de Socorro)
SGBD	*Database Management System* (Sistema de Gerenciamento de Banco de Dados)
SMS	*Short Message Service* (Serviço de Mensagens Curtas)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)

SUMÁRIO

1 INTRODUÇÃO

A estabilidade pública é uma preocupação crescente em muitas regiões urbanas e suburbanas, especialmente em áreas onde a criminalidade tem aumentado significativamente nos últimos anos. A sensação de insegurança afeta não apenas os moradores, mas também os pequenos comerciantes, que, muitas vezes, se tornam alvos de crimes como roubos e furtos.

De acordo com a *National Continuous Household Sample Survey* (Pesquisa Nacional por Amostra de Domicílios Contínua) (PNAD) do Instituto Brasileiro de Geografia e Estatística (IBGE), em 2021, 60,4% dos moradores do Norte do Brasil se sentem inseguros ao andar sozinhos à noite nos arredores de seus domicílios, enquanto no Sul essa porcentagem é de 38,1%, e no Brasil apenas 48,3% se sentem seguros (IBGE, 2021). Esses dados evidenciam a necessidade de soluções que aumentem a sensação de segurança entre os moradores e até mesmo comerciantes locais, demonstrando a base para o desenvolvimento de soluções que promovam proteção colaborativa.

O aspecto da confiança pública tem um profundo impacto na qualidade de vida e na vitalidade econômica das comunidades. Quando ocorrem aumentos nos índices de criminalidade, as pessoas tendem a modificar suas rotinas e, então, evitam determinada região e/ou horários, limitando o fluxo de indivíduos e afetando o comércio local. Esse quadro não apenas limita a chegada de novos negócios, como também compromete a confiança das pessoas com seu espaço de vida, o que limita o crescimento econômico e o desenvolvimento sustentável da região.

Nas situações cotidianas, como os trajetos para o trabalho, a maioria dos moradores desconhece, mediante informações em tempo real, os pontos mais críticos da segurança. Sem que exista uma forma clara e atualizada sobre quais ruas e/ou bairros apresentam riscos maiores, acabam, involuntariamente, se colocando em situações de risco. Essa ausência de informação favorece uma maior vulnerabilidade e uma maior sensação de insegurança, gerando um círculo no qual a própria comunidade vai se sentindo acuada e receosa em relação ao seu território.

Esse cenário é agravado pela ausência de um sistema abrangente de vigilância, especialmente nas áreas periféricas, o que intensifica a sensação de insegurança, dado que a cobertura de câmeras de monitoramento é escassa. Um exemplo é o caso de roubos em ruas desprovidas de monitoramento, sem registros visuais que facilitem a identificação dos responsáveis. Frequentemente, esses incidentes acabam no campo da impunidade, perpetuando o sentimento de insegurança entre os moradores. Essa realidade reforça a urgência na implementação de soluções eficazes que respondam a essas necessidades (MIYAZAKI, 2024).

O desenvolvimento de tecnologias para a circulação de informações de segurança em tempo real, pode transformar a experiência da vida e locomoção nas cidades. A criação de ferramentas intuitivas de geolocalização e de comunicação instantânea, por exemplo, poderia funcionar como uma rede protetora, por onde as pessoas normatizariam e compartilhariam informações sobre as áreas com risco. Com uma plataforma tecnológica acessível, as comuni-

dades ganhariam uma maneira mais imediata de observar seu meio e agir colaborativamente pela segurança, isto é, proporcionaria uma organização espontânea.

A implementação de tal solução funcionaria como uma aliada preventiva, onde a intensificação da vigilância e da organização do bairro não apenas aumentaria a sensação de segurança, mas, sobretudo, ajudaria desestimular atividades delituosas, promovendo, assim, o sentimento de união dos moradores. Dessa forma, a tecnologia se apresentaria como uma conexão essencial entre a necessidade de segurança e a vontade de uma convivência mais tranquila e confiante.

1.1 Justificativa

Em um momento em que os perigos estão sempre presentes, a necessidade de refletir sobre maneiras de tornar as regiões mais seguras transforma-se em um desafio de emergência. Diariamente, os cidadãos modificam seus comportamentos e evitam alguns lugares, mas a falta de informação precisa sobre os riscos reais diminui a atos de intuição. Em vez de gerar uma segurança edificadora, eles concedem acesso a um sentimento de incerteza, mantendo o público em uma posição defensiva, criando barreiras para o comércio local, que desempenha um papel crítico em um clima de confiança.

É nesse contexto que aparece a necessidade de uma solução prática e simples, assegurando que as populações possam obter dados confiáveis sobre a segurança no seu entorno e, com isto, tornar o cotidiano mais seguro e menos sujeito a imprevistos.

A possibilidade de colaborar de forma organizada por intermédio de uma aplicativo nóvel disponível permitiria dar maior segurança para quem mora e trabalha nessas regiões. Bairros que realizam vigilância e comunicação estruturada geralmente inibem os ilícitos, uma vez que o criminoso é dissuadido pela ideia de que suas ações podem ser observadas e denunciadas facilmente (ROSáRIO, 2025). Assim, a tecnologia não é apenas um instrumento para fomentar a segurança, mas sim uma base para construir os ambientes urbanos onde proteção e qualidade de vida são concomitantes, prestando atendimento para o desenvolvimento de comunidades mais coesas e resilientes.

1.2 Objetivos

Nesta seção serão apresentados o objetivo geral e objetivos específicos do presente trabalho.

1.2.1 Objetivo geral

Desenvolver um aplicativo nomeado como BairroForte para permitir a troca de informações ajudando na segurança local e comercial a partir da colaboração entre comerciantes e moradores, permitindo a troca de informações em tempo real sobre incidentes locais e promovendo ações de prevenção contra a criminalidade.

1.2.2 Objetivos específicos

- Desenvolver um aplicativo que utilize geolocalização para monitoramento de ocorrências de segurança.
- Implementar funcionalidades de envio e recebimento de alertas em tempo real para incidentes promovendo a integração entre moradores e comerciantes.
- Desenvolver funcionalidades que permita aos usuários, com o devido consentimento, registrar e compartilhar informações como câmeras de segurança, compartilhando a localização de câmeras.
- Desenvolver um sistema de notificações personalizáveis, mantendo a segurança dos dados dos usuários e a comunicação de informações pertinentes e customizadas a cada um deles.
- Implementar a configuração de informações no aplicativo mantendo a segurança e integridade dos usuários.

1.3 Organização do trabalho

Esta seção apresenta a organização geral do trabalho. O Capítulo 2 apresenta uma análise de aplicativos semelhantes existentes no mercado, destacando suas funcionalidades e limitações. Por sua vez, o Capítulo 3 apresenta e destaca toda a base tecnológica utilizada para o desenvolvimento do sistema. Em seguida, no Capítulo 4 são detalhados os materiais e métodos adotados para o desenvolvimento do projeto, com base no referencial teórico previamente abordado. O Capítulo 5 define o escopo do sistema, passando por artefatos e modelagens essenciais ao seu desenvolvimento. O Capítulo 6 detalha todo o processo de desenvolvimento e implementação do aplicativo, desde a configuração do ambiente até a entrega final das funcionalidades. Por fim o Capítulo 7 sintetiza e conclui sobre todo o processo ao longo do projeto e apresenta perspectiva para trabalhos futuros.

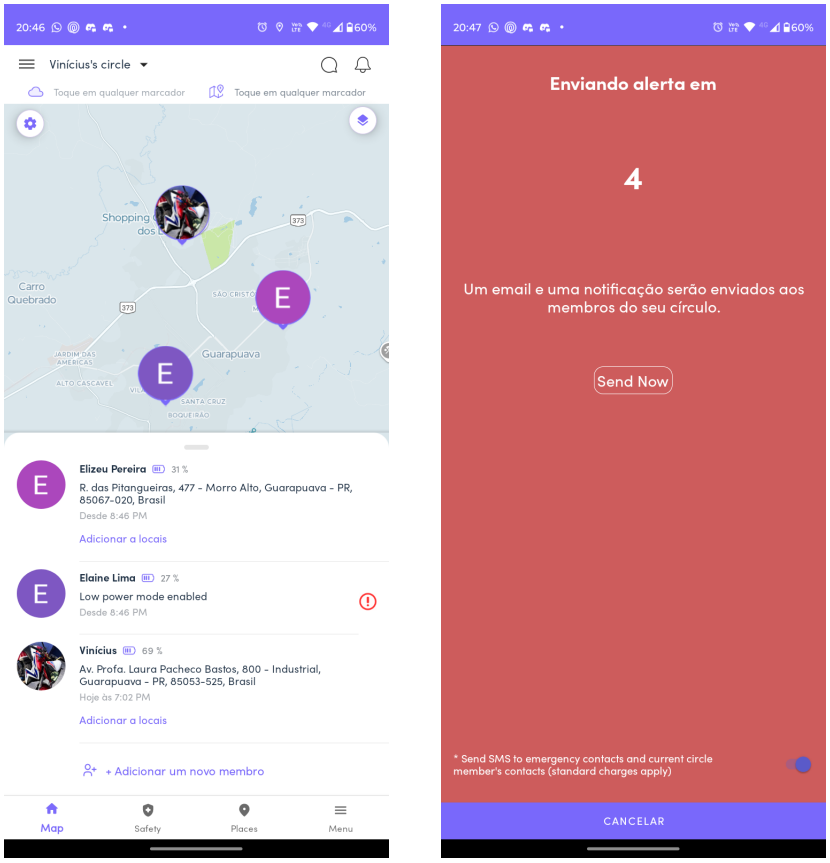
2 TRABALHOS RELACIONADOS

Atualmente, existem diversos sistemas e aplicativos que têm como objetivo integrar a vizinhança, promovendo a troca de informações e a colaboração entre moradores. Esses serviços, que emergem da necessidade de segurança e interação social, aproveitam a tecnologia para fortalecer os laços comunitários e criar ambientes mais seguros

2.1 Family360

O *Family360* se apresenta como uma escolha para quem deseja monitorar sua família auxiliando na segurança, como mostra a Figura 1a, que ilustra a visualização do círculo familiar no aplicativo. Ele inclui muitas funcionalidades voltadas para monitorar em tempo real onde estão todos os integrantes da família. Um dos recursos mais interessantes do *Family360* é o "*Watch Over Me*". Ele oferece a possibilidade de solicitar que outra pessoa acompanhe o trajeto do usuário em situações consideradas arriscadas, como no retorno para casa à noite. Essa funcionalidade gera uma sensação adicional de segurança para ambas as partes — tanto para quem está sendo monitorado quanto para quem realiza a vigilância (FAMILY360, 2024).

Outra funcionalidade disponível em situações de emergência é o envio de alertas por e-mail. Ao acionar o botão *Send Emergency Alert* (enviar alerta de emergência), o sistema encaminha automaticamente uma mensagem contendo a localização exata do usuário, garantindo que o aviso seja recebido mesmo quando os familiares do círculo não estiverem com notificações ativas. Isso que garante que o alerta será recebido, mesmo se alguém estiver com as notificações desligados ou sem acesso ao celular no momento como na Figura 1b.



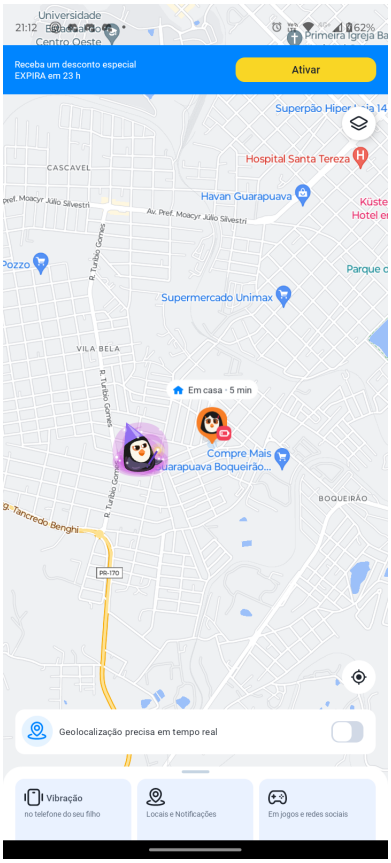
(a) Visualização do círculo familiar na interface do Family360

(b) Notificação de emergência no Family360

Figura 1 – Family360

2.2 Find My Kids

Outro aplicativo para o monitoramento familiar é o *Find My Kids*, desenvolvido especificamente para pais que desejam acompanhar, em tempo real, a localização de seus filhos, conforme ilustrado na Figura 2a. Além do rastreamento de localização, o aplicativo apresenta diversas funcionalidades voltadas à segurança e ao monitoramento infantil, como mostrado na Figura 2b, incluindo o alerta de localização e o registro de movimentação. Entre os recursos disponíveis, destaca-se o “Escute o que está acontecendo”, que possibilita aos responsáveis ouvir o som ambiente nas proximidades do dispositivo da criança. Essa funcionalidade auxilia na identificação de situações de risco e contribui para uma resposta mais rápida em casos de emergência. Outra função relevante é a criação de zonas seguras, como a escola ou a residência, que emite notificações automáticas sempre que a criança entra ou sai desses locais (FINDMYKIDS, 2024).



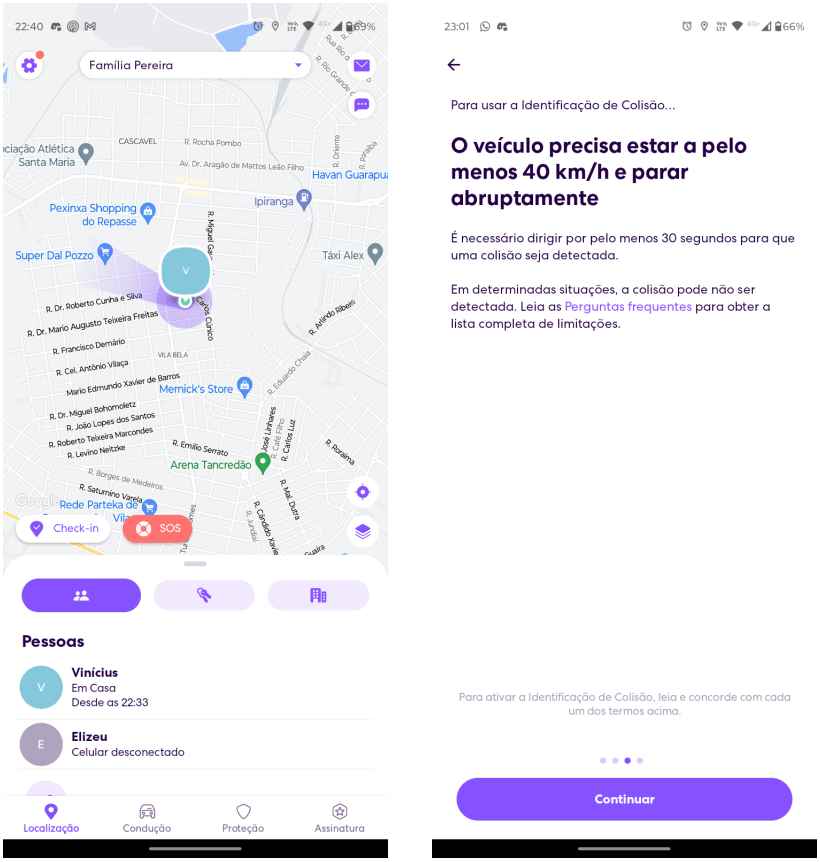
(a) Tela de localização em tempo real no aplicativo Find My Kids

(b) Tela de informações e atividades do filho no aplicativo Find My Kids

Figura 2 – Find My Kids

2.3 Life360

O Life360 é um aplicativo de rastreamento de celulares com mais de 70 milhões de usuários (LIFE360, 2024), amplamente utilizado para monitorar a localização de filhos e parentes. Embora o rastreamento em tempo real seja seu principal objetivo, o aplicativo oferece diversas funcionalidades adicionais, como a visualização do nível de bateria e a possibilidade de enviar alertas para informar quando um membro da família chega a um local específico, tal como ilustrado na Figura 3a (UOL, 2023). Outro recurso interessante é o sistema de detecção de colisão, que emite alertas em caso de paradas bruscas, proporcionando maior segurança durante o acompanhamento de trajetos, conforme a Figura 3b.



(a) Tela inicial de monitoramento familiar no aplicativo Life360

(b) Notificação de colisão no aplicativo Life360

Figura 3 – Family360

2.4 Considerações

Os aplicativos analisados, como *Family360*, *Find My Kids* e *Life360*, oferecem diversas funcionalidades focadas na segurança e no bem-estar familiar, como rastreamento em tempo

real, alertas de emergência e até detecção de colisões. Esses recursos são eficazes para o monitoramento familiar, mas apresentam limitações quando pensamos em uma abordagem mais ampla e comunitária, que envolva não só as famílias, mas também comerciantes e moradores em geral.

O BairroForte surge justamente para preencher essa lacuna, com uma proposta que vai além do monitoramento individual. O objetivo é criar uma solução colaborativa de segurança comunitária, permitindo que moradores e comerciantes compartilhem informações em tempo real sobre incidentes, ajudando a promover um ambiente mais seguro para todos.

O estudo dos aplicativos existentes foi fundamental para identificar funcionalidades essenciais, como geolocalização e alertas, que serão adaptadas e ampliadas para uma abordagem mais abrangente de segurança comunitária no BairroForte. Assim, essa análise não só serviu como ponto de partida para o desenvolvimento do projeto, mas também ajudou a identificar onde o BairroForte pode inovar e trazer benefícios reais para a comunidade.

3 REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos e as tecnologias utilizadas no desenvolvimento de aplicações modernas, destacando suas funcionalidades e contribuições para garantir segurança, eficiência e comunicação em tempo real. A estrutura de um sistema desse tipo geralmente se organiza em três camadas principais: *frontend*, também denominado lado cliente, responsável pela interface e interação com o usuário; *backend*, ou lado servidor, encarregado do processamento e da lógica do sistema; e banco de dados, voltado ao armazenamento e à gestão das informações. Essa arquitetura permite a escalabilidade, a segurança e a facilidade de manutenção da aplicação.

3.1 Tecnologias do lado cliente

As tecnologias do lado cliente englobam os recursos e *frameworks* responsáveis pela interface e interação direta com o usuário. Essa camada trata da apresentação das informações, da experiência visual e da comunicação com o servidor, utilizando abordagens modernas que priorizam responsividade, desempenho e usabilidade. Entre as soluções mais relevantes nesse contexto estão plataformas que permitem o desenvolvimento multiplataforma, como o Flutter e o *FlutterFlow*.

3.1.1 Flutter

O Flutter é um *framework* de código aberto desenvolvido pelo Google para a criação de interfaces visuais interativas e responsivas. Utilizando a linguagem *Dart*, permite o desenvolvimento de aplicativos nativos para dispositivos móveis, web e *desktop* a partir de uma única base de código. Seu conjunto abrangente de *widgets* e componentes facilita a construção de interfaces modernas e de alto desempenho, garantindo consistência e fluidez na experiência do usuário (FLUTTER, 2024).

3.1.2 FlutterFlow

O *FlutterFlow* é uma plataforma de desenvolvimento visual baseada no Flutter, permitindo a criação de aplicativos móveis sem a necessidade de codificação manual. Sua interface de arrastar e soltar possibilita a rápida prototipagem e implementação de soluções, além de oferecer integração com bancos de dados em tempo real e autenticação de usuários. O código gerado pelo *FlutterFlow* pode ser exportado para personalizações avançadas no Flutter, tornando-se uma opção flexível para desenvolvedores (FLUTTERFLOW, 2025).

3.2 Tecnologias do lado servidor

As tecnologias do lado servidor correspondem aos componentes que processam as informações e executam as regras de negócio de uma aplicação. Essa camada é responsável por gerenciar requisições, acessar o banco de dados e garantir a integridade e a segurança dos dados transmitidos. Ferramentas como NodeJS, NestJS, *Firebase Cloud Messaging* (Serviço de Mensageria em Nuvem do Firebase) (FCM), Docker e MySQL são amplamente utilizadas no desenvolvimento de sistemas escaláveis e integrados.

3.2.1 NodeJS e NestJS

Node.js é um ambiente de execução que permite a execução de código JavaScript no servidor. Com seu modelo de entrada/saída assíncrono e orientado a eventos, é altamente eficiente para lidar com múltiplas requisições simultâneas. É amplamente utilizado no desenvolvimento de servidores, *Application Programming Interface* (Interface de Programação de Aplicações) (API) e aplicações web escaláveis (NODEJS, 2024). NestJS é um *framework* baseado em Node.js que utiliza TypeScript para estruturar aplicações de *backend* de maneira modular e escalável. Ele incorpora princípios da programação orientada a objetos, funcional e reativa, tornando o desenvolvimento de APIs e microsserviços mais organizado e manutenível (NESTJS, 2024).

3.2.2 Firebase

O *Firebase* é uma plataforma de desenvolvimento de aplicações móveis e web mantida pelo Google, que oferece uma série de serviços integrados voltados à autenticação, armazenamento, análise e comunicação em tempo real. Entre suas principais funcionalidades estão a autenticação de usuários, o envio de notificações *push*, o gerenciamento de bancos de dados em tempo real e o monitoramento de desempenho das aplicações.

O serviço Firebase Authentication fornece mecanismos de autenticação seguros e escaláveis, com suporte a métodos como e-mail e senha, autenticação anônima ou integração com provedores externos, como Google e Facebook. Sua implementação é facilitada por bibliotecas e SDKs disponíveis para diversas plataformas, incluindo o *Flutter*.

Já o FCM é responsável pelo envio de mensagens e notificações *push* entre servidores e dispositivos móveis. O FCM permite a entrega eficiente de alertas em tempo real, sejam eles direcionados a usuários específicos ou distribuídos em massa, tornando-se uma solução amplamente utilizada em sistemas que exigem comunicação instantânea e confiável (GOOGLE, 2025).

3.2.3 Twilio

O *Twilio* é uma plataforma em nuvem voltada à comunicação programável, que oferece APIs para o envio de mensagens de texto *Short Message Service* (Serviço de Mensagens Curtas) (SMS), realização de chamadas de voz e integração com diversos canais de comunicação, como WhatsApp e e-mail. Seu principal objetivo é permitir que aplicações incorporem funcionalidades de comunicação de forma simples, sem a necessidade de infraestrutura telefônica dedicada.

Entre seus serviços mais utilizados estão o *Twilio SMS API*, responsável pelo envio automatizado de mensagens de texto, e o *Twilio Verify*, que fornece autenticação de dois fatores por meio de códigos temporários enviados ao dispositivo do usuário. A plataforma oferece alta disponibilidade e escalabilidade, sendo amplamente empregada em sistemas que necessitam de comunicação confiável em tempo real (Twilio Inc., 2025).

3.2.4 OneSignal

O *OneSignal* é uma plataforma de envio e gerenciamento de notificações *push* multiplataforma, amplamente utilizada em aplicações móveis e *web*. Seu objetivo é facilitar a comunicação direta entre sistemas e usuários por meio de alertas personalizados, permitindo o envio de mensagens em tempo real. A ferramenta oferece uma interface unificada e APIs que simplificam o uso dos serviços FCM e *Apple Push Notification Service* (Serviço de Notificações Push da Apple) (APNs), responsáveis pela entrega efetiva das notificações em dispositivos Android e iOS, respectivamente.

Além disso, o *OneSignal* disponibiliza recursos de segmentação, agendamento e acompanhamento de métricas de envio, permitindo monitorar o desempenho e o engajamento das mensagens. Sua facilidade de integração e suporte a notificações automáticas o tornam uma solução prática e robusta para o gerenciamento de comunicações em sistemas distribuídos (OneSignal, 2025).

3.2.5 Docker

O Docker é uma tecnologia de containerização que possibilita embalar uma aplicação e as suas dependências, em um ambiente isolado conforme apresentado na Figura 4, assegurando que o software opere da mesma forma em qualquer infraestrutura. A tecnologia se utiliza da criação de contêineres que são unidades leves e portáveis, propiciando uma carga maior de eficiência no desenvolvimento, teste e entrega do aplicativo. Com Docker é possível executar múltiplas instâncias de um sistema, sem conflitos, entre as dependências. Dessa forma, a gestão das aplicações torna-se mais ágil e segura (DOCKER, 2025).

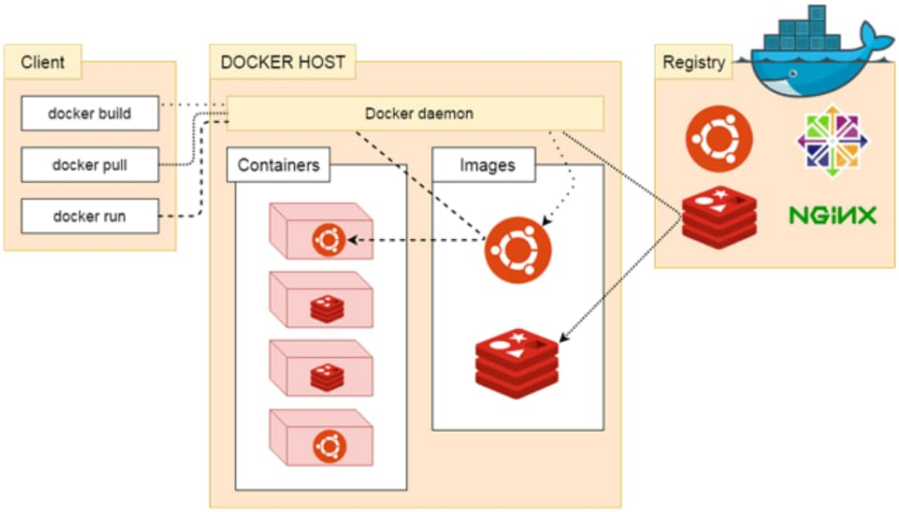


Figura 4 – Funcionamento do Docker
Fonte: (GULIYEVA, 2023).

3.2.6 Microsoft Azure

O *Microsoft Azure* é uma plataforma de computação em nuvem desenvolvida pela Microsoft, que fornece uma ampla gama de serviços voltados à hospedagem, processamento e gerenciamento de aplicações. Entre suas principais funcionalidades estão o provisionamento de máquinas virtuais, o gerenciamento de contêineres e o suporte a bancos de dados e serviços de rede.

O serviço *Azure Container Registry* (Registro de Contêineres do Azure) (ACR) permite armazenar e gerenciar imagens de contêineres *Docker* em um repositório privado, garantindo controle de versão, segurança e facilidade de integração com pipelines de entrega contínua. A utilização dessa plataforma é especialmente vantajosa em projetos que demandam escalabilidade e disponibilidade constantes, possibilitando que aplicações baseadas em contêineres sejam implantadas e gerenciadas de maneira eficiente (Microsoft Corporation, 2025).

3.2.7 MySQL

MySQL caracteriza-se como um *Database Management System* (Sistema de Gerenciamento de Banco de Dados) (SGBD) relacional de código aberto que é utilizado extensivamente para o armazenamento, organização e gerenciamento de grandes volumes de dados. Esse SGBD utiliza a linguagem de manipulação e consulta de dados *Structured Query Language* (Linguagem de Consulta Estruturada) (SQL) sendo muito conhecido por sua confiabilidade, seu alto desempenho e sua escalabilidade. Ele é utilizado extensivamente no ambiente das aplicações Web, nos sistemas empresariais e nos serviços de nuvem devidos à sua compatibilidade

com diversas linguagens de programação e a facilidade de integração que estas diferentes linguagens proporcionam (ORACLE, 2025).

3.3 Processo de desenvolvimento

O desenvolvimento de software envolve diversas etapas, desde a concepção da ideia até a entrega do produto final. Para garantir um fluxo de trabalho eficiente, são adotadas metodologias e práticas que promovem a organização, o acompanhamento do progresso e a colaboração entre os membros da equipe. As ferramentas ágeis, como o Scrum e o Kanban, destacam-se por promover ciclos curtos de desenvolvimento, comunicação contínua e adaptação às mudanças, resultando em maior eficiência e qualidade no produto final.

3.3.1 Metodologia ágil Scrum

O Scrum é um *framework* ágil de gerenciamento de projetos destinado a atingir a entrega incremental e iterativa de produtos conforme ilustrado na Figura 5, dividindo o desenvolvimento em ciclos curtos denominados Sprints, de 2 a 4 semanas. Assim, permite que a equipe possa se adaptar rapidamente a mudanças e melhorar continuamente o produto. O Scrum descreve os papéis essenciais, consistindo no *Product Owner*, que é o responsável por definir as prioridades, no *Scrum Master*, que atua como facilitador, garantindo a aplicação eficaz da metodologia, e por fim, o Time de Desenvolvimento, que é o encarregado por implementar as funcionalidades previstas. Esse *framework* fomenta colaboração, transparência e flexibilidade para garantir entregas frequentes e de alta qualidade (SCRUMGUIDES, 2025).

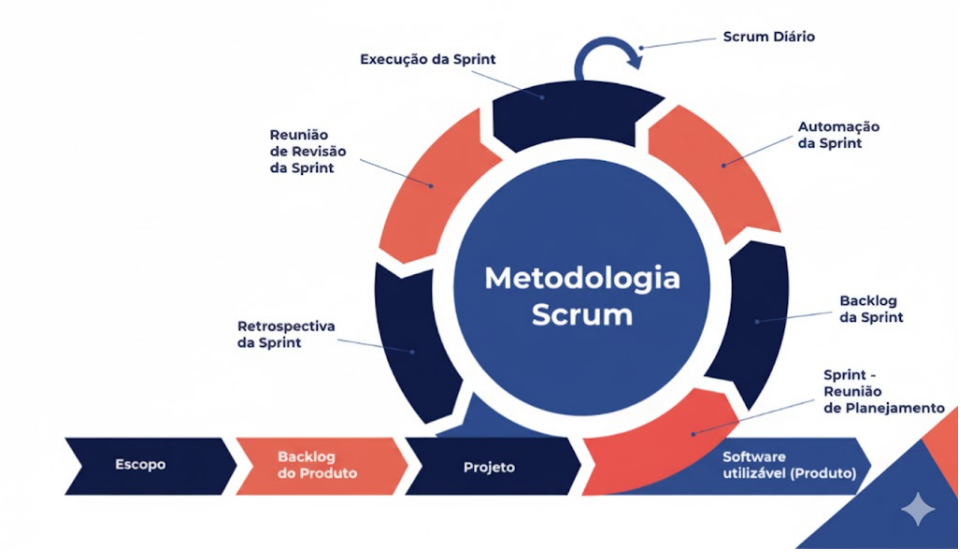


Figura 5 – Dinâmica do método Scrum.
Fonte: (EDUCAçãO, 2024).

3.3.2 Técnica MoSCoW

A técnica MoSCoW é um método de priorização de requisitos utilizado em projetos ágeis para definir o nível de importância de cada funcionalidade no produto. O nome é um acrônimo formado pelas iniciais de quatro categorias: *Must Have* (Deve Ter), *Should Have* (Deveria Ter), *Could Have* (Poderia Ter) e *Won't Have This Time* (Não Terá Desta Vez) (SEBRAE, 2024).

Esse método auxilia na gestão do escopo, permitindo que as entregas iniciais contemplem apenas os requisitos essenciais, enquanto as funcionalidades complementares são planejadas para versões futuras. Sua aplicação é especialmente útil em projetos com recursos limitados ou foco em entregas incrementais, como no desenvolvimento de um MVP.

3.3.3 Kanban

Kanban é uma abordagem visual para gerenciamento de fluxo de trabalho, originada no Sistema Toyota de Produção. Ele utiliza um quadro dividido em colunas, representando diferentes fases do processo, como Fazer, Fazendo e Feito, conforme ilustrado na Figura 6.

Os cartões visuais movidos entre as colunas permitem um acompanhamento claro das tarefas em andamento, facilitando a identificação de gargalos e a otimização da produtividade. Devido à sua flexibilidade, o Kanban pode ser aplicado em diferentes contextos, incluindo desenvolvimento de software, gestão de projetos e organização pessoal.

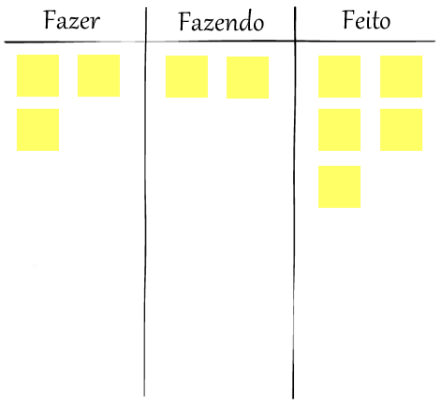


Figura 6 – Quadro Kanban simples
Fonte: (DONTUS, 2025).

3.4 Ferramentas de desenvolvimento

As ferramentas de desenvolvimento dão suporte à criação, manutenção e colaboração em projetos de software, permitindo controle de versão, acompanhamento de progresso e integração entre equipes. Plataformas como Git, GitHub e GitHub Projects são amplamente utili-

zadas por oferecerem recursos que favorecem a rastreabilidade do código e o gerenciamento visual das atividades.

3.4.1 Git e GitHub

O Git é um sistema de versionamento de código amplamente utilizado por desenvolvedores para gerenciar projetos de software de forma eficiente. Sua principal característica é o modelo distribuído, permitindo que cada colaborador tenha uma cópia completa do repositório, garantindo autonomia no desenvolvimento e maior segurança dos dados. Além disso, o Git possibilita a criação de ramificações (*branches*), permitindo que diferentes funcionalidades sejam desenvolvidas separadamente e, posteriormente, integradas ao código principal (GIT, 2025). Dentre os principais benefícios do Git, destacam-se:

- Histórico completo: Todas as alterações feitas no código são armazenadas, possibilitando que os desenvolvedores acompanhem e revertam mudanças quando necessário.
- Flexibilidade no desenvolvimento: O sistema de ramificações permite testar e desenvolver novas funcionalidades sem comprometer a estabilidade do projeto.
- Trabalho colaborativo: Com suporte a múltiplos contribuidores, permite que equipes trabalhem simultaneamente sem conflitos.

Complementando o uso do Git, o GitHub é uma plataforma online que facilita o armazenamento e a colaboração em projetos versionados. Ele fornece recursos para compartilhamento de código, revisão de alterações, rastreamento de problemas e automação de processos de desenvolvimento. Uma das ferramentas mais relevantes do GitHub é o GitHub Actions, que permite a automação de testes e implantações, otimizando o fluxo de trabalho de desenvolvimento contínuo (GITHUB, 2024).

O uso combinado do Git e do GitHub possibilita um desenvolvimento mais organizado, seguro e eficiente, sendo uma escolha essencial para equipes que buscam qualidade e controle no gerenciamento de software.

3.4.2 GitHub Projects

O GitHub Projects é uma ferramenta integrada ao GitHub, que realiza o gerenciamento de projetos de forma flexível e colaborativa. A Figura 7 apresenta o visual desta ferramenta. Basicamente, a ferramenta combina funcionalidades de quadros Kanban, listas de tarefas e automações para acompanhar o progresso do desenvolvimento de software. Ao contrário das outras ferramentas de gerenciamento, o GitHub Projects é fortemente integrado aos repositórios, o que possibilita vincular *issues*, *pull requests* e *commits* ao planejamento das tarefas e à execução do trabalho (GITHUB, 2025).

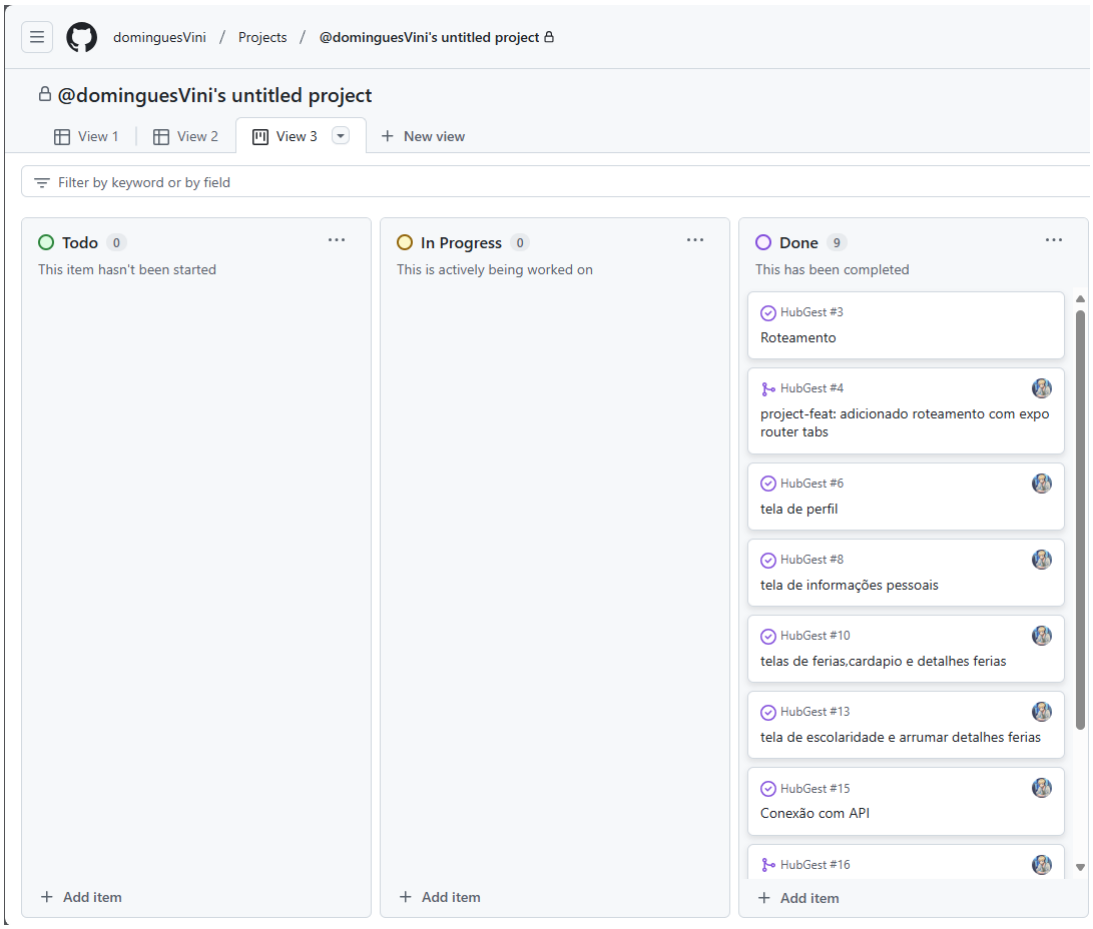


Figura 7 – GitHub Projects.
Fonte: Autoria própria.

4 MATERIAIS E MÉTODOS

4.1 Materiais

O desenvolvimento do aplicativo BairroForte utilizou as tecnologias apresentadas no Capítulo 3, aplicadas conforme as demandas de cada camada do sistema. No *frontend*, o *framework* **Flutter** foi utilizado para criação das interfaces e interação com o usuário. O **FlutterFlow** auxiliou na prototipação inicial, acelerando a construção visual das telas.

No *backend*, o **NestJS** em conjunto com o **Node.js** estruturou a lógica de negócio e a comunicação com o banco de dados **MySQL**. O ambiente foi containerizado com **Docker**, garantindo portabilidade e estabilidade durante o desenvolvimento. O aplicativo foi implantado em ambiente de nuvem utilizando a plataforma Microsoft Azure, que possibilitou o gerenciamento dos contêineres Docker e o versionamento das imagens do sistema.

Para o envio de notificações *push*(notificações enviadas diretamente ao aparelho), foi integrado o serviço **OneSignal**, responsável por encaminhar mensagens personalizadas aos usuários em tempo real. Essa integração promoveu uma comunicação direta entre o aplicativo e seus usuários, sem necessidade de configurar manualmente o FCM.

O controle de versão foi realizado com o **Git**, e o gerenciamento do código e das tarefas ocorreu por meio do **GitHub**. O **Visual Studio Code** foi utilizado como ambiente principal de desenvolvimento.

4.2 Métodos

O projeto foi desenvolvido seguindo práticas ágeis baseadas no **framework Scrum**, adaptadas à realidade de uma equipe composta por um único desenvolvedor. O professor orientador assumiu os papéis de *Product Owner* e *Scrum Master*, sendo responsável por priorizar requisitos, fornecer feedback e acompanhar o progresso das entregas. O autor deste trabalho atuou como desenvolvedor, responsável por todas as implementações.

As atividades foram organizadas em *sprints* quinzenais, cada uma com planejamento, execução e retrospectiva. No início de cada ciclo, as funcionalidades foram definidas e priorizadas; ao final, os resultados foram revisados e ajustados conforme o retorno do orientador. Essa dinâmica garantiu entregas incrementais, com evolução contínua do produto e alinhamento com os objetivos do projeto.

O processo foi monitorado através de um quadro *Kanban* no GitHub Projects, permitindo acompanhar o andamento das tarefas e identificar gargalos. Essa abordagem proporcionou um fluxo de trabalho ágil, organizado e transparente, assegurando a qualidade e consistência do desenvolvimento.

5 ANÁLISE E PROJETO DO SISTEMA

Neste capítulo, apresenta-se a maneira utilizada para desenvolver a aplicação móvel, com foco em auxiliar moradores e comerciantes de uma determinada área a monitorar a segurança local de forma colaborativa. O sistema foi concebido para permitir a troca de informações em tempo real sobre incidentes de segurança, incluindo a visualização de ocorrências, alertas de incidentes e a criação de corredores seguros com o auxílio de câmeras de vigilância. A aplicação BairroForte deve promover uma organização comunitária que aumente a sensação de segurança e ajude a inibir atividades criminosas na vizinhança.

5.1 Escopo do sistema

Para estruturar o desenvolvimento do *BairroForte*, foi adotado o método **MoSCoW** de priorização de funcionalidades, que classifica os requisitos em quatro categorias: *Must Have* (Deve Ter), *Should Have* (Deveria Ter), *Could Have* (Poderia Ter) e *Won't Have This Time* (Não Terá Desta Vez) (SEBRAE, 2024). O escopo deste projeto concentra-se na implementação de um MVP, voltado à validação inicial da proposta. Para essa versão, destinada exclusivamente à plataforma Android, foram contempladas apenas as funcionalidades *Must Have*, com o objetivo de atender às necessidades essenciais de segurança e monitoramento da comunidade. As demais funcionalidades, classificadas nas outras categorias, serão avaliadas e incorporadas em versões futuras do aplicativo, conforme a evolução do projeto e o feedback dos usuários.

A arquitetura inicial para o perfil de acesso do BairroForte foi planejada originalmente para abranger três perfis distintos: morador/comerciante, administrador e segurança privada. Contudo, em alinhamento com a estratégia de desenvolvimento de um MVP, foi tomada a decisão de focar os esforços de implementação em dois perfis centrais: o de morador/comerciante e o de administrador. A inclusão dos perfis de segurança privada e segurança pública foi estrategicamente designada para trabalhos futuros.

Nesta primeira versão, os moradores e comerciantes foram estabelecidos como os usuários-chave, sendo os principais agentes no app de segurança colaborativa. A plataforma foi construída para que eles pudessem compartilhar e receber informações sobre ocorrências, permitindo uma ação comunitária para ações de proteção em comum. O perfil de administrador, por sua vez, foi implementado com a responsabilidade de gerenciar a organização das informações e garantir a integridade dos dados na plataforma.

Os outros perfis propostos — segurança privada, para a resposta a incidentes; e segurança pública, para o monitoramento oficial — permanecem como parte da visão de longo prazo do produto. A implementação destes perfis compõe o roteiro de evolução da aplicação, a ser explorado em trabalhos futuros.

Para delimitar o escopo prático do que foi desenvolvido, a Tabela 1 apresenta as histórias do usuário classificadas como *Must Have* no método *MoSCoW*. Essas histórias constituíram o

núcleo do MVP, focando nas funcionalidades essenciais tanto para a comunidade (através do perfil de morador) quanto para a gestão da plataforma (através do perfil de administrador), de modo a garantir que a versão inicial atendesse às necessidades primordiais do projeto.

ID	Descrição (História do Usuário)
HU01	**Geolocalização para Monitoramento de Ocorrências** – Como morador ou comerciante, quero visualizar o mapa da região com marcações de incidentes, para evitar áreas de risco e planejar rotas mais seguras.
HU02	**Envio e Recebimento de Alertas em Tempo Real** – Como morador ou comerciante, quero receber notificações e enviar alertas sobre ocorrências de segurança, para contribuir com a comunidade e estar ciente de possíveis perigos.
HU03	**Comunicação Direta com Órgãos de Segurança Pública** – Como morador ou comerciante, quero me comunicar diretamente com a polícia local ou guarda municipal em casos de emergência via meio de comunicação oficial disponibilizado pelas autoridades.
HU04	**Registro e Compartilhamento de Localização de Câmeras de Segurança** – Como morador ou comerciante, quero cadastrar e compartilhar a localização de câmeras de segurança residenciais ou comerciais na área, para criar “corredores seguros” e aumentar a vigilância na comunidade.
HU05	**Alertas para Empresas de Segurança Privada e Vigilantes de Bairro** – Como morador ou comerciante, quero que o aplicativo envie notificações para a empresa de segurança privada ou vigilante do bairro, para que estes possam monitorar e responder a incidentes em tempo real.
HU06	**Canal para tirar dúvidas e reportar falhas** – Como morador ou comerciante, quero um canal para tirar dúvidas relacionadas às funcionalidades e poder reportar possíveis falhas encontradas no aplicativo.
HU07	**Recebimento de Alertas Prioritários** – Como segurança privada, quero receber notificações de incidentes em minha área de patrulha, para responder rapidamente a ameaças e proteger os moradores.
HU08	**Configuração e Monitoramento de API Segura** – Como administrador, quero implementar e monitorar uma API segura para gerenciar a comunicação de dados, protegendo a privacidade e a integridade das informações dos usuários.
HU09	**Gerenciamento de Notificações Personalizadas** – Como administrador, quero configurar as notificações do sistema de forma personalizada, para garantir que os usuários recebam informações relevantes.

Tabela 1 – Histórias de usuário

Durante o desenvolvimento do projeto, foram removidas algumas histórias do usuário. Estas estão apresentadas na Tabela 2. Essas histórias inicialmente planejadas, foram estrategicamente descontinuadas após uma análise crítica que as identificou como não essenciais para

o Produto MVP. Por outro lado, foram adicionadas novas histórias conforme apresentado na Tabela 3.

ID	Descrição (História do Usuário)
HU05	**Alertas para Empresas de Segurança Privada e Vigilantes de Bairro** – Como morador ou comerciante, quero que o aplicativo envie notificações para a empresa de segurança privada ou vigilante do bairro, para que estes possam monitorar e responder a incidentes em tempo real.
HU07	**Recebimento de Alertas Prioritários** – Como segurança privada, quero receber notificações de incidentes em minha área de patrulha, para responder rapidamente a ameaças e proteger os moradores.

Tabela 2 – Histórias removidas

ID	Descrição (História do Usuário)
HU10	**Entrar em grupo do meu bairro para receber alertas** – Eu, como morador ou comerciante, quero entrar em um grupo do bairro para compartilhar e receber notificações do meu bairro de interesse.
HU11	**Gerenciamento de informações pessoais** – Eu, como morador ou comerciante, quero gerenciar minhas informações de cadastro para garantir consistência nos dados.
HU12	**Mapa de Calor de Criminalidade** – Como usuário, quero visualizar um mapa de calor com base em dados de ocorrências passadas, para identificar áreas com maior incidência de crimes e tomar precauções ao transitar por essas regiões.
HU13	**Cadastro e Autenticação** – Como um novo usuário, quero me cadastrar e criar uma conta segura com login e senha, para proteger minhas informações pessoais e ter acesso a funcionalidades personalizadas.
HU14	**Preferências e privacidade** – Eu, como como morador ou comerciante, quero personalizar minha privacidade e preferências, para receber alertas do meu interesse e ter a privacidade ao meu controle.

Tabela 3 – Histórias adicionadas

5.2 Prototipação das telas

A prototipação das telas do BairroForte foi um passo fundamental para definir a experiência do usuário e o fluxo de interação da aplicação. As telas a seguir representam as interfaces centrais projetadas para o monitoramento colaborativo e a inserção de dados.

As Telas ilustradas na Figura 8, podem ser consideradas as principais telas do aplicativo. Nelas, o usuário pode visualizar um mapa detalhado da região, com indicadores que representam diferentes eventos e alertas, além do cadastro de informações como câmeras e incidentes(Figura 8c):

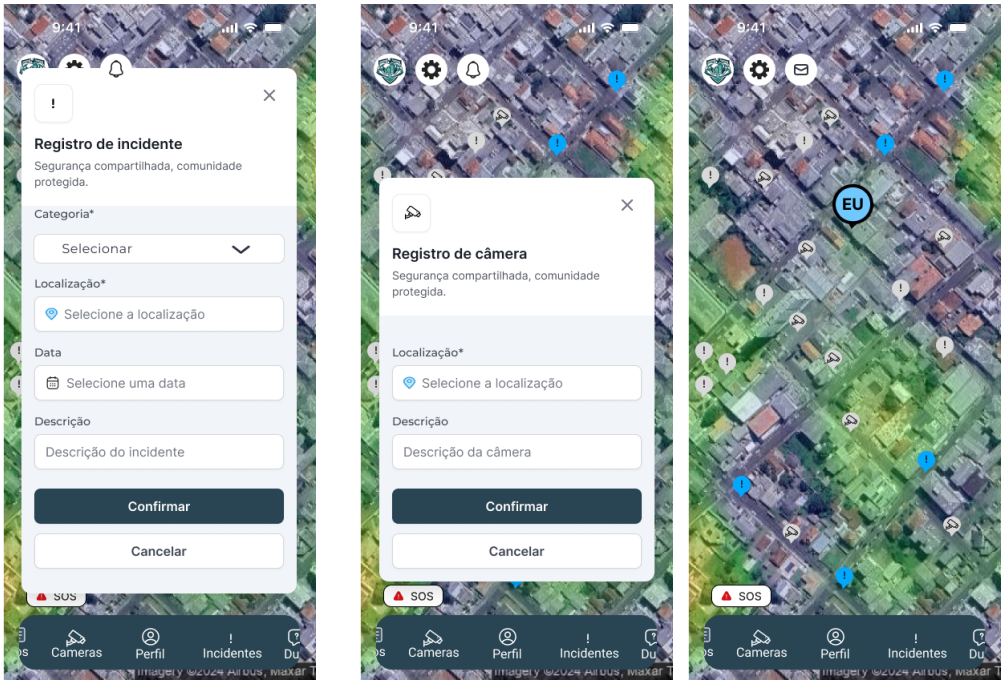
- Os ícones de exclamação indicam alertas de segurança cadastrados pelos usuários.
- Os ícones de câmera simbolizam câmeras de segurança cadastradas pelos usuários.
- A localização do usuário é destacada com um marcador.
- Há botões para acessar câmeras de vigilância, incidentes registrados, perfil do usuário e o botão de *Save Our Souls* (Código Universal de Socorro) (S.O.S.) para acesso ao número da polícia local, além de acesso a dúvidas.

Para registrar incidentes, foi prototipada a tela apresentada na Figura 8a, que é essencial para o envio e recebimento de alertas em tempo real. Esta tela possui:

- Um campo para o usuário inserir a categoria, que pode ser Roubo, Furto, Vandalismo ou Outro.
- Um campo para selecionar a localização do incidente.
- Uma opção para selecionar a data da ocorrência.
- E a opção para o usuário colocar alguma descrição.

Para o registro de câmeras, foi prototipada a tela ilustrada na Figura 8b. Essa tela é fundamental para o acompanhamento de áreas com maior cobertura de vigilância. Seus elementos incluem:

- Um campo para compartilhar a localização da câmera.
- E a opção de adicionar alguma descrição.

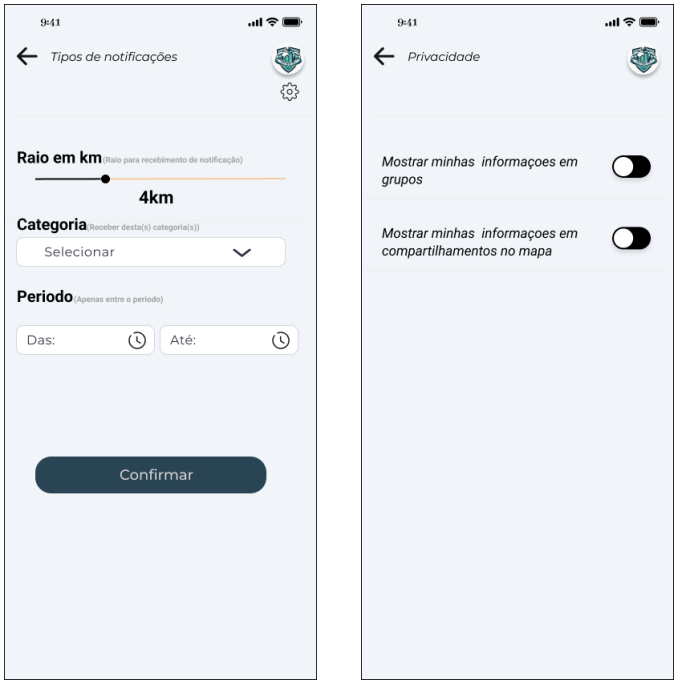


(a) Tela de registro de incidentes **(b) Tela de registro de câmeras** **(c) Tela do mapa interativo**

Figura 8 – Telas iniciais

Além das telas principais de interação, foram criadas interfaces de configuração para complementar a experiência do usuário e fornecer funcionalidades adicionais que aprimoram a usabilidade do BairroForte. Destacam-se:

- **Tela de Preferências** (Figura 9a): Permite que o usuário configure suas preferências de alertas, ajustando categorias de incidentes, raio de cobertura e somente de grupos a qual faz parte.
- **Configuração de Privacidade** (Figura 9b): Permite que o usuário ajuste suas preferências de privacidade de acordo com suas necessidades.

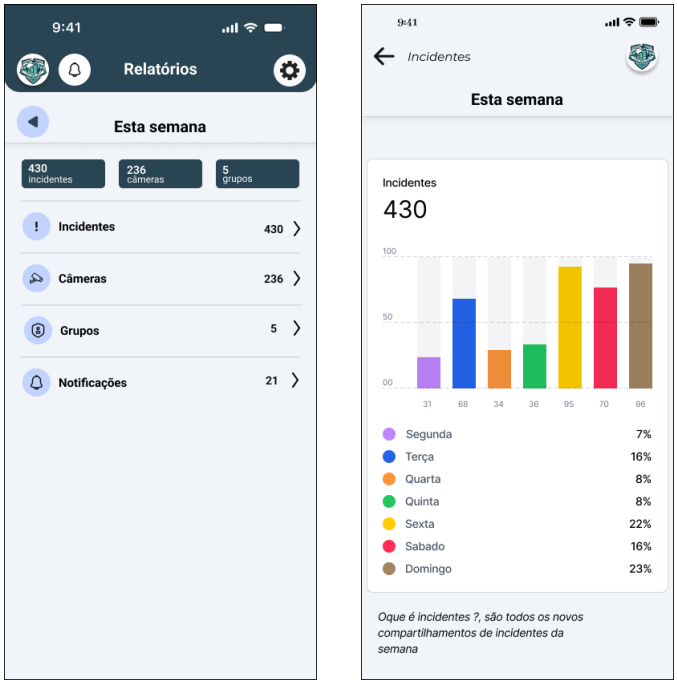


(a) Preferências **(b) Configurações de privacidade**

Figura 9 – Telas de configurações

Em conjunto, essas telas estabelecem a base para o funcionamento da aplicação, permitindo que os usuários registrem, consultem e personalizem informações relevantes para a segurança da vizinhança de forma clara e intuitiva.

- **Tela de Estatísticas e Relatórios** (Figura 10a e 10b): Exibe dados analíticos sobre os incidentes, câmeras, grupos e notificações registradas.



(a) Opções de relatórios **(b) Relatório de incidentes**

Figura 10 – Estatísticas

5.3 Modelagem do Banco de Dados

A modelagem inicial do banco de dados para o sistema pode ser visualizada na Figura 11, e consiste nas seguintes tabelas principais:

- **users:** tabela que contém todos os usuários do sistema, com suas informações de contato, como nome e e-mail, além de configurações de autenticação, como senha e tipo de usuário (morador, segurança privada, etc.). Também armazena a localização geográfica do usuário e suas configurações de privacidade, como visibilidade em grupos e compartilhamento de informações reportadas no mapa.
- **notification_settings:** tabela que armazena as configurações de notificações de cada usuário, permitindo personalizar o raio em quilômetros para receber notificações, as categorias de alertas (roubo, furto, vandalismo, etc.), o período do dia em que notificações podem ser enviadas e se as notificações devem ser limitadas aos grupos aos quais o usuário pertence.
- **incidents:** tabela que registra todos os incidentes reportados, incluindo tipo (roubo, furto, vandalismo, etc.), descrição, localização e status (aberto, em análise ou resolvido). Cada incidente é associado ao usuário que o reportou.

- **cameras:** tabela que gerencia o cadastro de câmeras de segurança, contendo a localização, área de cobertura e informações sobre o compartilhamento (se a câmera é pública ou privada).
- **security_groups:** tabela que armazena os grupos de segurança formados por moradores e comerciantes. Cada grupo tem um nome, um criador e pode ser associado a diversos usuários.
- **user_security_groups:** tabela de conexão *N para N* entre usuários e grupos de segurança. Registra quais usuários fazem parte de quais grupos, além de armazenar a data de entrada no grupo.
- **notifications:** tabela que registra todas as notificações enviadas pelo sistema. Contém o tipo de notificação (geral, geolocalizada ou personalizada), o conteúdo da mensagem e um indicador de se a notificação é direcionada à segurança privada.
- **notification_recipients:** tabela de conexão entre notificações e usuários, garantindo que cada notificação possa ser enviada para múltiplos destinatários. Registra quais usuários receberam cada notificação e a data em que foi recebida.
- **messages:** tabela que armazena mensagens trocadas entre usuários, contendo o remetente, destinatário e o conteúdo da mensagem.
- **permissions:** tabela que gerencia as permissões de acesso no sistema. Define os tipos de permissão (leitura, escrita, gerenciamento) e os usuários associados a cada uma delas.
- **faqs:** tabela que armazena as perguntas frequentes e suas respostas. Inclui categorias (como notificações, segurança e grupos) e um contador de visualizações, permitindo identificar as perguntas mais acessadas.
- **security_assignments:** tabela que associa vigilantes ou empresas de segurança a áreas ou grupos específicos. Permite gerenciar áreas de atuação e enviar notificações baseadas na localização dos incidentes.

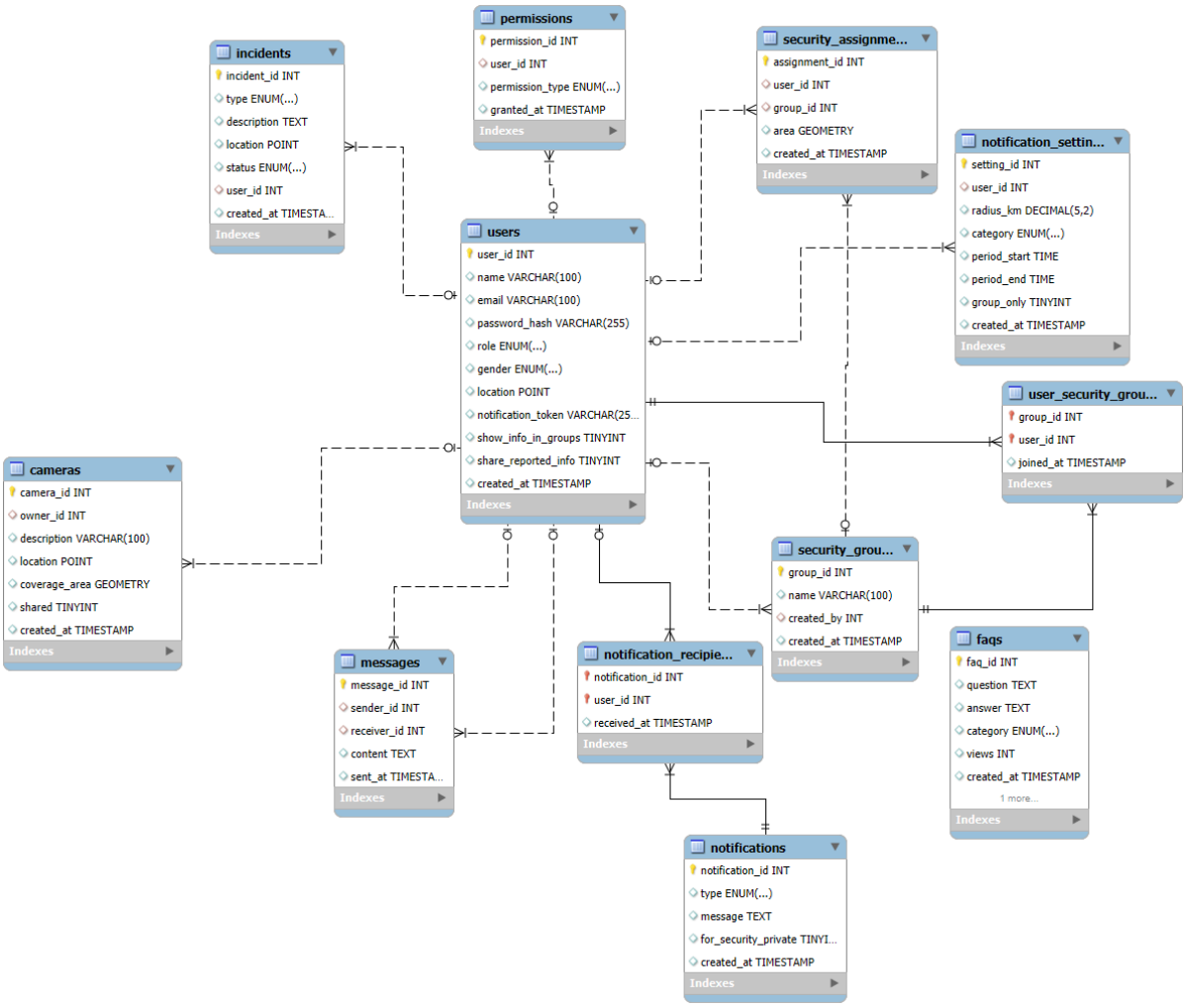


Figura 11 – Arquitetura inicial do banco de dados

Durante o desenvolvimento da aplicação, foi observado que algumas tabelas não seriam necessárias nesse primeiro momento. Por isso, o banco de dados sofreu algumas alterações, conforme apresentado na Figura 12.

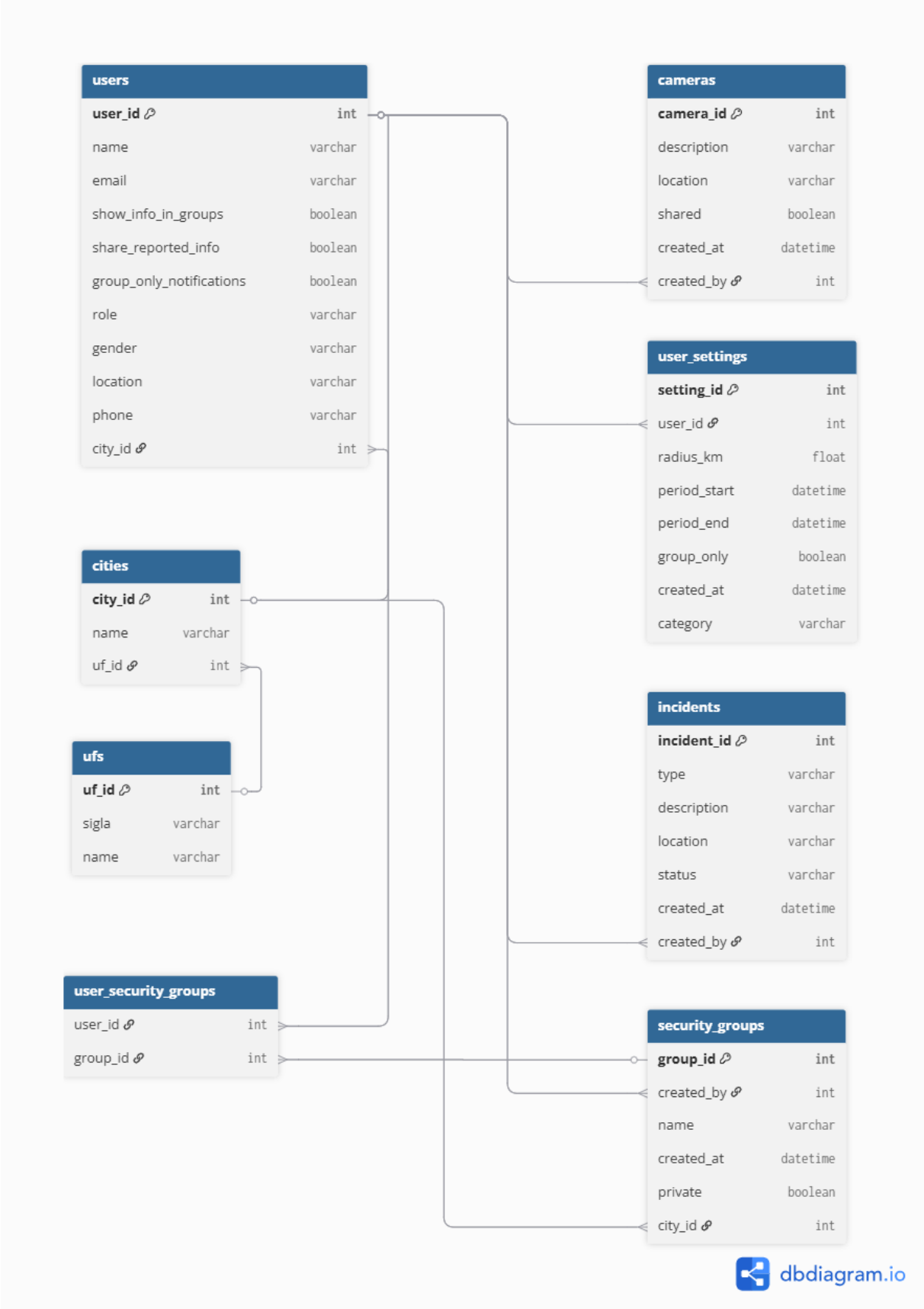


Figura 12 – Modelagem final do banco de dados

Durante o desenvolvimento, as tabelas removidas ou alteradas foram:

- **notification_settings:** Esta tabela não foi removida, mas sim renomeada para users_-settings. A decisão foi tomada porque seu escopo foi ampliado para armazenar configurações gerais do usuário, indo além de apenas notificações. O novo nome reflete de forma mais precisa a sua finalidade.

- **notifications:** A responsabilidade de registrar o histórico de notificações foi delegada ao serviço externo OneSignal. Uma vez que a plataforma OneSignal já armazena um log detalhado de todos os disparos e entregas, manter uma tabela local com a mesma informação tornou-se redundante, e sua remoção simplificou a arquitetura.

- **notification_recipients:** Pelos mesmos motivos da tabela ‘notifications‘, a gestão de quais usuários receberam cada notificação também passou a ser controlada inteiramente pela plataforma OneSignal, eliminando a necessidade desta tabela de conexão e evitando a duplicação de dados.

- **messages:** A funcionalidade de troca de mensagens diretas entre usuários foi postergada para versões futuras. Por não ser considerada um requisito essencial para o MVP, a tabela foi removida do escopo atual para permitir foco total no desenvolvimento das funcionalidades prioritárias.

- **permissions:** A remoção desta tabela ocorreu após uma reavaliação do modelo de acesso para o MVP. Foi definido que, na versão inicial, existiria apenas um perfil de usuário: o de "morador". Como todos os usuários possuem o mesmo nível de permissão, um sistema granular para gerenciar diferentes perfis de acesso tornou-se desnecessário. A reintrodução desta tabela está planejada para trabalhos futuros, quando novos perfis, como o de "segurança privada", forem implementados.

- **faqs:** Para o MVP, optou-se por uma solução mais ágil de suporte: um canal de reporte de falhas direto por e-mail. A criação de uma base de conhecimento estruturada na forma de um *Frequently Asked Questions* (Perguntas Frequentes) (FAQ) foi considerada desnecessária para a primeira versão, embora possa ser implementada no futuro, com base no feedback e nas dúvidas recorrentes dos usuários.

- **security_assignments:** Esta tabela estava diretamente ligada à HU05. Com a descontinuidade da funcionalidade de integração com segurança privada no escopo do MVP, a tabela que associava vigilantes a áreas específicas perdeu sua finalidade.

- **users:** Esta tabela foi alterada para armazenar o telefone do usuário com o campo phone e o vinculo com a cidade de cadastro no campo city_id, além da remoção do campo password e notification_token, uma vez que a autenticação e notificações ficaram designadas a outros serviços.

As alterações realizadas na estrutura do banco de dados atenderam ao propósito central do projeto, voltado ao compartilhamento comunitário de informações relacionadas à segurança. A renomeação de tabelas e a exclusão ou adiamento da criação de algumas entidades não representaram retrocessos, mas estratégias planejadas para priorizar o valor essencial do MVP.

6 DESENVOLVIMENTO

O desenvolvimento deste projeto foi realizado de maneira iterativa e incremental, baseando-se no Scrum, conforme explicado no Capítulo 4. O projeto foi organizado em sprints quinzenais, estratégia que possibilitou a entrega constante de valor e a rápida adaptação a novas necessidades, assegurando que o produto final estivesse de acordo com os objetivos estabelecidos durante todo o processo de desenvolvimento.

6.1 Fluxo de desenvolvimento

O desenvolvimento adotado neste projeto foi desenhado para assegurar controle, organização e um ciclo de entregas consistente. Inicialmente, o ambiente de desenvolvimento foi padronizado por meio do uso de Docker. Essa abordagem garantiu que a base de código se comportasse de maneira estável e previsível. O código-fonte foi versionado em um repositório privado no GitHub, assegurando a confidencialidade e o controle de acesso ao projeto.

Para a gestão das tarefas, as histórias de usuário (Tabela 1) foram cadastradas como *issues* e organizadas em um painel no GitHub Projects, seguindo o método Kanban apresentado na seção 3.3.3. Isso proporcionou uma visão clara do andamento, com colunas que representavam as etapas do fluxo de trabalho.

A estratégia de *commits* foi focada em registrar o progresso em marcos importantes. O código era enviado ao repositório após a conclusão de uma funcionalidade significativa. Ao final de cada ciclo, a versão estável do código era manualmente implantada (deploy) em um ambiente de desenvolvimento hospedado na infraestrutura em nuvem da Microsoft Azure, permitindo que as novas funcionalidades fossem validadas em um servidor análogo ao de produção.

Após a implantação neste ambiente, o processo de testes era dividido em duas frentes. A principal validação era realizada pelo *Product Owner*, que executava testes funcionais (testes de aceitação) para garantir que a funcionalidade entregue atendia aos requisitos da história de usuário. Paralelamente, para garantir a qualidade interna do código do back-end, foram implementados testes de unidade isolados no NestJS.

Esses testes unitários utilizavam o *framework* Jest para criar simulações ("*mocks*") das dependências, como os serviços ou repositórios, permitindo validar o comportamento de cada controlador de forma isolada. Por exemplo, o teste para o *GroupController* verificava se a rota *groups/available* chamava corretamente o serviço, conforme a Listagem 1.

Listagem 1 – Teste unitário do método `findAvailableGroups`

```
describe('GroupController - /groups/available', () => {
  let controller: GroupController;
  let service: GroupService;

  const mockService = { findAvailableGroups: jest.fn() } as any;

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      controllers: [GroupController],
      providers: [{ provide: GroupService, useValue: mockService }],
    }).compile();

    controller = module.get<GroupController>(GroupController);
    service = module.get<GroupService>(GroupService);
  });

  afterEach(() => jest.clearAllMocks());

  it('chama service.findAvailableGroups com email do request', async
  ↪  () => {
    mockService.findAvailableGroups.mockResolvedValueOnce([{
    ↪  group_id: 1 }]);
    const req: any = { user: { email: 'test@mail.com' } };
    const res = await controller.listAvailableGroups(req);

    ↪  expect(mockService.findAvailableGroups).toHaveBeenCalledWith('test@mai
    expect(res).toEqual([{ group_id: 1 }]);
  });
});
```

Além dos testes, um dos maiores desafios gerais no desenvolvimento do *backend* foi a curva de aprendizado conceitual do próprio NestJS. Por ser um *framework*, ele exige que o desenvolvimento siga um padrão de arquitetura estrito, baseado em Módulos e Injeção de Dependência. A maior dificuldade não foi escrever a lógica de negócio em si, mas sim aprender a pensar dentro das abstrações do *framework*. Entender como declarar corretamente as dependências, como um serviço de um módulo deveria ser exposto e consumido por outro, essa separação rigorosa de responsabilidades foi o principal desafio. Foi um exercício mais arquitetural do que lógico, focado em como estruturar o código para respeitar o design do NestJS.

6.2 Detalhamento das sprints

6.2.1 Sprint 0 - Configuração do ambiente e modelagem inicial

A Sprint 0, com duração de duas semanas, foi dedicada ao estabelecimento dos fundamentos técnicos e da infraestrutura do projeto. Embora não resulte em uma entrega de valor direto ao usuário, essa etapa é crucial para o início do desenvolvimento. O foco principal foi a configuração da infraestrutura na nuvem Microsoft Azure, onde foi criado um Grupo de Recursos e provisionado o ACR para atuar como um repositório privado para a imagem Docker da aplicação. O principal desafio técnico desta fase concentrou-se na correta configuração do ACR e na criação da imagem Docker do *backend*, garantindo um ambiente de desenvolvimento e produção padronizado e escalável. Ao final desta etapa, o *backend* estava operacional no ambiente de nuvem. Adicionalmente, a modelagem inicial do banco de dados, apresentada no capítulo 5, foi utilizada como ponto de partida para o desenvolvimento.

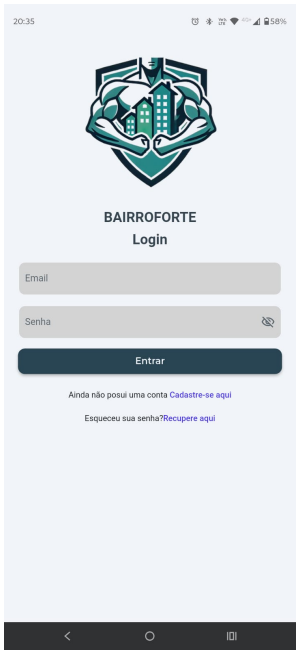
6.2.2 Sprint 1 - Cadastro e autenticação do usuário

A primeira sprint teve foco na implementação da jornada completa de cadastro e autenticação do usuário, correspondente à HU13[1]. O fluxo foi implementado como uma *timeline* de passos sequenciais, cujas telas podem ser vistas em detalhe na Figura 13.

O processo começa com a tela de login (Figura 13a) e segue para o cadastro, que se inicia com a inserção do e-mail (Figura 13b). A jornada continua com a coleta de dados pessoais como nome, cidade, data de nascimento e gênero (Figuras 13c, 13d, e 13e), a criação da senha (Figura 13f) e, por fim, a verificação do número de telefone via SMS (Figuras 13g e 13h).

O desenvolvimento foi particularmente trabalhoso devido à integração com a API externa do *Twilio* para o serviço de verificação. O principal desafio técnico em *Flutter* e *FlutterFlow* foi gerenciar o estado do formulário ao longo de múltiplas etapas, garantindo que os dados fossem mantidos de forma consistente. A autenticação, por sua vez, foi delegada ao serviço Firebase Authentication, devido a facilidade de integração com o FlutterFlow.

[1] **Cadastro e Autenticação** – Como um novo usuário, quero me cadastrar e criar uma conta segura com login e senha, para proteger minhas informações pessoais e ter acesso a funcionalidades personalizadas.



(a) Tela de login

(b) Início do cadastro

(c) Dados do usuário

(d) Data de nascimento

(e) Gênero

(f) Criação da senha

(g) Número de telefone

(h) Confirmação do código

Figura 13 – Fluxo de telas de Autenticação e Cadastro do Usuário

6.2.3 Sprint 2 - Visualização no mapa

Nesta sprint, o foco foi a implementação da funcionalidade de geolocalização, correspondente à História de Usuário HU01[2], e do botão de S.O.S., referente à HU03[3]. O desenvolvimento da tela principal do mapa, utilizando a API do Google Maps, apresentou um desafio inicial significativo em relação a plataforma FlutterFlow.

A integração da API do *Google Maps* exigiu a modificação manual de arquivos de configuração nativos do projeto para inserir a chave da API e definir as permissões de acesso à localização do dispositivo. Esse processo demonstrou a necessidade de um conhecimento mais aprofundado da estrutura de um projeto Flutter, indo além das facilidades visuais da ferramenta de desenvolvimento.

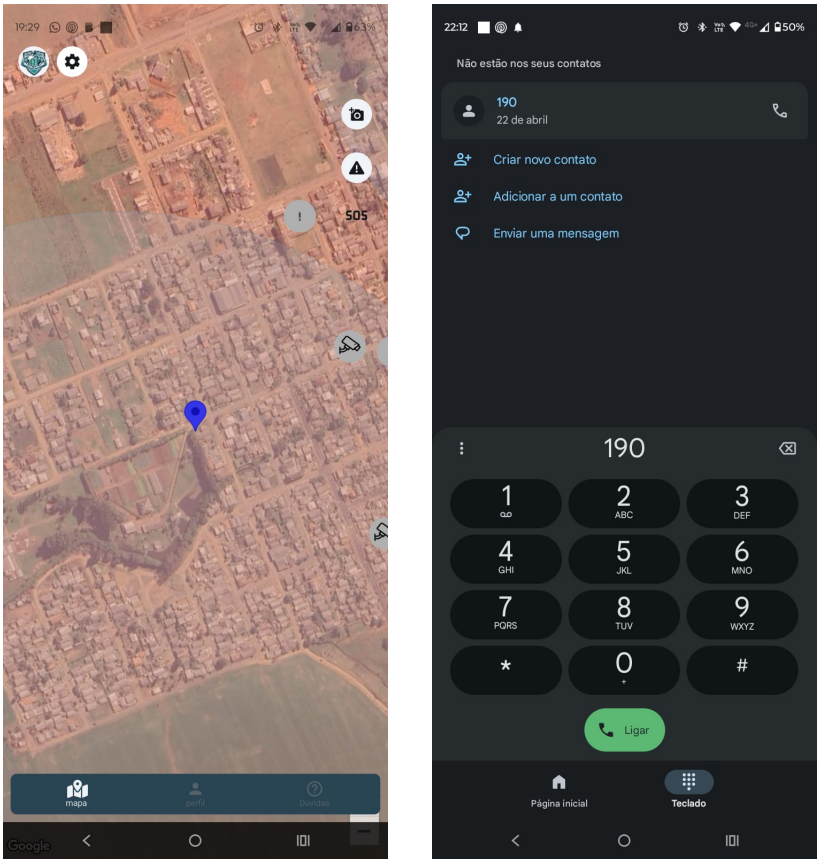
Superada essa etapa de configuração, os desafios seguintes foram a renderização de marcadores customizados para diferenciar visualmente câmeras de incidentes e a criação de um mapa de calor (*heatmap*), referente à HU12[4], para indicar a densidade de ocorrências (Figura 14a). Por fim, o botão de S.O.S. foi implementado para abrir diretamente o discador do celular com o número da polícia local (Figura 14b), finalizando as entregas de valor desta sprint.

---

[2] Geolocalização para Monitoramento de Ocorrências – Como morador ou comerciante, quero visualizar o mapa da região com marcações de incidentes, para evitar áreas de risco e planejar rotas mais seguras.

[3] Comunicação Direta com Órgãos de Segurança Pública – Como morador ou comerciante, quero me comunicar diretamente com a polícia local ou guarda municipal em casos de emergência via meio de comunicação oficial disponibilizado pelas autoridades.

[4] Mapa de Calor de Criminalidade – Como usuário, quero visualizar um mapa de calor com base em dados de ocorrências passadas, para identificar áreas com maior incidência de crimes e tomar precauções ao transitar por essas regiões.



(a) Mapa de calor e marcadores **(b) Tela do S.O.S**

Figura 14 – Telas de visualização do mapa e S.O.S

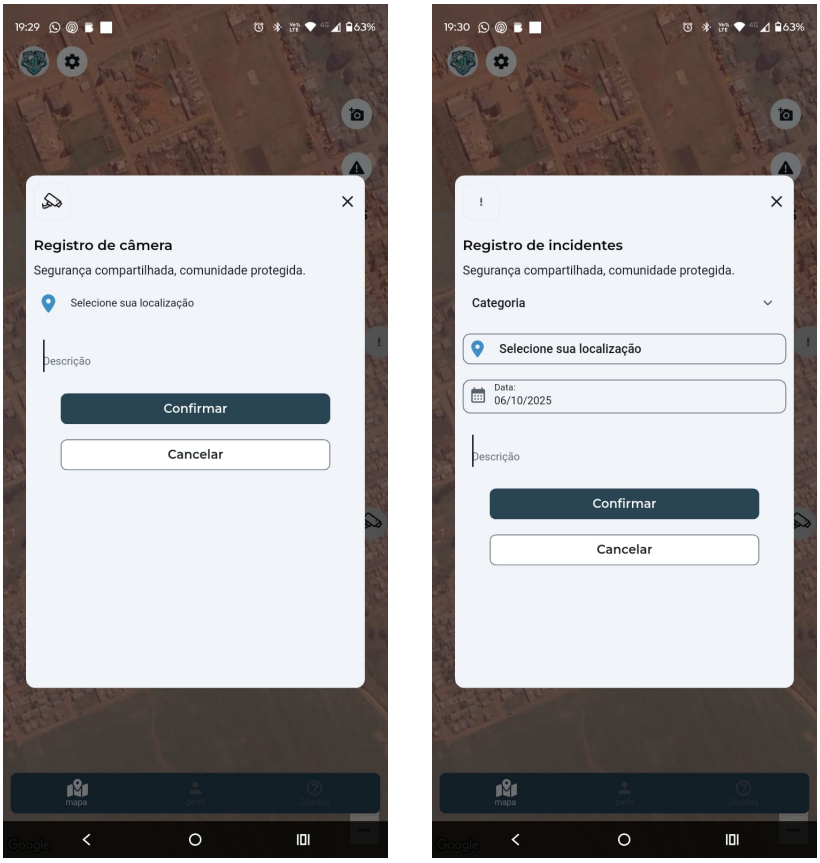
6.2.4 Sprint 3 - Cadastro de câmeras e incidentes

A terceira *sprint* deu continuidade às funcionalidades do mapa, focando na implementação do cadastro de câmeras HU04[5] e de incidentes HU02[6]. Foram criadas as telas de formulário para ambos os cadastros, como pode ser visto na Figura 15. O principal desafio técnico no *FlutterFlow* foi além da montagem visual das telas. A necessidade de capturar a localização exata de um incidente ou câmera a partir do clique no mapa precisou de personalização, pois essa interação específica não estava disponível nos componentes visuais padrão da plataforma. A solução foi implementar uma função customizada em *Dart*, aproveitando a capacidade do *FlutterFlow* de permitir a personalização do código. Este código foi desenvolvido para extrair as coordenadas geográficas do ponto selecionado e passá-las para os campos do formulário,

[5] Registro e Compartilhamento de Localização de Câmeras de Segurança – Como morador ou comerciante, quero cadastrar e compartilhar a localização de câmeras de segurança residenciais ou comerciais na área, para criar “corredores seguros” e aumentar a vigilância na comunidade.

[6] Envio e Recebimento de Alertas em Tempo Real – Como morador ou comerciante, quero receber notificações e enviar alertas sobre ocorrências de segurança, para contribuir com a comunidade e estar ciente de possíveis perigos.

demonstrando a necessidade de combinar o desenvolvimento *low-code* com programação tradicional para atender a requisitos específicos do projeto.



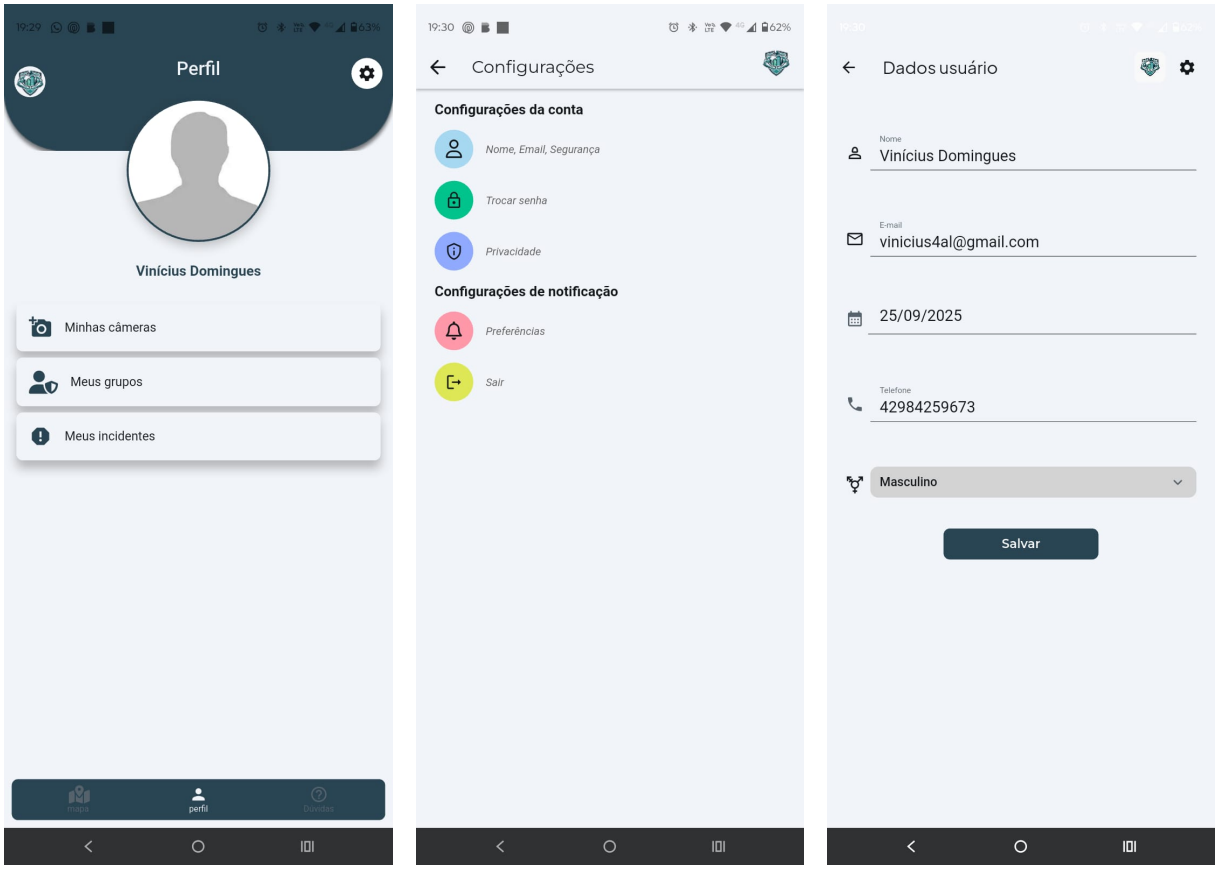
(a) Cadastro de câmeras **(b) Cadastro de incidentes**

Figura 15 – Telas de cadastro no mapa

6.2.5 Sprint 4 - Gerenciamento de conta e perfil

Esta *sprint* foi dedicada à implementação da HU11[7]. Foi desenvolvida a tela de perfil (Figura 16a), que serve como navegação para as seções do usuário. A partir dela, foi criada a tela de configurações (Figura 16b), que centraliza os acessos às diferentes áreas de personalização da conta. Também, a tela de informações pessoais(Figura 16c) foi implementada, permitindo que os usuários visualizem e editem seus dados cadastrais. No *backend*, um *endpoint* foi criado para receber e atualizar essas informações no banco de dados.

[7] Gerenciamento de informações pessoais – Eu, como morador ou comerciante, quero gerenciar minhas informações de cadastro para garantir consistência nos dados.



(a) Tela de perfil **(b) Tela de configurações** **(c) Tela de informações pessoais**

Figura 16 – Telas de navegação do perfil e gerenciamento de dados

Durante a fase de validação com o usuário, foi levantado o requisito de implementar uma funcionalidade de logout. Em resposta a esse *feedback*, foi desenvolvida a ação do botão "Sair" para encerrar a sessão do usuário via *Firebase*. Esta implementação marcou a conclusão das atividades da *sprint*.

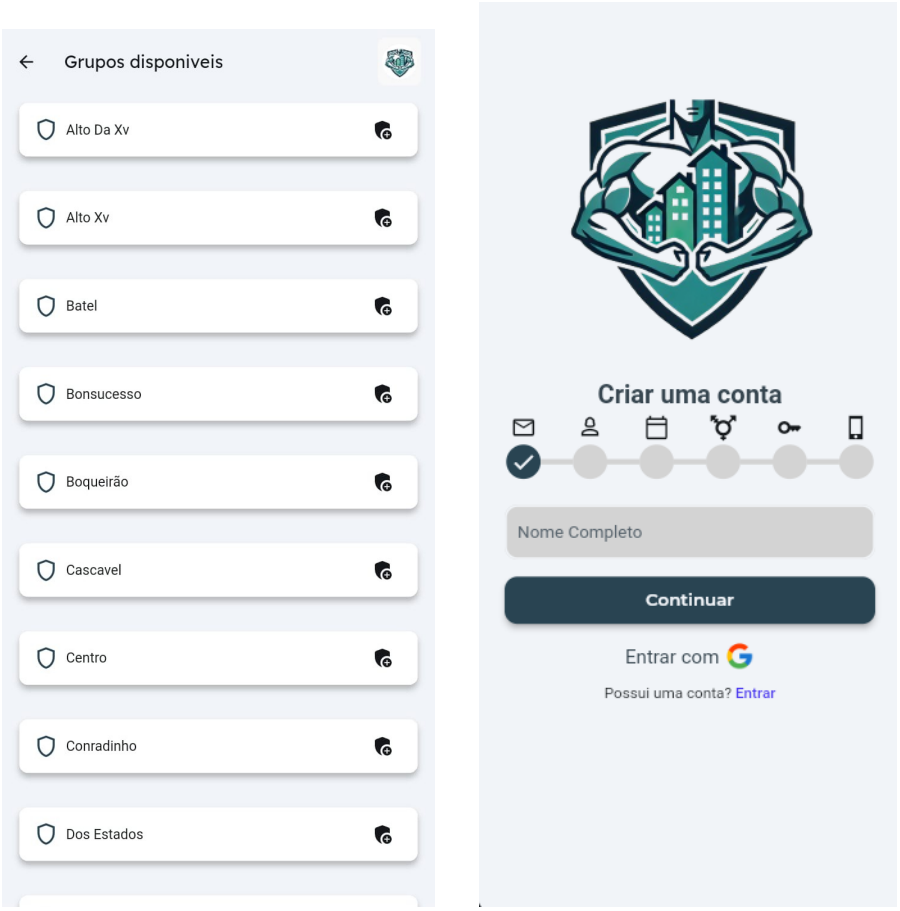
6.2.6 Sprint 5 - Ingresso em grupos de bairro

O foco desta *sprint* foi a implementação da HU10[8], que nasceu de uma reavaliação de escopo. A implementação revelou uma necessidade não prevista: para que o sistema pudesse filtrar e exibir os grupos corretos para cada pessoa (Figura 17a), era preciso saber sua cidade.

Isso levou a uma evolução na modelagem de dados, com a adição das tabelas *ufs* e *cities*, e impôs o desafio de adaptar novamente o fluxo de cadastro (Figura 17b), que já havia sido concluído. Reabrir essa funcionalidade para inserir a seleção de estado e cidade em uma *timeline* já existente exigiu um esforço considerável de refatoração do código e do gerenciamento de estado no Flutter.

[8] Entrar em grupo do meu bairro para receber alertas – Eu, como morador ou comerciante, quero entrar em um grupo do bairro para compartilhar e receber notificações do meu bairro de interesse.

Além disso, para popular os campos de seleção de localidade, surgiu o desafio de realizar a integração com a API de localidades do IBGE. Este processo envolveu o desenvolvimento de chamadas para um serviço externo, o tratamento da resposta para popular dinamicamente a lista de cidades com base no estado selecionado e a criação de uma rotina para lidar com possíveis falhas de comunicação com a API. A superação desses desafios foi crucial para viabilizar a funcionalidade de grupos, que dependia diretamente da localização do usuário.



(a) Tela de grupos de bairro **(b) Antiga tela de cadastro de dados**

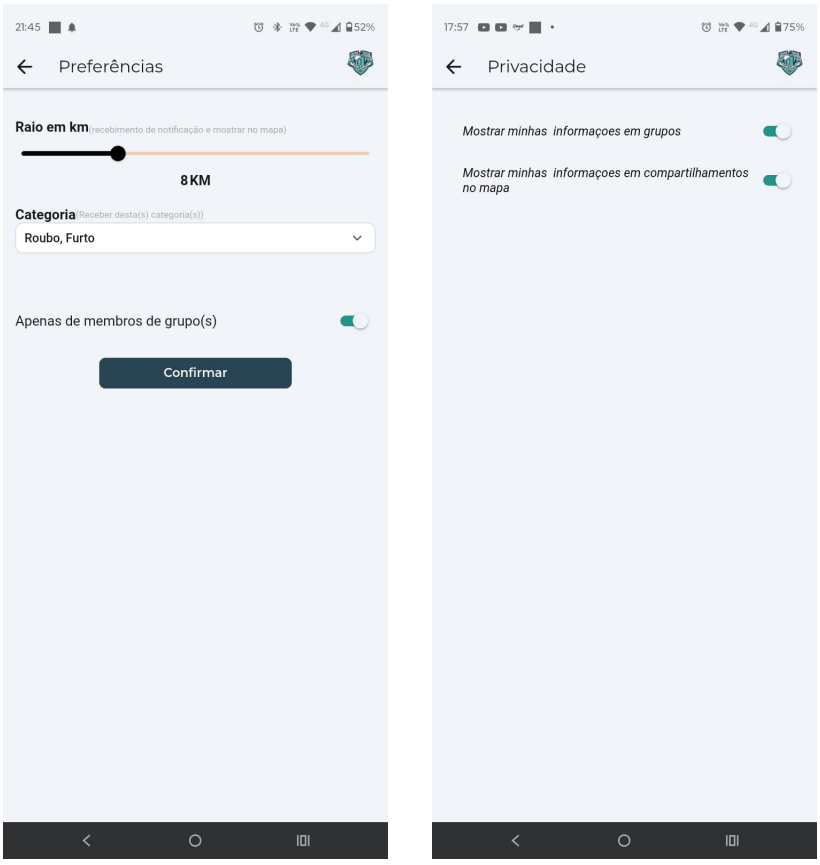
Figura 17 – Tela de grupos e a tela de cadastro que precisou ser adaptada

6.2.7 Sprint 6 - Configurações de preferências e privacidade

Nesta *sprint* o foco foi a implementação das telas de “Preferências” e “Privacidade” da HU14[9]. A Figura 18a detalha a tela onde o usuário pode ajustar o raio para recebimento de notificações e filtrar as categorias de incidentes de seu interesse. Adicionalmente, foi desenvolvida

[9] Preferências e privacidade – Eu, como como morador ou comerciante, quero personalizar minha privacidade e preferências, para receber alertas do meu interesse e ter a privacidade ao meu controle.

a tela de "Privacidade" conforme a Figura 18b, que permite ao usuário controlar a visibilidade de suas informações dentro de grupos e no mapa, garantindo maior controle sobre seus dados.



(a) Tela de preferências **(b) Antiga tela de cadastro de dados**

Figura 18 – Telas de preferências e privacidade

6.2.8 Sprint 7 - Configuração de notificações Push

O objetivo desta *sprint* foi implementar o sistema de notificações *push*, correspondente à HU09[10]. O *frontend* foi integrado ao serviço OneSignal. No *backend*, foi desenvolvida a lógica que cruza os dados de um novo incidente com as preferências definidas pelo usuário conforme implementada na *sprint* passada, garantindo que apenas alertas relevantes sejam disparados. A Figura 19 ilustra um exemplo de notificação de incidente recebida no dispositivo.

A maior dificuldade desta *sprint* foi desenvolver a consulta para cruzar os dados de um novo incidente com as preferências de usuários. A lógica precisava filtrar os usuários não

[10] Gerenciamento de Notificações Personalizadas – Como administrador, quero configurar as notificações do sistema de forma personalizada, para garantir que os usuários recebam informações relevantes.

apenas por categoria de interesse, mas principalmente pelo raio de distância geográfico que cada um configurou.

A solução adotada foi delegar todo o processamento geoespacial para o banco de dados, utilizando funções nativas do MySQL. Para isso, os pontos de latitude e longitude, do incidente foram convertidos para o tipo GEOMETRY com um sistema de referência espacial padrão (SRID 4326), utilizando a função ST_SRID. Em seguida, na consulta principal, a função ST_Distance_Sphere foi utilizada para calcular a distância esférica em metros entre o local do incidente e a localização de cada usuário. Essa função permitiu a criação de uma cláusula WHERE, que retornava apenas os usuários cujo raio de preferência era maior ou igual à distância calculada. A Figura 19 ilustra um exemplo de notificação recebida no dispositivo após essa filtragem.

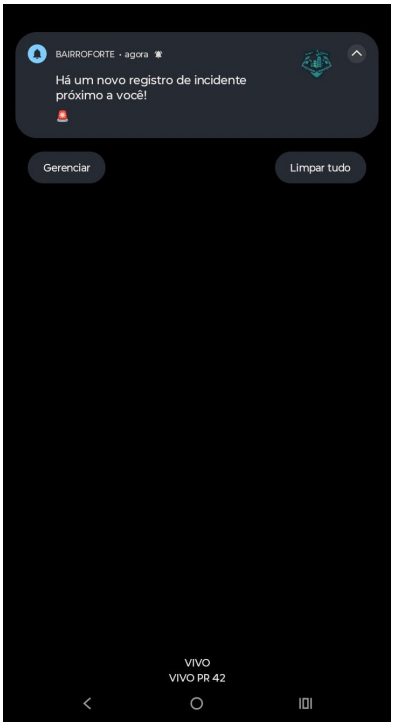


Figura 19 – Exemplo de notificação de incidente recebida

6.2.9 Sprint 8 - Geolocalização em segundo plano

Após a *sprint* passada foi percebido que a localização do usuário não era atualizada. O aplicativo dependia do ciclo de vida padrão do Flutter, e o sistema operacional interrompia a execução em segundo plano para economizar bateria. Esta sprint focou em um desafio para viabilizar os alertas geolocalizados. Foi criado um serviço em segundo plano (*background job*) no aplicativo que envia a localização do usuário ao servidor a cada cinco minutos.

A implementação de tarefas em segundo plano em dispositivos móveis é desafiadora, pois os sistemas operacionais (Android e iOS) impõem restrições rigorosas para economizar bateria. O desenvolvimento exigiu o uso de um *plugin* em Flutter para agendar e executar uma

tarefa a cada cinco minutos. O principal desafio foi configurar este serviço garantindo o envio das coordenadas geográficas para o *backend*.

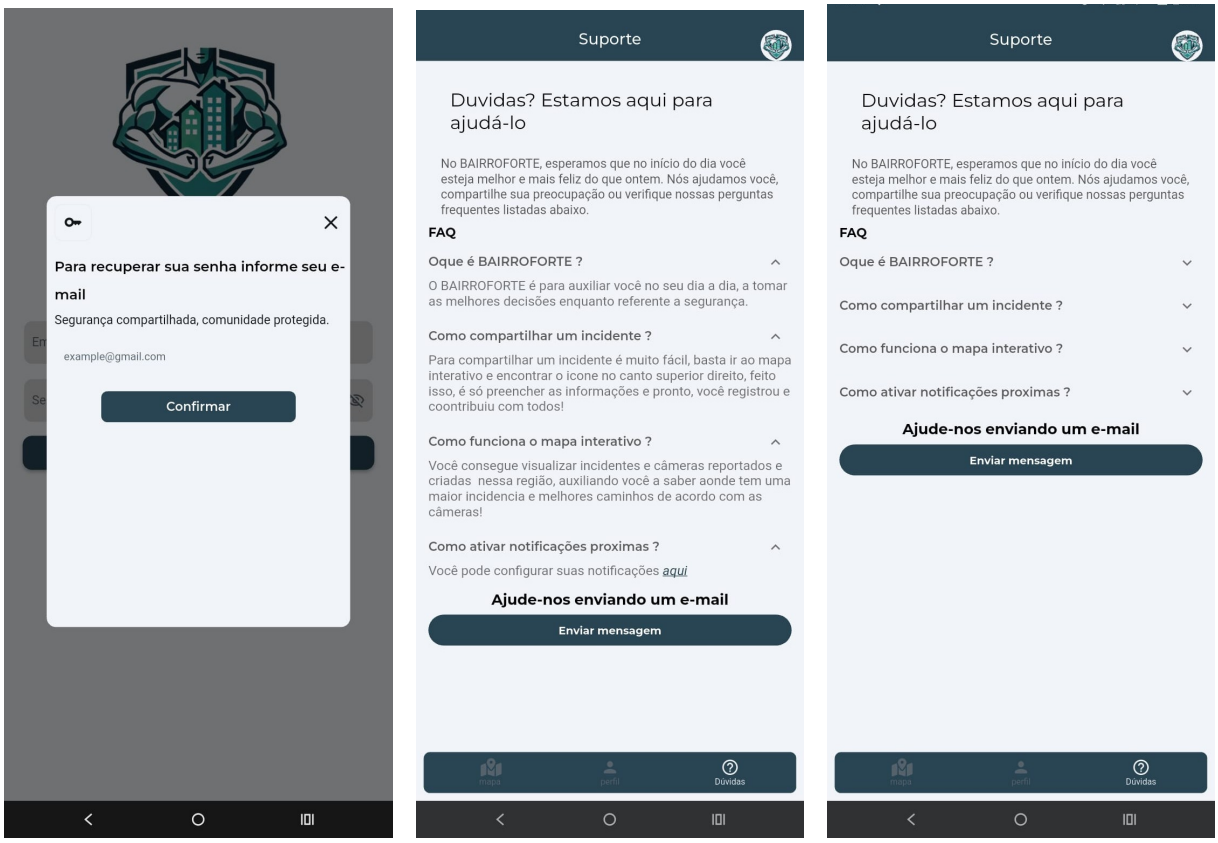
Com a conclusão desta *sprint*, a aplicação passou a ter o que que faltava para o sistema de alertas: a capacidade de receber atualizações constantes da localização do usuário. Isso tornou o envio de notificações geolocalizadas verdadeiramente eficaz, permitindo que o sistema alertasse os usuários sobre incidentes em sua proximidade independentemente de estarem com o aplicativo aberto ou não.

6.2.10 Sprint 9 - Suporte e recuperação de conta

A última *sprint* do ciclo foi dedicada a funcionalidades de suporte, referentes à HU06[11]. Foi implementado o fluxo de recuperação de conta via “Esqueceu sua senha” conforme a Figura 20, utilizando os serviços do Firebase. Também foi criada uma seção de Ajuda com um campo para reporte de falhas.

Durante a revisão da sprint, foi identificado, a partir do *feedback* do *Product Owner*, que os textos do FAQ (ilustrado na Figura 20b) não deveriam permanecer pré-expandidos. A correção foi aplicada, resultando em uma interface mais limpa e organizada, conforme ilustrado na Figura 20c.

[11] Canal para tirar dúvidas e reportar falhas – Como morador ou comerciante, quero um canal para tirar dúvidas relacionadas às funcionalidades e poder reportar possíveis falhas encontradas no aplicativo.



(a) Recuperar senha **(b) Tela de suporte (antes)** **(c) Tela de suporte (depois)**

Figura 20 – Telas de recuperação de senha e suporte (FAQ)

6.3 Considerações sobre o desenvolvimento

O ciclo de desenvolvimento do projeto foi realizado de forma iterativa através de *sprints*, resultando na implementação de 12 histórias de usuário que formam a base do aplicativo. O processo se mostrou bastante adaptativo, com o escopo e a arquitetura evoluindo conforme o entendimento das necessidades do produto se tornava mais claro.

Um bom exemplo dessa flexibilidade foi a gestão do escopo. A HU05[12] foi adiada por não ser considerada essencial para o MVP. Em seu lugar, foi priorizada a funcionalidade de ingresso em grupos de bairro HU10[13], que foi identificada como um recurso de maior impacto. Outro exemplo de adaptação ocorreu na Sprint 9(6.2.10), quando o *feedback* do *Product Owner* sobre a tela de FAQ levou a um ajuste imediato na interface.

Essa abordagem incremental também foi aplicada na modelagem do banco de dados, que não se manteve fixa. Com a evolução do projeto, o modelo inicial sofreu alterações im-

[12] Alertas para Empresas de Segurança Privada e Vigilantes de Bairro – Como morador ou comerciante, quero que o aplicativo envie notificações para a empresa de segurança privada ou vigilante do bairro, para que estes possam monitorar e responder a incidentes em tempo real

[13] Entrar em grupo do meu bairro para receber alertas – Eu, como morador ou comerciante, quero entrar em um grupo do bairro para compartilhar e receber notificações do meu bairro de interesse

portantes. Por exemplo, 6 tabelas do plano original foram removidas, pois suas funções foram delegadas a outros serviços ou adiadas. Em compensação, 2 novas tabelas *ufs* e *cities* foram adicionadas para suportar a funcionalidade de grupos. Tabelas existentes, como a users, também foram modificadas com novos campos, incluindo phone e city_id para salvar o número do usuário e a cidade de relação.

A abordagem para o desenvolvimento da API correspondente à HU08[14] também foi incremental. Em vez de construir todo o *backend* de uma só vez, a API foi desenvolvida em partes. A cada nova funcionalidade implementada no aplicativo, o trecho correspondente da API era criado e protegido em paralelo. Esse método garantiu que, ao final do projeto, a API estivesse completa.

O projeto também apresentou desafios técnicos que exigiram soluções específicas, como a utilização de consultas nativas do SQL para o cálculo de raio, o gerenciamento de estado no fluxo de cadastro em etapas no Flutter com *FlutterFlow*, e a criação de um serviço em segundo plano para o envio da localização do usuário.

Portanto, o desenvolvimento foi um processo dinâmico de construção e aprendizado. As adaptações no escopo e na arquitetura foram otimizações para focar no que traria mais valor ao usuário final. O resultado é um produto com os seus objetivos principais de segurança colaborativa.

[14] Configuração e Monitoramento de API Segura – Como administrador, quero implementar e monitorar uma API segura para gerenciar a comunicação de dados, protegendo a privacidade e a integridade das informações dos usuários.

7 CONCLUSÃO

O presente trabalho teve como motivação a sensação de insegurança em centros urbanos e a necessidade de ferramentas que capacitassem as comunidades a agir de forma colaborativa. A proposta central foi desenvolver o aplicativo BairroForte, uma plataforma de segurança comunitária que permitisse a troca de informações em tempo real entre moradores e comerciantes, visando a prevenção de crimes e o fortalecimento da segurança local.

Ao final do ciclo de desenvolvimento, pode-se afirmar que o objetivo central do projeto e proposto foi atendido. Com um MVP, que materializa a visão de uma rede de segurança colaborativa. Todos os objetivos específicos foram atingidos com sucesso:

- Foi desenvolvida um app com geolocalização que permite o monitoramento de ocorrências e câmeras em um mapa interativo.
- Foram implementadas funcionalidades para o envio e recebimento de alertas em tempo real, utilizando um sistema de notificações push.
- O aplicativo permite que os usuários registrem e compartilhem a localização de suas câmeras de segurança.
- Foi implementado um sistema de notificações personalizáveis, onde o usuário tem controle sobre o raio geográfico e as categorias de incidentes dos quais deseja ser alertado.
- A configuração de informações pessoais foi desenvolvida, permitindo ao usuário gerenciar seus dados e preferências, com a autenticação delegada a serviços como o Firebase.

O percurso, detalhado em nove sprints de desenvolvimento, demonstrou o funcionamento da metodologia ágil Scrum. A abordagem iterativa e incremental não apenas guiou a construção, mas também permitiu uma valiosa capacidade de adaptação. Um exemplo claro foi a decisão estratégica de substituir uma história de usuário de menor impacto HU05[1] pela funcionalidade de ingresso em grupos de bairro HU10[2], que se mostrou mais alinhada à proposta de valor central do aplicativo. Além disso, a superação de desafios técnicos, como a integração de APIs, a implementação de código customizado no FlutterFlow, validou as escolhas tecnológicas e a capacidade de resolução de problemas do projeto.

[1] Alertas para Empresas de Segurança Privada e Vigilantes de Bairro – Como morador ou comerciante, quero que o aplicativo envie notificações para a empresa de segurança privada ou vigilante do bairro, para que estes possam monitorar e responder a incidentes em tempo real

[2] Entrar em grupo do meu bairro para receber alertas – Eu, como morador ou comerciante, quero entrar em um grupo do bairro para compartilhar e receber notificações do meu bairro de interesse

O BairroForte não é apenas um agregado de funcionalidades, mas uma solução que busca atender a uma demanda social. O resultado é um MVP que cumpre seus objetivos principais de segurança colaborativa, com potencial para transformar a forma como as comunidades interagem e se protegem. O projeto estabelece uma base para evoluções futuras, como a integração com profissionais de segurança, reforçando sua visão de longo prazo como uma ferramenta completa para a segurança comunitária.

7.1 Trabalhos Futuros

O presente trabalho foi o desenvolvimento de um MVP funcional, que implementa as funcionalidades centrais propostas para a aplicação.

A etapa imediata consiste na publicação do aplicativo nas lojas oficiais — Google Play Store (Android) e Apple App Store (iOS). Esse processo tornará a ferramenta acessível ao público, viabilizando seu uso prático pelas comunidades.

Após o lançamento, o foco de desenvolvimento será direcionado as funcionalidades descritas na Tabela 4 cuja implementação foi postergada para manter o escopo do MVP

ID	Descrição (História do Usuário)
HU13	**Canal de Comunicação Exclusivo com Moradores** – Como segurança privada, quero ter um canal exclusivo para comunicação com moradores e comerciantes, para coordenar ações preventivas de seguranças.
HU14	**Histórico de Incidentes Personalizado** – Como usuário, quero acessar um histórico de incidentes em minha área, para analisar padrões de segurança e melhorar minha rotina de prevenção.
HU15	**Integração com Redes Sociais e Plataformas de Comunicação** – Como administrador, quero permitir que alertas e dados de segurança sejam compartilhados em redes sociais, para ampliar a visibilidade e o engajamento da comunidade.

Tabela 4 – Histórias de usuário propostas para trabalhos futuros

A implementação conjunta destes recursos tem como objetivo conectar os usuários da plataforma aos profissionais de segurança. A intenção é criar um canal de comunicação que possibilite uma resposta mais ágil e coordenada aos incidentes reportados. Essa evolução contínua é fundamental para consolidar o aplicativo como uma ferramenta de segurança colaborativa completa para a comunidade.

REFERÊNCIAS

DOCKER. **Develop faster. Run anywhere.** 2025. Disponível em: https://www.docker.com/. Acesso em: 07 fev. 2025.

DONTUS. **O que é Kanban e como aplicar em sua clínica!** 2025. Disponível em: https://dontus.com.br/o-que-e-kanban-e-como-aplicar-em-sua-clinica-2/. Acesso em: 07 fev. 2025.

EDUCAçãO, G. **Metodologia Scrum: o que é, para que serve e exemplos para aplicar no seu negócio**. 2024. Disponível em: https://g4educacao.com/blog/metodologia-scrum. Acesso em: 07 fev. 2025.

FAMILY360. **FREQUENTLY ASKED QUESTIONS**. 2024. Disponível em: https://www.family360locator.com/faqs. Acesso em: 14 nov. 2024.

FINDMYKIDS. **1 Kids GPS Tracker App**. 2024. Disponível em: https://findmykids.org/pt-br/. Acesso em: 14 nov. 2024.

FLUTTER. **Build for any screen**. 2024. Disponível em: https://flutter.dev/. Acesso em: 08 nov. 2024.

FLUTTERFLOW, I. A. r. r. **Rethink how you build**. 2025. Disponível em: https://www.flutterflow.io/product. Acesso em: 06 jan. 2024.

GIT. **About - Branching and Merging**. 2025. Disponível em: https://git-scm.com/about/branching-and-merging. Acesso em: 08 nov. 2024.

GITHUB. **Documentação GitHub**. 2024. Disponível em: https://docs.github.com/pt. Acesso em: 08 nov. 2024.

GITHUB. **Planning and tracking with Projects**. 2025. Disponível em: https://docs.github.com/en/issues/planning-and-tracking-with-projects. Acesso em: 07 fev. 2025.

GOOGLE. **Faça seu app o melhor possível com o Firebase e a IA generativa**. 2025. Disponível em: https://firebase.google.com/?hl=pt-br. Acesso em: 07 fev. 2025.

GULIYEVA, V. **Docker**. 2023. Disponível em: https://medium.com/@vezife07102002/docker-ec68f72bccfb. Acesso em: 07 fev. 2025.

IBGE. **Pesquisa Nacional por Amostra de Domicílios Contínua**. 2021. Instituto Brasileiro de Geografia e Estatística. Disponível em: https://biblioteca.ibge.gov.br/visualizacao/livros/liv101984_informativo.pdf. Acesso em: 10 out. 2024.

LIFE360. **More wandering, less wondering.** 2024. LIFE360. Disponível em: https://www.life360.com/intl/location-safety/. Acesso em: 30 out. 2024.

Microsoft Corporation. **Microsoft Azure: Cloud Computing Services**. 2025. Acesso em: 5 nov. 2025. Disponível em: https://azure.microsoft.com/.

MIYAZAKI, S. Y. M. **FATORES FISCAIS E SOCIOECONÔMICOS QUE AFETAM A CRIMINALIDADE NO BRASIL**. 2024. Disponível em: https://www.scielo.br/j/cgpc/a/rhTv7DbF6qw8ndHZYYYFJ3K/. Acesso em: 12 nov. 2024.

NESTJS. **Hello, nest!** 2024. Disponível em: https://docs.nestjs.com/. Acesso em: 11 nov. 2024.

NODEJS. **Executar a JavaScript em Toda Parte**. 2024. Disponível em: https://nodejs.org/pt. Acesso em: 08 nov. 2024.

OneSignal. **OneSignal: Customer engagement platform for notifications and messaging**. 2025. Acesso em: 5 nov. 2025. Disponível em: https://onesignal.com.

ORACLE. **MySQL and HeatWave Summit**. 2025. Disponível em: https://www.mysql.com/. Acesso em: 07 fev. 2025.

ROSáRIO, P. F. e M. **Vizinhos se unem no Whatsapp para garantir segurança**. 2025. Veja. Disponível em: https://vejasp.abril.com.br/cidades/whatsapp-vizinhos-seguranca/. Acesso em: 05 dez. 2025.

SCRUMGUIDES. **What is Scrum?** 2025. Disponível em: https://scrumguides.org. Acesso em: 07 fev. 2025.

SEBRAE. **O método MoSCoW para definição de prioridades**. 2024. Disponível em: https://sebrae.com.br/Sebrae/Portal%20Sebrae/Arquivos/ebook_sebrae_metodologia_moscow.pdfl. Acesso em: 12 nov. 2024.

Twilio Inc. **Twilio: Communication APIs for SMS, Voice, Video, and Authentication**. 2025. Acesso em: 5 nov. 2025. Disponível em: https://www.twilio.com/.

UOL. **App espião ou ajuda com filhos? Como funciona app que rastreia parentes**. 2023. Universo Online. Disponível em: https://www.uol.com.br/tilt/noticias/redacao/2023/12/10/app-life-360-como-funciona-rastrear-familia.htm. Acesso em: 30 out. 2024.