Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Desenvolvimento de Aplicativo Multiplataforma para Segurança Patrimonial

Alan Queiroz Pinho

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Euler Horta Marinho

Janeiro, 2022
João Monlevade–MG

Alan Queiroz Pinho

Desenvolvimento de Aplicativo Multiplataforma para Segurança Patrimonial

Orientador: Euler Horta Marinho

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Janeiro de 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

P654d Pinho, Alan Queiroz.
Desenvolvimento de aplicativo multiplataforma para segurança patrimonial. [manuscrito] / Alan Queiroz Pinho. - 2022.
63 f.: il.: color., gráf., tab..

Orientador: Prof. Me. Euler Horta Marinho.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Engenharia de software. 2. Aplicativos móveis. 3. Sistemas operacionais (Computadores). 4. Android (Programa de computador). I. Marinho, Euler Horta. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.72:004.451

Bibliotecário(a) Responsável: Sione Galvão Rodrigues - CRB6 / 2526

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Alan Queiroz Pinho

Desenvolvimento de aplicativo multiplataforma para segurança patrimonial

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 13 de janeiro de 2022

Membros da banca

Mestre - Euler Horta Marinho - Orientador (Universidade Federal de Ouro Preto)
Doutor - Diego Zuquim Guimarães Garcia - (Universidade Federal de Ouro Preto)
Mestra - Kattiana Fernandes Constatino - (Doutoranda em Ciência da Computação - Universidade Federal de Minas Gerais)

Euler Horta Marinho, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 26/01/2022

Documento assinado eletronicamente por **Euler Horta Marinho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 26/01/2022, às 09:31, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0272035** e o código CRC **A93B4BF2**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é dedicado a minha família e amigos que me apoiaram e incentivaram durante essa trajetória.

Agradecimentos

Agradeço aos meus pais Nilton e Lecy por sempre estarem presentes em minha vida, me apoiando, incentivando e torcendo pelo meu sucesso.

Aos meus amigos e familiares que sempre me apoiaram durante toda minha graduação.

Aos professores, demais funcionários da faculdade e colegas com quem convivi durante minha trajetória acadêmica, em especial as amizades que fiz, as quais levarei comigo para o resto da vida.

Ao meu orientador Euler, por estar sempre me auxiliando durante todo esse processo, mostrando-me as diversas oportunidades na busca pelo conhecimento e crescimento acadêmico e profissional.

"The cosmos is within us. We are made of star-stuff. We are a way for the universe to know itself."

— Carl Sagan (1934 – 1996),
in: Cosmos.

Resumo

Todos os anos, muitos brasileiros sofrem com o problema de invasão domiciliar. Diante desta problemática, a tecnologia surge como grande aliada para auxiliar na questão da segurança. Com o avanço da tecnologia e a possibilidade de produzir componentes eletrônicos cada vez menores e mais eficientes, foi possível a criação dos dispositivos móveis, em particular os *smartphones*. Junto a uma infinidade de capacidades oferecidas, houve um grande investimento em desenvolvimento de softwares para esses dispositivos. Grandes empresas buscam mais essa forma de atingir seu público, mostrando a relevância do desenvolvimento em aplicativos móveis. Este trabalho propõe o desenvolvimento de um aplicativo móvel multiplataforma para a gestão de segurança patrimonial utilizando as práticas da Engenharia de Software.

Palavras-chaves: dispositivos móveis. aplicações móveis. IOS. Android. Flutter.

Abstract

Every year, many Brazilians suffer with the home invasion problem. Faced with this problem, technology appears as a great ally to help with the issue of security. With the advancement of technology and the possibility of producing ever smaller and more efficient electronic components, it was possible to create mobile devices such as smartphones. Along with a multitude of capabilities offered, there was a large investment in software development for these devices. Large companies are looking more for this way to reach their audience, showing the relevance of mobile application development. This work proposes the development of a cross-platform mobile application for the management of property security using Software Engineering practices.

Key-words: mobile devices. IOS. Android. mobile applications. Flutter.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API Application Programming Interface

APK Android Application Pack

AVD Android Virtual Device

IDE Integrated Development Environment

JSON JavaScript Object Notation

PMMG Polícia Militar de Minas Gerais

PNG Portable Network Graphics

PWA Progressive Web App

RegEx Regular Expression

RVP Rede de Vizinhos Protegidos

SQL Structured Query Language

SI Sistemas de Informação

SDK Software Development Kit

SO Sistema Operacional

Sumário

1 Introdução

A tecnologia está cada vez mais presente no dia a dia das pessoas. Seja no trabalho, na escola ou em casa, conta-se com a tecnologia para a realização de tarefas ou até mesmo divertimento. Segundo Sommerville (2019), o mundo moderno não funciona sem algum tipo de software. Softwares são utilizados na infraestrutura de serviços públicos, na produção e distribuição industrial, no entretenimento (músicas, jogos, cinema e televisão) e, também por meio do telefone celular, uma vez que 75% da população mundial possui um telefone. Isso propicia a interação das pessoas em ambientes virtuais por meio de Sistemas Colaborativos onde o foco é a troca rápida de informações e os relacionamentos sociais.

Para garantir o desenvolvimento de um software confiável e de qualidade, é necessário utilizar técnicas da Engenharia de Software. Existem diversos cenários quando se trata do desenvolvimento. Cada projeto demanda um uso diferente de recursos, tendo como base a área a qual será destinado. Mesmo com essas diferenças, a Engenharia de Software possui diversas práticas que podem ser utilizadas, com base no contexto e necessidade, sendo necessário analisar quais serão as ferramentas utilizadas para cada etapa do projeto. De acordo com Sommerville (2019), empresas que não utilizam essas ferramentas e métodos costumam escrever programas de computador mais facilmente, porém o seu não uso no cotidiano irá resultar em softwares mais caros e menos confiáveis do que deveriam.

O termo software muita vezes está associado a programa para computadores. Porém, como ressaltou Sommerville (2019), quando se trata da Engenharia de Software, esse significado vai além, sendo também associado a documentação produzida no decorrer de seu desenvolvimento, bibliotecas e *websites* de apoio e também os dados de configurações associados. Assim, este trabalho consiste no desenvolvimento de um software aplicativo multiplataforma para a segurança patrimonial, seguindo conceitos da Engenharia de Software.

1.1 Problema

Todos os anos muitos brasileiros sofrem com o problema de invasão domiciliar. Segundo Fraga (2015), o Brasil está entre os países mais violentos do mundo e as maiores ocorrências são de crimes contra o patrimônio. Muito se investe na segurança das residências com instalação de aparatos para repreender e coibir tentativas de roubo e furto. Porém, esse tipo de crime não é evitado, basta que o residente esteja longe, para alguém mal intencionado se aproveitar.

A Polícia Militar possui um programa chamado Rede de Vizinhos Protegidos (RVP),

o qual conta com a solidariedade entre os moradores vizinhos para combater a criminalidade dentro de sua comunidade. De acordo com Emídio (2011), em Minas Gerais quando a Polícia Militar de Minas Gerais (PMMG) iniciou este programa, na cidade de Belo Horizonte, no bairro Caiçara, visando diminuir os crimes contra o patrimônio na região, foram obtidos resultados muito satisfatórios. Com o passar do tempo, outras regiões também foram sendo aderidas ao programa.

Diante dos relatos de sucesso do Rede de Vizinhos Protegidos, este trabalho surge da inspiração deste programa. Ao utilizar esse conceito, será desenvolvido um aplicativo utilizando tecnologias atuais, com a finalidade de produzir uma rede de vizinhos, na qual possam alertar uns aos outros sobre atividades suspeitas nos entornos de suas residências.

1.2 Objetivos

O presente trabalho consiste no desenvolvimento de um aplicativo móvel multiplataforma para a gestão de segurança patrimonial, que gerencie uma rede comunitária de vizinhos, com base em sua localização geográfica, objetivando criar de forma automatizada um grupo de proteção mútua, relatando-se atividades suspeitas nos entornos. Para além disso, visa-se criar de forma indireta, um banco de dados contendo relatos de atividades suspeitas, a fim de monitorar o grau de periculosidade de uma região, para que ações possam ser tomadas para diminuição da violência.

Durante o desenvolvimento deste trabalho, serão analisadas plataformas de dispositivos móveis, *frameworks* de desenvolvimento multiplataforma de aplicativos, além do desenvolvimento do aplicativo, assim como as ferramentas e conceitos de desenvolvimento *mobile* e Engenharia de Software.

1.2.1 Objetivo geral

Desenvolver um aplicativo móvel multiplataforma para a gestão de segurança patrimonial.

1.2.2 Objetivos específicos

- Descrever conceitos sobre dispositivos móveis e seu desenvolvimento;
- Analisar aplicativos correlatos em uso;
- Identificar os requisitos do aplicativo;
- Realizar a escolha e modelagem do banco de dados e suas ferramentas;

- Desenvolver o aplicativo móvel multiplataforma com uso de ferramentas que tornem esse processo rápido;
- Utilizar ferramenta de testes para a validação do aplicativo.

1.3 Organização do trabalho

O restante deste trabalho é organizado como se segue. O Capítulo 2 apresenta uma revisão bibliográfica do trabalho, aplicativos semelhantes e as tecnologias utilizadas em seu decorrer. O Capítulo 3, aborda o desenvolvimento do aplicativo e discute alguns resultados, apresentando os requisitos levantados, a modelagem do banco de dados, a implementação do aplicativo e validação com os testes. O Capítulo 4 são apresentadas as considerações finais e as propostas para trabalhos futuros.

2 Revisão bibliográfica

Contextualizado este trabalho, serão apresentados neste capítulo uma revisão bibliográfica dos principais conceitos relacionados e as tecnologias ligadas ao desenvolvimento de aplicativos multiplataformas, com foco em dispositivos móveis.

2.1 Sistemas de Informação

Muito anos após a revolução dos computadores e do surgimento do microprocessador, a Internet moderna ascendeu e começou a ser usada em nível global, possibilitando o desenvolvimento de novos softwares e soluções. A Internet agora está presente na vida de bilhões de pessoas, através de computadores e de dispositivos móveis, estes últimos responsáveis pela democratização e acesso à Internet de qualquer lugar que disponha de cobertura de operadoras de telefonia, conforme Cegielski e Rainer Junior (2015).

Um Sistemas de Informação (SI) coleta, processa, armazena, analisa e dissemina informações para um propósito específico (CEGIELSKI; RAINER JUNIOR, 2015). Esses sistemas são feitos por profissionais que buscam realiza-los a fim de solucionar algum tipo de necessidade de um projeto ou empresa. Essas necessidades variam bastante. Por isso, é necessário fazer uma análise para notar com calma quais são as necessidades levantadas para cada projeto. Um sistema de informação é um software produzido utilizando algumas práticas da Engenharia de Software. Maxim e Pressman (2021) definem como:

> Software de computador é o produto que profissionais de software desenvolvem e ao qual dão suporte por muitos anos. Esses artefatos incluem programas executáveis em computador de qualquer porte ou arquitetura. A Engenharia de Software abrange um processo, um conjunto de métodos (práticas) e uma série de ferramentas que possibilitam aos profissionais desenvolverem software de altíssima qualidade.

Seguindo essas práticas é possível produzir um software robusto e que atenda, em todos os aspectos, às necessidades de seus usuários e clientes, aplicando-as em todas as etapas de planejamento, construção e testes.

2.2 Engenharia de Software

Segundo Maxim e Pressman (2021), a Engenharia de Software é uma tecnologia que se fundamental em diferentes aspectos do desenvolvimento do software, tais como processos, métodos e ferramentas. Qualquer abordagem escolhida para o desenvolvimento deve ser fundamentada em um comprometimento organizacional com o foco na qualidade,

sendo sua pedra fundamental. Seus processos são a base e auxiliam no desenvolvimento do software de forma racional, levando em conta seus prazos estabelecidos, onde nesse processo, são produzidos diversos produtos (documentos, modelos, relatórios, etc), que servem para garantir a qualidade e o controle de todas as etapas do desenvolvimento. Os métodos têm como função o fornecimento das informações técnicas que serão utilizadas no desenvolvimento do software incluindo a comunicação, o levantamento de requisitos, a modelagem, o desenvolvimento e, por fim, a validação, principalmente com os testes. As ferramentas têm como função dar suporte aos processos e aos métodos. Ferramentas integradas ao projeto fornecem diversas informações que podem ser utilizadas por outras ferramentas, dando suporte assim ao desenvolvimento do software.

Sendo assim, Maxim e Pressman (2021) também falam que a Engenharia de Software tem uma grande importância por capacitar as pessoas que a utilizam, para o desenvolvimento de sistemas complexos, respeitando os prazos e com um alto nível de qualidade, impondo disciplina em um trabalho que pode ser caótico e ajudando as pessoas a produzirem softwares adaptados à sua abordagem e atendendo suas necessidades.

2.3 Sistemas colaborativos

Os Sistemas Colaborativos são espaços onde os usuários podem interagir entre si em um ambiente digital, de forma a poderem trocar informações em suas interações. Segundo Pimentel e Fuks (2011), para se desenvolver esse tipo de sistema, é necessário estar preparado para se adaptar aos modos de produção e estilos de vida das pessoas, proporcionados pela *Internet* pois, neste ambiente, as coisas acontecem de foma mais rápida, sendo necessário estar sempre atualizado. Esta troca de informações em ambiente virtual também pode ser definido pela Computação Social.

A Computação Social trata de aplicativos em que o software atua como uma ferramenta para proporcionar os relacionamentos sociais. Para Garcia et al. (2020), podemos dividir esses tipos de aplicativos em dois grupos: os de propósito geral, que apoiam grupos em tarefas essenciais como comunicação, cooperação e coordenação de grupo, objetivando um propósito em comum (por exemplo, Whatsapp, Google Meets, Google Docs, etc) e os de propósito específico, que tem um propósito de um determinado grupo e atende às suas necessidades específicas (por exemplo, Uber, Waze, etc). Tendo em mente para qual grupo o aplicativo irá ser desenvolvido, é possível observar quais serão as ferramentas agregadas ao projeto, a fim de facilitar as interações humanas.

2.4 Desenvolvimento de aplicativos para dispositivos móveis

Com o avanço da tecnologia e a possibilidade de produzir componentes eletrônicos cada vez menores e mais eficientes, foi a vez do surgimento de dispositivos que não necessitam de estarem presos em locais físicos, são pequenos e podem ser transportados e usados com facilidade pelos mais diversos usuários. Estes dispositivos, cada vez mais assumiram o papel de um computador, devido às diversas tecnologias que foram sendo integradas nele, sendo suficientes para realizar tarefas de forma produtiva, que antes eram delegadas a computadores maiores. Outro ponto é a capacidade de se comunicar com outros dispositivos utilizando redes sem fio, abrindo espaço para seu uso nos mais diversos locais possíveis. Como foi dito por Cegielski e Rainer Junior (2015), os *smartphones* modernos oferecem várias capacidades ao usuário:

> (...) telefonia celular, Bluetooth, Wi-Fi, câmera digital para imagens e vídeo, sistema de posicionamento global (GPS), organizador, calendário, agenda, calculadora, acesso a *e-mail* e Short Message Service (SMS – enviar e receber mensagens de texto curtas, com até 160 caracteres de extensão), mensagens instantâneas, mensagens de texto, player de música MP3, player de vídeo, acesso à internet (...).

Junto a essa infinidade de capacidades oferecidas, houve um grande investimento em desenvolvimento de softwares para esses dispositivos. Vendo essa grande popularização, várias empresas grandes buscam mais essa forma de atingir seu público, mostrando a relevância do desenvolvimento em aplicativos móveis.

Para o desenvolvimento de aplicativos móveis é necessário escolher para qual Sistema Operacional (SO) o aplicativo será desenvolvido. Atualmente, os SOs mais populares são o Android e o IOS. Por isso, são esses geralmente escolhidos como foco de desenvolvimento. Para esses casos é possível escolher pelo desenvolvimento multiplataforma, utilizando de *frameworks* que facilitam na hora de obter aplicativos para as plataformas mais populares.

2.4.1 Android

O Android é um SO baseado em Linux, utilizado principalmente em dispositivos móveis. Desenvolvido pela Google, o SO é disponibilizado sob licença de código aberto, podendo ser utilizado por outras empresas que, personalizam as *features* disponibilizadas, atendendo sua necessidade ou *design* de projeto. Por apresentar o código aberto, muitas empresas optam por utilizá-lo, ao invés de produzir todo um SO novo para seus aparelhos, que custaria muito caro e ainda contaria com a aprovação do público. Os aplicativos para estes dispositivos estão disponíveis para download na Google Play Store, local de publicação de aplicativos oficiais para esse sistema, sendo eles gratuitos ou pagos. Ainda sim é possível baixar aplicativos de outras fontes, porém os mesmo não possuem garantia

de serem oficiais e podem estar infectados com vírus e *malwares* ou ainda estar prontos para roubar os dados do usuário (AVER, 2021).

O Android Software Development Kit (SDK) é um conjunto de ferramentas utilizadas para o desenvolvimento de aplicativos para a plataforma Android. A cada nova versão do SO, é lançada um novo SDK compatível para que os desenvolvedores possam desenvolver seus aplicativos com os recursos mais recentes. Para que possam ser desenvolvidos os aplicativos, é recomendado o uso de um ambiente de desenvolvimento integrado [Integrated Development Environment (IDE)], por exemplo, o Android Studio ou um editor de código, por exemplo o Visual Studio Code. Essas ferramentas apresentam uma interface gráfica que permite que o desenvolvedor possa codificar de forma mais rápida e interativa.

2.4.2 IOS

O IOS é um SO desenvolvido pela empresa Apple, para rodar em seus dispositivos móveis. É um sistema proprietário, por isso está presente apenas em produtos da empresa sendo exclusivamente usado em seus aparelhos, garantindo um melhor controle de qualidade, pois, apenas aqueles dispositivos capazes de suportar as novas atualizações, irão recebê-las. Os aplicativos para esse sistema estão disponíveis para download na App Store, sendo o local oficial de publicação dos mesmos.

Por ser um sistema projetado pensando em na proteção dos dados de seus usuários e em sua privacidade, o IOS não permite que seu usuário personalize tanto seus aparelhos, tendo poucas permissões de sistema liberadas para evitar que os usuários habilitem alguns recursos que possam diminuir sua própria proteção (PIMENTA, 2020).

2.5 Banco de dados não relacionais

Os bancos de dados não relacionais, NoSQL ou “não somente SQL”, utilizam um padrão de armazenagem de dados diferente dos vistos nos bancos de dados que usam a Structured Query Language (SQL). Como ressaltam Laudon e Laudon (2014), estes bancos de dados utilizam um modelo de gestão de dados mais flexível, projetados para grande conjuntos de dados em computação distribuída, facilitando a escalabilidade. Em geral, eles apresentam respostas rápidas para consultas simples para grandes volumes de dados estruturados ou não. Esse tipo de banco de dados tem uma ótima capacidade de responder a situações não planejadas, além de proporcionar maior flexibilidade e liberdade aos desenvolvedores, por permitir a alteração de seus esquemas e consultas, conforme são descobertos novos requisitos para os dados.

Existem vários modelos de bancos de dados NoSQL, que também possuem um alto desempenho para processar dados estruturados. Foram criados para atender algumas necessidades específicas de uso sendo os quatro tipos mais comuns:

- **Chave-valor:** este tipo realiza um o armazenamento de chaves, ligada a valores utilizando o conceito de tabela de *hash*. É recomendado quando o valor associado é desconhecido, enquanto a chave é conhecida. Muito utilizado para o armazenamento de *big data*.

- **Documento:** os bancos de dados de documentos também possuem a organização utilizando chave-valor, porém, no campo de valor, ele armazena documentos. Os documentos não precisam estar organizados de forma estruturada, por isso são muito utilizados para armazenar dados semiestruturados.

- **Coluna:** armazena os dados em famílias de colunas, cujo os dados possuem algum tipo de relação e são acessados em conjunto. Este tipo de modelo realiza inserções muito grandes de dados. Muito utilizado por soluções que precisem de muitas leituras de dados do banco.

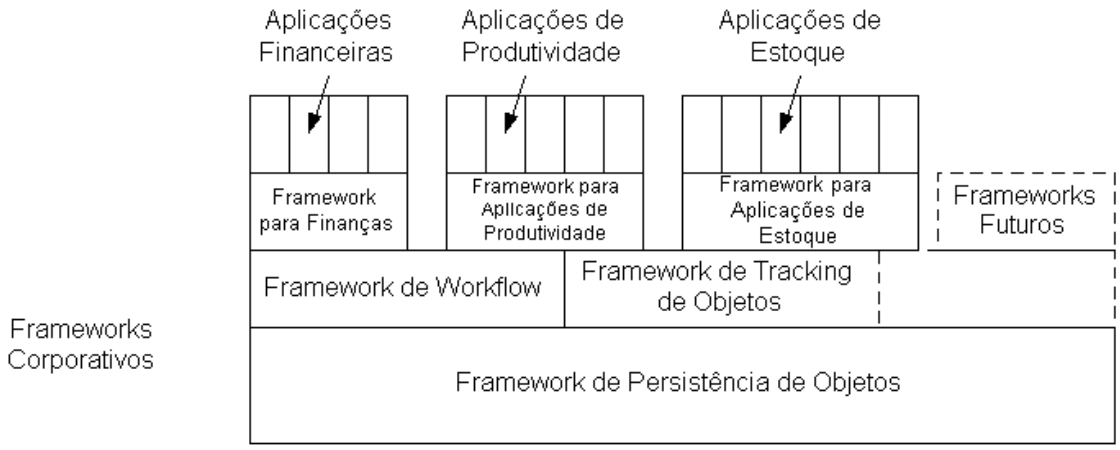
- **Grafos:** este tipo utiliza o modelo de grafos (nós, arestas e atributos) para armazenar e representar dados que possuam algum tipo de conexão. Os nós representam entidades e as arestas representam uma relação entre dois nós. Muito utilizado para analisar diferentes entidades que possuem um interconexão entre si.

Os banco de dados NoSQL apresentam características que os fazem ideais para aplicativos móveis, pelo fato de que esses tipos apresentam requisitos situacionais, forçando que o banco seja estruturalmente mais dinâmico. À medida que novos recursos vão sendo adicionados no aplicativo, o banco de dados necessita de acompanhar essas alterações e, em modelos SQL, esses tipos de alterações são mais demoradas, sendo necessárias constantes alterações em seus esquemas. Outro ponto é que com a popularização do aplicativo, a quantidade de dados armazenados vai aumentando proporcionalmente. Por isso, a fácil escalabilidade encontrada em bancos NoSQL são vantajosas para esse tipo de contexto.

2.6 *Frameworks*

Os *frameworks* são ferramentas projetadas para facilitar a resolução de problemas semelhantes na programação, apresentando soluções de problemas semelhantes, encontrados em diversos projetos. Eles apresentam ferramentas que facilitam o desenvolvimento de um sistema, e cabe aos desenvolvedores escolher qual é a melhor solução de *framework*, apresentado pelas linguagens de programação. Com essas partes já providas, o desenvolvedor deve utilizá-las e combinar com as soluções específicas do software que está desenvolvendo, adaptando seu código às necessidades do *framework*, para então conseguir concluir seu projeto, cumprindo todas as especificações.

Figura 1 – Estrutura de *frameworks*



Fonte: Sauvé (2021, p. 21)

Dentre os *frameworks*, existe uma divisão quanto a especifidade. Aqueles que resolvem problemas semelhantes de vários projetos, tendo como ponto em comum por exemplo, as camadas de persistência e de interface, são chamados de *frameworks* horizontais. Aqueles que resolvem problemas em uma classe especializada, ou seja, se especializa em uma fatia particular de problemas, por exemplo, aplicativos de controle de manufatura, financeiro, etc, são chamados de *frameworks* verticais.

2.7 Aplicativos correlatos

Foram analisados alguns aplicativos que possuem certo grau de semelhança com o aplicativo que será desenvolvido neste trabalho de conclusão de curso. Foram desenhadas as Histórias de Usuários, para podermos entender melhor o grau de semelhança, além das funcionalidades que os mesmos apresentam, a fim de validar a realização deste trabalho.

2.7.1 My Patrimonial Segurança

Trata-se de um aplicativo fornecido por uma empresa de segurança que oferece vários produtos e serviços nesta área. Desde equipamentos de segurança a vigilância orgânica. Esta empresa oferece seus serviços para pessoas comuns, empresas e condomínios. Um de seus produtos são os aplicativos para celulares Android, Apple e para *tablets*, dentre eles vale citar o My Patrimonial Segurança. As informações sobre este aplicativo são básicas, pois mesmo estando disponível em lojas de aplicativos, é necessário ter um acesso prévio aos sistemas da empresa, sendo necessário contratar seus serviços para o mesmo. A Figura 2 representa uma tela deste aplicativo encontrada pela Internet.

Figura 2 – Tela do aplicativo My Patrimonial Segurança

Fonte: https://appadvice.com/. Acesso em 28 de dez. de 2021.

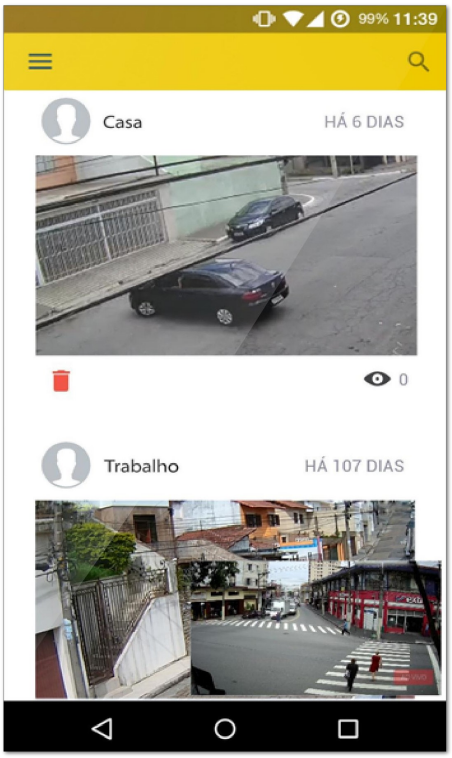
- Como cliente, devo acessar o aplicativo e escolher algum sensor do meu sistema de segurança (câmera ou alarme), para poder ter acesso às minhas câmeras de segurança instaladas.
- Como cliente, devo acessar o aplicativo e escolher ver ordens de serviço em aberto, para ver como está minha solicitação de serviço junto a empresa.
- Como cliente, devo acessar o aplicativo e escolher chamadas para contatos cadastrados no perfil, para poder ligar para algum contato de segurança para pedir ajuda.

2.7.2 Vigilância Solidária

Trata-se de um projeto da empresa Tecvoz, que atua no setor de segurança eletrônica, fornecendo produtos e serviços para diversos clientes que necessitem dos mesmos. Este projeto busca unir moradores e conscientizá-los que a solidariedade entre vizinhos é uma ferramenta muito poderosa quando tratamos de segurança e para isso apresentam a plataforma Vigilância Solidária, disponível para Android, IOS e Web. Como trata-se de um serviço pago (junto a uma franquia aplicativo + câmeras de vigilância comunitárias), não é possível ter acesso às ferramentas presentes nos aplicativos, porém no site constam

algumas informações a respeito do mesmo. A Figura 3 representa uma tela deste aplicativo encontrada pela Internet.

Figura 3 – Tela do aplicativo Vigilância Solidária



Fonte: https://play.google.com/. Acesso em 28 de dez. de 2021.

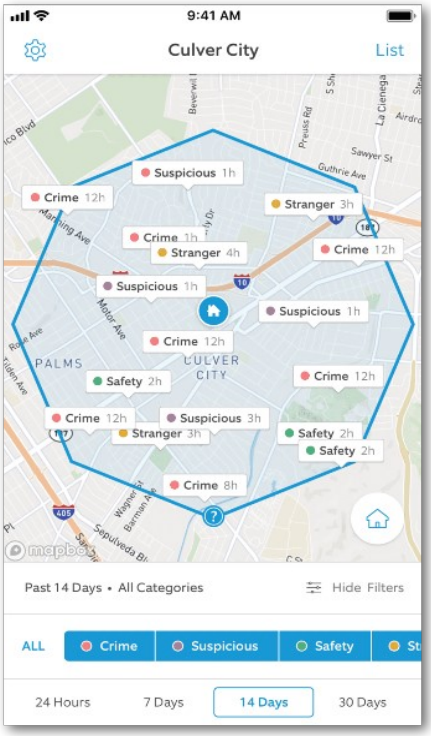
- Como cliente, devo acessar o aplicativo e escolher a funcionalidade Linha do tempo, para poder acompanhar as imagens das câmeras de segurança ao vivo ou verificar gravações.
- Como cliente, devo acessar o aplicativo e escolher a funcionalidade Mosaico de câmeras, para poder criar e acessar um mosaico das câmeras de segurança selecionadas por mim.
- Como cliente, devo acessar o aplicativo e escolher a funcionalidade Botão de pânico, para enviar um alerta vermelho para todos os vizinhos, com uma notificação para abrir uma janela com a câmera (em tempo real) de onde está acontecendo o evento notificado.

2.7.3 Neighbors by Ring

É um aplicativo da empresa Ring, uma empresa que oferece produtos e serviços de segurança patrimonial. Este aplicativo apresenta alertas de crimes em tempo real,

apresenta um *feed* contendo um quadro de informações ou *reports* sobre a vizinhança (Ex.: pessoa desaparecida, cachorro avistado, etc), sendo, em geral, uma ferramenta para unir vizinhos em prol da segurança comunitária, seja ela contra crimes ou desastres naturais. Este aplicativo é gratuito, está presente para as plataformas IOS e Android, porém, não está disponível para nosso país, tornando difícil avaliar suas *features* com precisão. A Figura 4 representa uma tela deste aplicativo encontrada pela Internet.

Figura 4 – Tela do aplicativo Neighbors by Ring



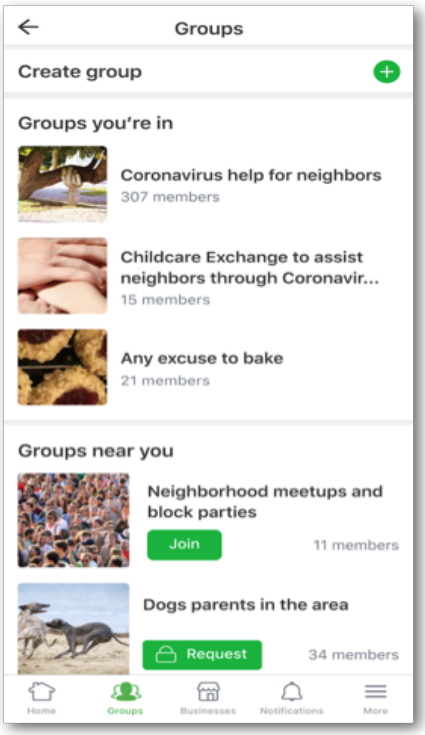
Fonte: https://www.csoonline.com/. Acesso em 28 de dez. de 2021.

- Como usuário, devo acessar o *Neighbors feed* para obter as informações sobre a vizinhança.
- Como usuário, devo acessar a opção de localização por mapa, para saber de onde vem o alerta de alguma situação ocorrida na vizinhança.
- Como usuário, devo acessar a opção *Crime Report* para ver status de crimes reportados por outros usuários.
- Como usuário devo acessar a opção *Inivite Friends* para poder convidar algum amigo a utilizar o aplicativo.

2.7.4 Nextdoor

É um aplicativo que busca reunir vizinhos em comunidades, para facilitar o compartilhamento de informações locais, anunciar algum tipo de serviço ou comércio local, emitir alertas de segurança a respeito de situações críticas sobre algum evento, entre outras *features*. Ele utiliza a geolocalização para criar e incluir os usuários automaticamente, em comunidades de vizinhos, além da possibilidade de criar grupos para organizar em algum tipo de evento. Este aplicativo está presente nas plataformas Android e IOS, porém não está disponível para nosso país, tornando difícil avaliar suas *features* com precisão. A Figura 5 representa uma tela deste aplicativo encontrada pela Internet

Figura 5 – Tela do aplicativo Nextdoor



Fonte: https://blog.nextdoor.com/. Acesso em 28 de dez. de 2021.

- Como usuário, tenho a possibilidade de escolher ver algumas das últimas atualizações das comunidades locais.
- Como usuário, posso criar grupos com vizinhos a fim de organizar algum evento dentro da comunidade.
- Como usuário, posso descobrir negócios locais, para ver avaliações e recomendações de comércios na vizinhança.

- Como usuário, posso ver anúncios e anunciar algum produto ou serviço para a vizinhança.

2.8 Tecnologias utilizadas

Nesta seção, serão apresentadas todas as tecnologias utilizadas no desenvolvimento deste trabalho.

2.8.1 Firebase

O Firebase[1] é uma plataforma digital fornecida pela Google, para facilitar o desenvolvimento de aplicativos móveis (Android e o IOS) ou Web. Conta com diversas ferramentas que integram os serviços de todos aplicativos registrados dentro da plataforma, providenciando a configuração adequada para cada contexto escolhido. O Firebase conta com um plano gratuito (Plano Spark), onde são oferecidos alguns serviços grátis e outros com algum limite baseados em sua frequência de uso ou quantidade de dados transferidos, sendo este plano utilizado neste projeto, devido às necessidades encontradas em seu decorrer.

Dentre as ferramentas apresentadas pelo Firebase, algumas foram utilizadas neste trabalho e serão detalhadas abaixo.

2.8.1.1 Firebase Authentication

Alguns aplicativos necessitam de confirmar a identidade de seus usuários para manter resguardada a privacidade dos dados, garantir o uso devido a esses usuários ou fornecer serviços personalizados para cada tipo de usuário. Esta ferramenta integra ao projeto Firebase vários métodos de autenticação dentro de seus aplicativos. As formas de autenticação externas são realizadas através da vinculação de contas, ou seja, você consegue acessar o aplicativo utilizando contas Google, Apple, Facebook, Twitter, Github, e outros e, além disso, é possível registrar usuários dentro do próprio projeto Firebase, utilizando *e-mail* e senha, ou através do número de celular.[2] Outros serviços fornecidos por essa ferramenta são:

- **Redefinição de senhas:** serviço utilizado para o usuário recuperar o acesso a sua conta no aplicativo registrado no Firebase. Esta ferramenta envia ao endereço de *e-mail* cadastrado pelo usuário, um *e-mail* com uma mensagem (personalizável pelo administrador da plataforma), contendo um *link* de recuperação de senha.

[1] Disponível em: https://firebase.google.com/
[2] Disponível em: https://firebase.google.com/docs/auth

- **Verificação de *e-mail*:** serviço utilizado para verificar se o endereço de *e-mail* que o usuário cadastrou no aplicativo, é válido, enviando para esse endereço, um *e-mail* contendo uma mensagem (personalizável pelo administrador da plataforma) e um *link* para confirmar se o *e-mail* cadastrado de fato existe.

- **Revogação de alterações de endereços de *e-mail*:** este serviço notifica o usuário quando seu endereço de *e-mail* foi alterado no aplicativo. A plataforma envia um *e-mail* (personalizável pelo administrador da plataforma) ao antigo endereço do usuário notificando-o da troca, além de mostrar o novo endereço e conter um *link* de revogação do pedido, caso o mesmo não tenha solicitado alteração de seu *e-mail*.

2.8.1.2 Firebase Realtime Database

O Realtime Database[3], é um banco de dados NoSQL hospedado na nuvem, que sincroniza os dados entre todos os usuários conectados em tempo real. Os dados são armazenados em forma de JavaScript Object Notation (JSON) e, mesmo *offline*, os eventos continuam funcionando e, quando há o restabelecimento da conexão, os dados do cliente são sincronizados com o servidor, mesclando qualquer possível conflito. Contam com regras de segurança, que definem como serão estruturados os dados e por quem poderão ser gravados e lidos, através de uma integração com o Firebase Authentication.

2.8.1.3 Firebase Cloud Storage

O Cloud Storage[4] é um serviço utilizado para armazenar arquivos em nuvem, no qual o cliente pode efetuar *downloads* e *uploads*, utilizando os aplicativos registrados pelo Firebase. Contando com integração com o Firebase Authentication, é possível definir regras de segurança para o acesso devido aos dados armazenados.

2.8.1.4 Firebase Test Lab

O Test Lab[5] é uma ferramenta que provê uma infraestrutura em nuvem para os testes dos aplicativos registrados na plataforma Firebase. Em um *data center* da Google, são executados os testes em dispositivos reais e emulados virtualmente, permitindo detectar possíveis problemas de execução em alguns tipos específicos de *Hardware*, que serão encontrados no uso real dos aplicativos. Por fim, após a conclusão dos testes é possível ter acesso aos resultados dentro da própria plataforma do Firebase.

[3] Disponível em: https://firebase.google.com/docs/database
[4] Disponível em: https://firebase.google.com/docs/storage
[5] Disponível em: https://firebase.google.com/docs/test-lab

2.8.1.5 Firebase Cloud Messaging

O Cloud Messaging[6] é uma ferramenta de envio de mensagens, para aplicativos registrados na plataforma Firebase. Com esta ferramenta, é possível notificar os aplicativos clientes sobre algum evento que precise da atenção dos mesmos. Com esta ferramenta, podemos programar os aplicativos clientes para enviar mensagens entre si, utilizando o registro de *tokens* (chave de identificação única) obtidos na inicialização do aplicativo em cada aparelho. Também, para enviar mensagens para determinados usuários e configurar o aplicativo para tomar algum tipo de ação no momento em que receber essa mensagem, podendo criar notificações que abrem telas especiais de comunicação.

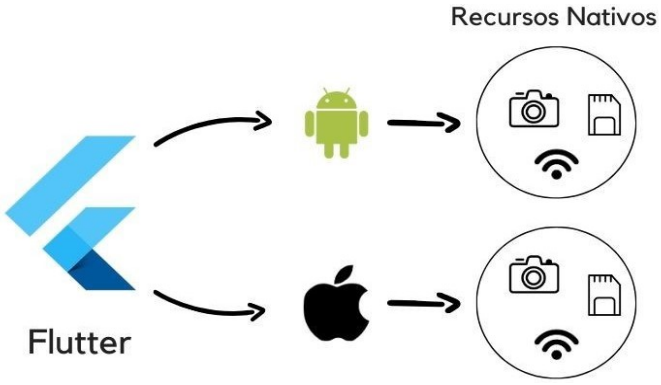
2.8.2 Flutter

O Flutter [7] é um *framework* de código aberto criado pela Google, com a finalidade de facilitar o desenvolvimento de aplicativos multiplataforma. Através de um único código base em linguagem Dart, é possível produzir para as plataformas móveis (Android e IOS), *desktop* (Windows) e Web. O Flutter conta com elementos estruturais prontos para *layout*, menus, botões, e entre outros elementos, chamados de *Widgets*, sendo possível também produzi-los do zero conforme as necessidades do usuário. Os *Widgets*, são a base da construção de aplicativos e, por serem nativos *framework*, garantem a eles uma vida útil maior, por não dependerem de intermediários para manter esses elementos em pleno funcionamento, além de garantir que não hajam muitas diferenças visuais entre os dispositivos.

Utilizando o Flutter, o código produzido em Dart é compilado para linguagem nativa da plataforma desejada, produzindo aplicativos nativos, em geral, mais rápidos e responsivos, além de ter acesso direto aos recursos presentes no dispositivo como, câmeras, *Wi-Fi*, *Bluetooth*, dentre outros.

[6] Disponível em: https://firebase.google.com/docs/cloud-messaging
[7] Disponível em: https://flutter.dev/

Figura 6 – Recursos nativos Flutter



Fonte: Andrade (2020, p. 23)

Este *framework* apresenta o *hot reload*, que permite que as alterações feitas no código sejam carregadas em tempo real no dispositivo que exibe a tela do programa. Fazendo com que os *bugs* e problemas possam ser identificados de forma mais rápida pelos desenvolvedores.

Tabela 1 – Tabela comparativo entre *frameworks*.

Framework	Ionic	Reactive Native	Flutter	Xamarin
Google Maps API	Sim	Sim	Sim	Sim
Android	Sim	Sim	Sim	Sim
IOS	Sim	Sim	Sim	Sim
Windows	Sim	Sim	Sim	Sim
Web	Sim	Não	Sim	Não
Progressive Web App (PWA)	Sim	Não	Sim	Não

Fonte: elaborada pelo autor (2021).

A Tabela 2, apresenta para quais plataformas, os *frameworks* analisados produzem aplicativos.

2.8.3 Dispositivo virtual Android

Um dispositivo virtual Android ou Android Virtual Device (AVD), é um dispositivo virtual que emula as características de um *smartphone* ou outro dispositivo que utiliza o SO Android. É possível escolher as configurações do dispositivo virtual, com base em modelos reais. Com isso, as suas configurações de hardware e limitações de computação, assim como tamanho de tela e responsividade, são usados para testar os aplicativos que irão rodar virtualmente nele. As vantagens de utilizar um AVD, é a possibilidade de testar em vários perfis de dispositivos móveis, sem necessariamente ter o aparelho fisicamente. Com esses

dispositivos, é possível acessar todas as ferramentas presentes em um *smartphone*, além de simular os diversos cenários em que o mesmo possa apresentar em seu uso cotidiano, como por exemplo:

- **Câmera:** é possível simular a câmera de um dispositivo, tirando foto de uma cena virtual que vem previamente configurada e simula um ambiente, ou carregar uma imagem para ser usada como cena, sendo possível assim adicionar códigos QR ou imagens personalizadas, para serem realizados testes de *features* que utilizem câmera.
- **Localização:** é possível escolher uma localização personalizada e o aparelho virtual irá simular estar localizado nesta localização geográfica.
- **Bateria:** é possível simular vários estados de carga de bateria, além de estado de saúde da bateria (integridade), podendo medir qual é o desempenho do aplicativo em diferentes condições.
- **Arquivos:** é possível carregar e acessar todos os arquivos gerados pelo dispositivo ou pelo aplicativo que será testado no aparelho.

Outra *feature* interessante é a possibilidade fazer capturas de tela do AVD, que poderão ser utilizadas para registrar os comportamentos apresentados pelo aplicativo testado. As capturas são salvas na área de trabalho do computador em formato de imagem Portable Network Graphics (PNG), sendo possível personalizar o local de salvamento.

2.8.4 Figma

O Figma[8] é um editor gráfico *online*, muito utilizado para desenhar interfaces gráficas e definir fluxos. Possui facilidades que permitem que várias pessoas possam trabalhar em conjunto em um mesmo projeto, permitindo que os mesmo trabalhem remotamente e de forma simultânea. Possui um plano gratuito e permite aos usuários exportarem diversos temas de graça, para acelerar a produção dos modelos. Dessa forma, é uma excelente ferramenta para produção de protótipos de software.

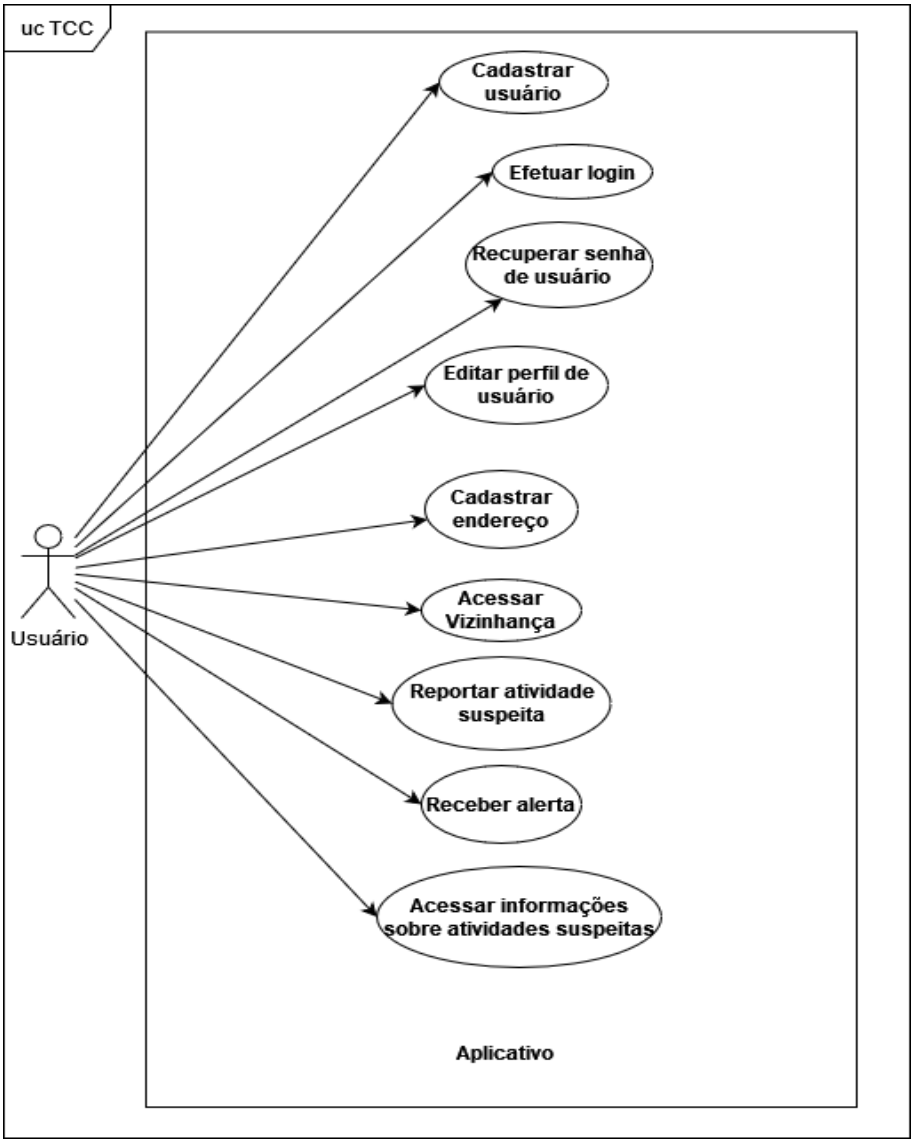
[8] Disponível em: https://www.figma.com/

3 Desenvolvimento e Resultados

3.1 Requisitos

Os requisitos são a descrição que o sistema deve fazer, seus serviços e quais serão suas restrições. Esses requisitos são feitos de acordo com as necessidades dos clientes (SOMMERVILLE, 2019).

Figura 7 – Diagrama de Casos de Uso.



Fonte: elaborada pelo autor (2022).

Foram levantados alguns requisitos para construção do aplicativo desenvolvido neste

Trabalho de Conclusão de Curso. Foram elaborados os Casos de Uso conforme mostrado na Figura 7 e, em seguida, foram detalhados em forma de Histórias de Usuários, para podermos entender melhor o funcionamento e a navegabilidade do aplicativo. As seções seguintes irão apresentar as histórias de usuário identificadas, agrupadas pelas diferentes telas do aplicativo.

3.1.1 Efetuar *login*

Trata da seção onde serão efetuadas as partes de autenticação do usuário. Conta com campos para inserção de *e-mail* e senha de acesso, botões para efetuar o *login* e cadastro e por fim um *link* para caso o usuário tenha esquecido de sua senha. Como usuário devo:

- Preencher os campos de "E-mail" e "Senha" e em seguida clicar no botão "Entrar" para Autenticar junto ao aplicativo, caso já cadastrado no aplicativo.
- Clicar no botão "Cadastrar" para ter acesso a seção Cadastrar.
- Clicar em "Esqueceu sua Senha?", para ter acesso a seção de Recuperar Senha, caso já possua conta no aplicativo e tenha esquecido da senha.

3.1.2 Cadastrar usuário

Trata da seção onde o usuário inserirá seus dados afim de criar um conta para acessar o aplicativo. Como usuário, devo:

- Preencher o campo "Nome", com o nome, de modo a registrar no aplicativo.
- Preencher o campo "Sobrenome", com o sobrenome, de modo a registrar no aplicativo.
- Preencher o campo "E-mai", com o *e-mail*, de modo a registrar no aplicativo.
- Preencher o campo "Senha" e "Confirmação de senha", com a senha desejada (mínimo oito caracteres, símbolos especiais, números, letras maiúsculas e minusculas), de modo a registrar no aplicativo.
- Clicar no botão "Já possui conta?", para cancelar o cadastro ou caso lembre que já possuo conta cadastrada, de modo a voltar para seção Autenticar.

3.1.3 Recuperar Senha de usuário

Trata da seção contendo o campo "E-mail" mais dois botões: "Enviar" e "Voltar", utilizado para que o usuário possa recuperar seus acessos ao aplicativo. Como usuário, devo:

- Preencher o campo “E-Mail” com o *e-mail* utilizado no cadastro, de modo que, possa receber as informações de recuperação de senha neste *e-mail*.
- Clicar no botão “Voltar”, para cancelar o cadastro ou caso não possua nenhum endereço de *e-mail* previamente cadastrado no aplicativo, de modo a voltar para a “Tela de Login”.

3.1.4 Editar Perfil de usuário

Nesta tela, podemos observar as informações do perfil do usuário: foto de perfil, nome, sobrenome, *e-mail*, número de celular, telefone residencial e senha. Como usuário, devo:

- Clicar no botão de máquina fotográfica na moldura circular, de modo que possa atualizar a foto de perfil.
- Clicar no botão no formato de lápis em frente ao “Nome”, de modo que possa editar o nome.
- Clicar no botão no formato de lápis em frente ao “Sobrenome”, de modo que possa editar o sobrenome.
- Clicar no botão no formato de lápis em frente ao “E-mail” cadastrado,de modo que possa editar o *e-mail*.
- Clicar no botão no formato de lápis em frente ao “Celular”, de modo que possa editar o número de celular.
- Clicar no botão no formato de lápis em frente ao “Telefone Residencial”, de modo que possa editar o telefone residencial.
- Clicar no botão no formato de lápis em frente a “Senha” cadastrado, de modo que possa editar a senha.

3.1.5 Acessar informações sobre atividades suspeitas

Trata-se de uma tela contendo todas as atividades suspeitas que foram notificadas, dentro do grupo da vizinhança do usuário.

- Como usuário, posso visualizar todos os eventos que ocorreram dentro do grupo da vizinhança, para poder saber qual a data e hora ocorreram, de modo possa ser informado como está a vizinhança.
- Como usuário, devo clicar no botão “Voltar”, para voltar para tela inicial.

3.1.6 Cadastrar endereço

Trata de uma tela contendo os campos necessários para o cadastro do endereço dentro do aplicativo. Como usuário, devo:

- Preencher os campos “Cep”, para que o aplicativo possa pesquisar em uma API, de modo que alguns dos campos seguintes possam ser preenchidos de forma automática.
- Preencher o campo “Logradouro” (caso o mesmo esteja em branco) com o nome da rua ou avenida, de modo que possa ser registrado no banco de dados.
- Selecionar no campo “Tipo de complemento”, se a residência for em uma Casa ou em um Apartamento, de modo que possa ser registrado no banco de dados.
- Preencher o campo “Complemento”, com o devido complemento de onde resido. Caso seja “Casa A”, devo apenas preencher com a letra A; caso não tenha complemento, deixar em branco; caso seja apartamento, preencher com o número do apartamento, de modo que possa ser registrado no banco de dados.
- Preencher o campo “Número” (caso o mesmo esteja em branco) com o número residencial, de modo que possa ser registrado no banco de dados.
- Preencher o campo “Bairro” (caso o mesmo esteja em branco) com o bairro, de modo que possa ser registrado no banco de dados.
- Preencher o campo “Cidade” (caso o mesmo esteja em branco) com a cidade, de modo que possa ser registrado no banco de dados.
- Preencher o campo “Estado” (caso o mesmo esteja em branco) com o estado, de modo que possa ser registrado no banco de dados.
- Clicar no botão “Cadastrar”, após preencher todas as informações pedidas, de modo que o aplicativo possa registrar no banco de dados as informações e retornar algum *feedback*.
- Clicar no botão “Voltar”, para cancelar o cadastro da residencia, de modo a voltar para a tela inicial.

3.1.7 Acessar Vizinhança

Trata-se de uma tela contendo uma visão mais detalhada do mapa do grupo da vizinhança do usuário. Ela contém o cabeçalho, com o botão de retorno e os botões contidos no rodapé. No mapa, temos dois tipos de marcadores: o com formato de casa, que é a localização do usuário e o marcador de mapa, indicando os vizinhos dentro do aplicativo. Como usuário, devo:

- Clicar no botão de seta apontada para esquerda para voltar a Tela Principal.
- Clicar no marcador do mapa, para abrir a tela de notificação de vizinho, de modo a enviar uma notificação.
- Clicar em "Alertas", para ler as informações sobre atividades suspeitas registradas anteriormente.

3.1.8 Reportar atividade suspeita

Trata-se de uma tela contendo os campos que serão preenchidos com as características do registro de atividade suspeita. Os campos presentes são: descrição curta, descrição longa e endereço do vizinho. Também contém dois botões: reportar e voltar. Como usuário, devo:

- Selecionar no campo "Descrição curta" qual o tipo pré definido de atividades a serem reportadas, para serem enviadas ao vizinho.
- Preencher o campo "Descrição longa", com detalhes da atividade a ser reportada, para ser enviada ao vizinho.
- Observar se o campo "Endereço do vizinho" está preenchida com o endereço do vizinho para o qual será reportada a atividade.
- Apertar o botão "Reportar", após ter preenchido os campos contendo as informações da atividade suspeita, para poder enviar ao vizinho.
- Apertar o botão "Voltar", para cancelar o alerta de atividade suspeita e retornar à Vizinhança.

3.1.9 Receber alerta

Trata-se de uma tela que contém todas as informações fornecidas pelo vizinho, quando o mesmo detectou e registrou alguma ação suspeita no aplicativo. Como usuário, devo:

- Clicar na notificação que será exibida em meu telefone, para abrir a tela contendo as informações da atividade suspeita relatada.
- Olhar para o campo "Registrado às", de modo que eu fique ciente da hora em que o evento foi registrado pelo vizinho.
- Olhar para o campo "Evento reportado por um vizinho", de modo que eu fique ciente da descrição detalhada do evento foi registrado pelo vizinho.

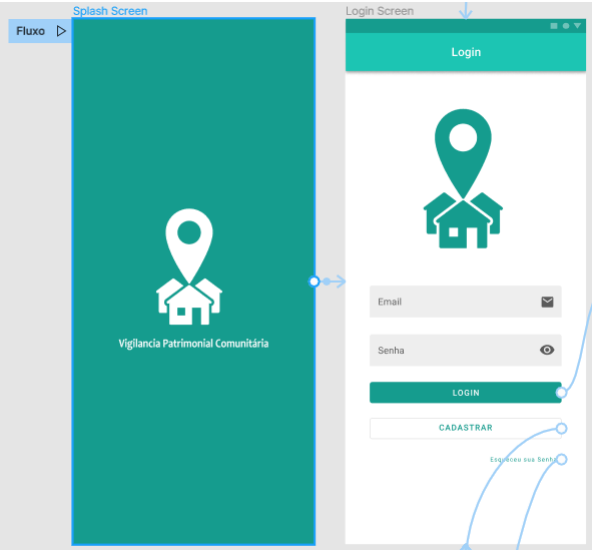
- Olhar para o campo “Seu endereço”, de modo que eu fique ciente se o endereço está correto no evento que foi registrado pelo vizinho.
- Clicar no botão “Confirmar leitura”, de modo que fique registrado no banco de dados a leitura do evento e, em seguida, a tela inicial será exibida.

3.2 Protótipos

O protótipo é uma versão experimental do software, onde é levado em conta a aplicação dos conceitos básicos levantados pelos requisitos. É botado em prática alguns conceitos e detalhamentos dos software, e podem também se abordar aspectos visuais e estruturais do software.

Com a definição das Histórias de Usuário, foi possível ter uma ideia melhor de qual o fluxo de telas e quais funcionalidades deveriam ser aplicadas no projeto. Tendo isso, foi possível elaborar o *design* das telas que seriam utilizadas no aplicativo, através da ferramenta de prototipação Figma. Devido ao uso do *Material Design*[1], pelo *framework* Flutter, como tema padrão de telas, foi utilizado o kit base de *design Material 2 Design Kit*[2], que é uma base para desenhar telas de aplicativos usando o Material Design dentro do Figma. Foram feitas alterações nas paletas de cores e foram adicionados alguns elementos para deixar o aplicativo mais personalizado.

Figura 8 – Início do fluxo Figma.

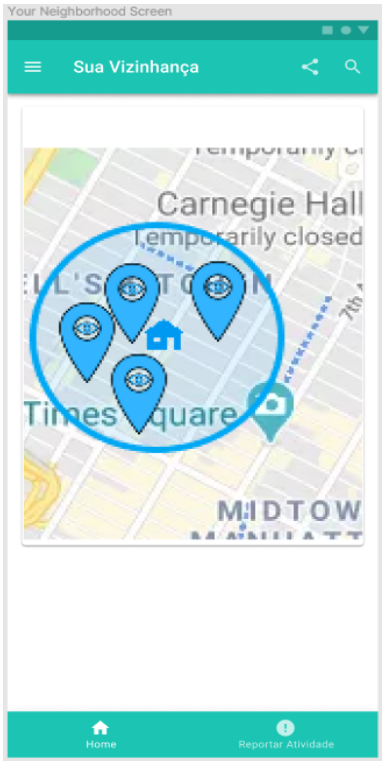


Fonte: elaborada pelo autor (2021).

[1] Disponível em: https://material.io/
[2] Disponível em: https://www.figma.com/@materialdesign

A Figura 8 mostra o início do fluxo das telas do aplicativo, as setas mostram as transições de uma tela para outra. Os elementos posicionados neste protótipo, possuem algum tipo de interação que retorna como que o aplicativo irá se comportar quando estiver pronto. A tela mais importante do aplicativo é a "Tela de Vizinhança", representada conforme a Figura 9, por um dos modelos de tela produzidos no Figma.

Figura 9 – Protótipo da Tela de Vizinhança.



Fonte: elaborada pelo autor (2021).

3.3 Modelagem do Banco de Dados

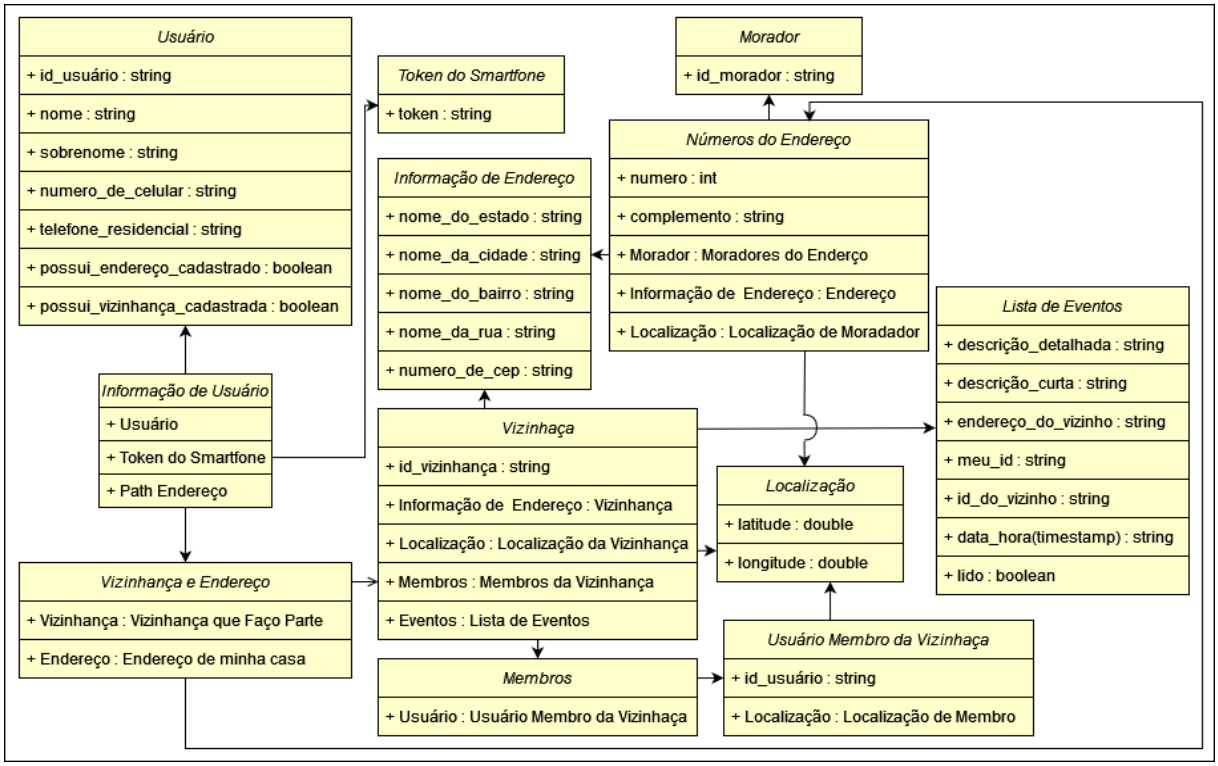
Conforme exposto na Sessão 2.5, os banco de dados NoSQL, apresentam muitas características que o fazem muito recomendado para aplicativos móveis, por essa razão, foi escolhido para o desenvolvimento do aplicativo proposto neste trabalho. Com o modelo escolhido, foi a vez da plataforma que forneceria esse serviço, sendo o Firebase Realtime Database da Google, o escolhido. Com o levantamento dos requisitos e a elaboração das Histórias de Usuário, foi possível modelar o banco de dados que será utilizado neste projeto. Para melhor visualização, foi escolhido o Diagrama de Classes, para representar a estrutura do banco de dados utilizado no aplicativo.

3.3.1 Diagrama de Classes

O Diagrama de Classes tem como função representar sistemas orientados a objeto. A classe pode ser representada como um quadro dividido em partes. Na primeira, é colocado o nome. Em sequencia, os atributos encontrados nesta classe e, por fim, em algumas ocasiões, são colocados os métodos. Quando as classes desse sistema possuem algum relacionamento, este relacionamento é representado com uma linha que ligam os quadros.

Segundo (SOMMERVILLE, 2019), os diagramas de classes são parecidos com modelos semânticos de dados e esses tipos de modelos são usados no projeto de banco de dados. Por esse motivo, este tipo de diagrama foi escolhido para representar o modelo de dados utilizado neste trabalho, conforme representado na Figura 10.

Figura 10 – Diagrama de Classes do Banco de Dados.



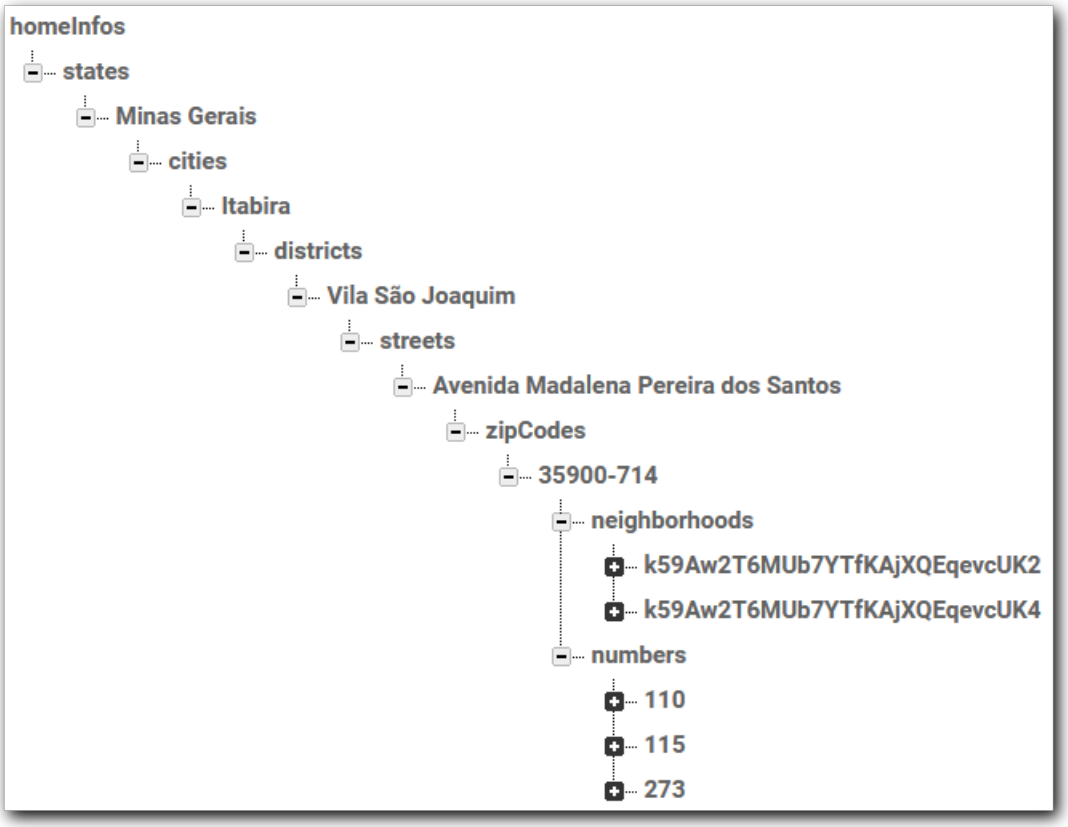
Fonte: elaborada pelo autor (2021).

3.3.2 Endereços no Banco de Dados

No Firebase Realtime Database, os dados são salvos na forma de uma estrutura de árvore JSON. É possível utilizar essa estrutura para tornar a pesquisa dos dados de endereço mais eficazes. Cada parte da informação do endereço é colocada um nível de uma arvore n-ária, onde é possível agrupar os endereços com relativa proximidade, em um ramo específico da árvore, conforme representado na Figura 11. Os endereços e as

vizinhanças ficam agrupadas de forma natural, tornando a pesquisa de vizinhos mais fácil de ser obtida, pois, as pessoas e vizinhanças localizadas no mesmo CEP são agrupadas nas folhas da árvore, assim como os grupos de vizinhança.

Figura 11 – Arvore de endereços.



Fonte: elaborada pelo autor (2021).

Para que a recuperação de dados seja feita de forma mais simples e rápida, o caminho percorrido nos ramos da árvore para armazenar o endereço e o grupo da vizinhança de cada usuário, fica armazenado para futuras consultas, tornando pontual a busca dessas informações, assim que necessárias.

3.4 Implementação

Nesta seção, será apresentada a implementação do aplicativo proposto que foi nomeado temporariamente como `Vizinhança Segura`. O aplicativo foi desenvolvido utilizando o *framework* Flutter, escolha feita devido às facilidades oferecidas, sendo elas os *Widgets* (elementos pré prontos), que auxiliaram em um rápido desenvolvimento do aplicativo e uma vasta biblioteca de pacotes, contendo ferramentas que auxiliam no desenvolvimento.

Essas bibliotecas contêm integrações com plataformas Application Programming Interfaces (APIs) e algumas funções especiais, que irão facilitar muito o desenvolvimento do aplicativo. Foi escolhido o Android como a plataforma central para o desenvolvimento deste trabalho, devido a uma série de fatores, dentre eles:

- Poder programar e testar o aplicativo em computadores que possuem como SO o Windows, sendo mais populares e acessíveis aos desenvolvedores devido ao seu preço geralmente mais baixo que plataformas da Apple.
- Poder testar aplicativos em vários dispositivos utilizando os Dispositivos Virtuais Android. Neste trabalho foram utilizados as seguintes AVDs: Pixel XL da Google, utilizando o Android 10 e o Pixel 4 da Google utilizando o Android 11.

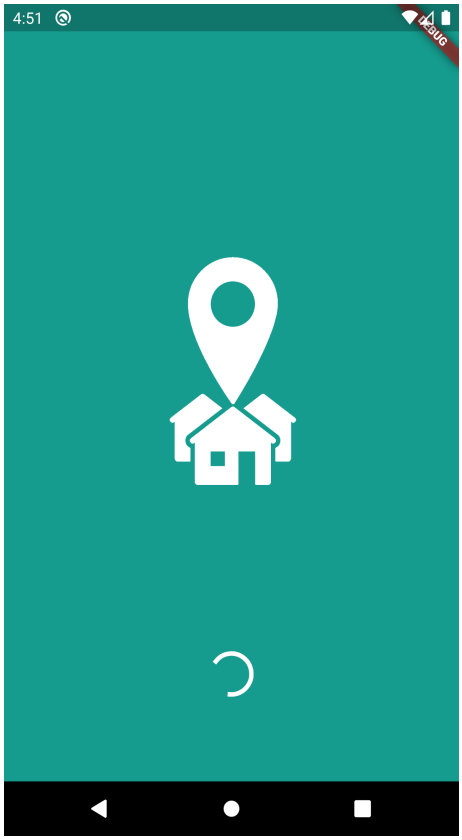
Nas subseções a seguir, serão apresentadas as telas do Vizinhança Segura e serão ressaltadas qualquer biblioteca ou tecnologia utilizada quando necessário. O fluxo ocorrerá da forma que um usuário utilizaria o aplicativo e os dados utilizados são fictícios.

3.4.1 Splash Screen

Esta tela é a primeira a ser exibida quando abrimos o aplicativo e indica que o mesmo esta carregando, transitando para a Tela de Login ou para a Tela de Principal, caso o usuário esteja logado. Esta transição foi personalizada com o uso da biblioteca *Flutter Page Transition Package*[3], que facilita a implementação do efeito da transição. Ela está representada na Figura 12

[3] Disponível em: https://pub.dev/packages/page_transition

Figura 12 – Splash Screen.



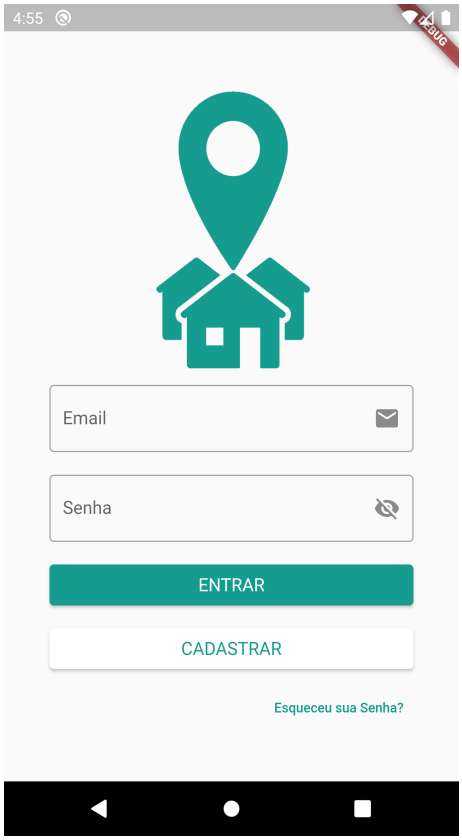
Fonte: elaborada pelo autor (2021).

3.4.2 Autenticação de Usuários

Como o acesso ao aplicativo Vizinhança Segura é restrito para moradores de um endereço, então é necessário proteger as informações desse endereço e de sua vizinhança. Para ter acesso ao aplicativo, o usuário tem que se registrar na Tela de Cadastro e, caso tenha esquecido sua senha, é possível recuperar a mesma através da Tela de Recuperação. Mas, de uma forma ou outra, o mesmo precisa de inserir informações de *e-mail* e senha para poder ter acesso ao aplicativo, passando pela Tela de Login, conforme representado Figura 13 e clicar no botão Entrar para validar suas informações de entrada. O serviço de autenticação é fornecido pelo Firebase Authentication e a integração com o aplicativo é feito pela biblioteca *Firebase Auth for Flutter*[4].

[4] Disponível em: https://pub.dev/packages/firebase_auth

Figura 13 – Tela de *login* do usuário.



Fonte: elaborada pelo autor (2021).

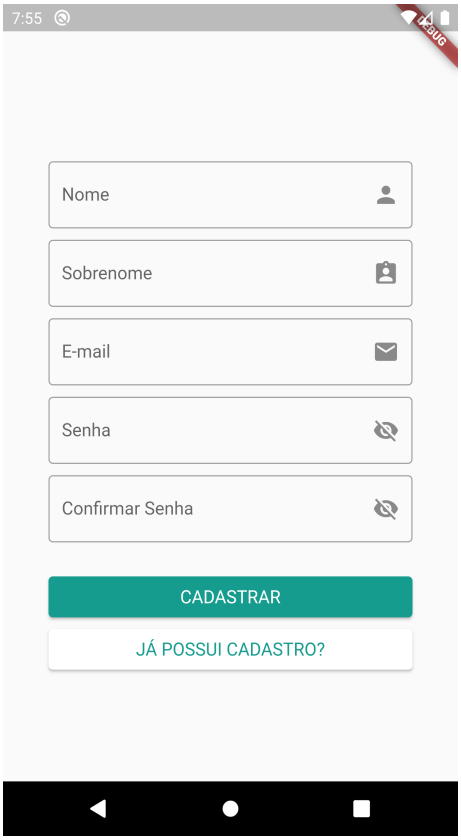
Os campos de *e-mail* e senha possuem um validador caso o usuário deixe os campos em branco e se as informações apresentadas forem incorretas, o aplicativo irá informar. O ícone localizado dentro do campo senha é um botão que serve para mostrar ou ocultar a senha digitada, caso o usuário necessite conferir as informações inseridas.

3.4.3 Cadastro de Usuários

Para que o usuário tenha acesso ao aplicativo, o mesmo deve se registrar na Tela de Cadastro. Nesta tela, ele deve fornecer informações de nome, sobrenome, *e-mail* e senha para realizar seu cadastro. Este campos estão dispostos conforme o representado na Figura 14 e, ao clicar no botão Cadastrar, o aplicativo realiza a validação dos dados. O Firebase Authentication provê a criação da nova conta do usuário, armazenando em seu banco de dados, as informações de nome, *e-mail* e senha. O sobrenome é armazenado no serviço Firebase Realtime Database, já que esse campo não esta presente no Authentication. Para fazer essa inserção no banco de dados do Realtime, o aplicativo conta com o auxilio da biblioteca *Firebase Database Plugin for Flutter*[5].

[5] Disponível em: https://pub.dev/packages/firebase_database

Figura 14 – Tela de cadastro do usuário.



Fonte: elaborada pelo autor (2021).

Todos os campos contam com uma validação de campo em branco e cada campo tem um validador próprio. O campo Nome tem um validador que limita um nome com no máximo trinta caracteres, assim como o de Sobrenome que está limitado a cinquenta caracteres. O campo "E-mail" conta com um validador provido pela biblioteca *Email validator*[6], que checa se o endereço obedece o padrão vistos em um endereço de *e-mail* padrão utilizando Regular Expression (RegEx). O campo Senha e Confirmar Senha contam com um validador que utiliza RegEx para validar se o usuário digita uma senha contendo no mínimo oito caracteres, símbolos especiais, números, letras maiúsculas e minúsculas, assim evitando senhas inseguras. Também conta com um botão para mostrar ou ocultar a senha caso o usuário deseje ver se digitou sua senha corretamente.

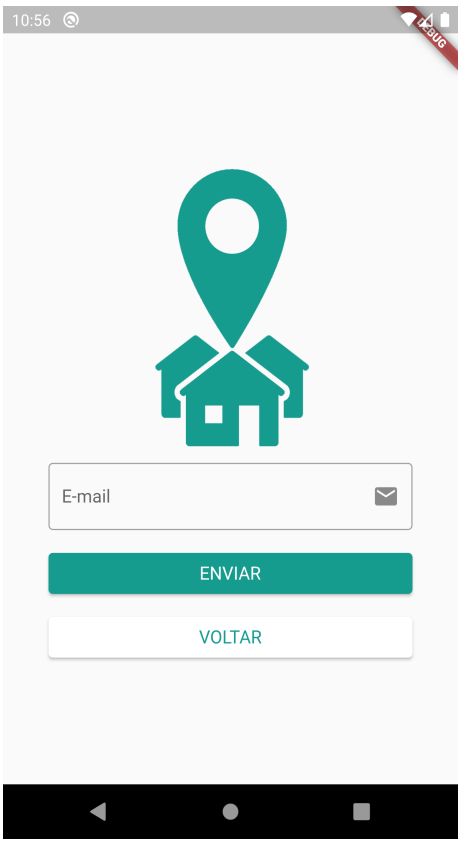
3.4.4 Recuperação de senha

Quando um usuário esquece sua senha de acesso ao aplicativo, ele deve acessar a Tela de Recuperação de Senha, representado pela Figura 15. O usuário deve preencher seu endereço de *e-mail* cadastrado no aplicativo. Em seguida, deve pressionar o botão

[6] Disponível em: https://pub.dev/packages/email_validator

Enviar, para receber um *e-mail* enviado pelo serviço Firebase Authentication contendo a informação de recuperação de senha e um *link* para realizar essa ação.

Figura 15 – Tela de recuperação de senhas.



Fonte: elaborada pelo autor (2021).

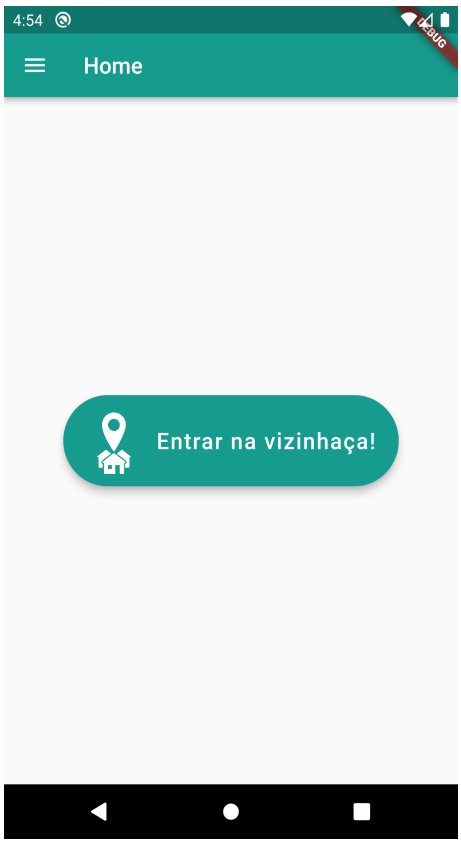
3.4.5 Tela inicial do usuário

A tela inicial conta com um menu superior, contendo o nome da página e um ícone no canto superior esquerdo e um botão animado no meio da tela escrito "Entrar na vizinhança" conforme representado na Figura 16. Quando apertamos o ícone no menu superior, o menu lateral se revela sobre a tela inicial, mostrando algumas informações sobre o usuário e alguns ícones utilizados para acessar as funções implementadas no aplicativo, conforme representado na Figura 17.

No momento em que o aplicativo abre a Tela inicial, o mesmo faz uma busca no banco de dados do Firebase Realtime Database para verificar se existe algum *token* registrado, do atual aparelho em que o usuário está utilizando. Caso não exista, ele irá registrá-lo no banco de dados. Caso exista um registro, mas não seja igual ao encontrado (cenário de mudança de *smartphone* ou desinstalação do aplicativo) atualmente, o mesmo é atualizado no banco de dados com o novo *token* encontrado. Caso exista um registro e o

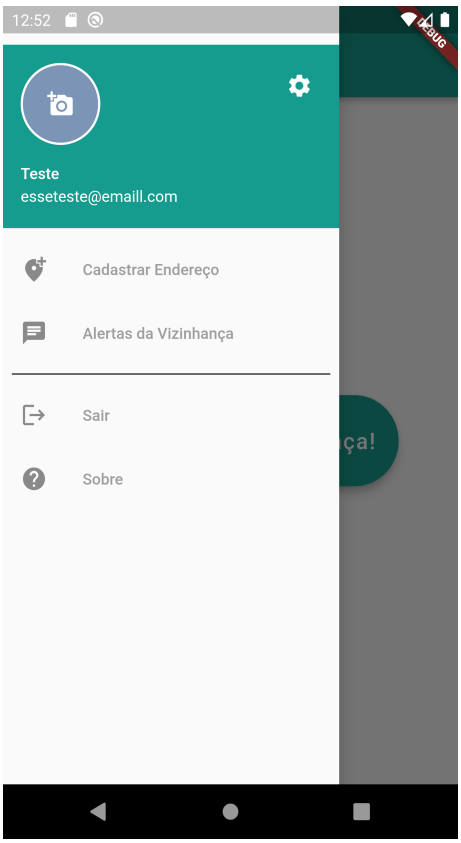
mesmo seja igual ao *token* encontrado no aplicativo, então, nenhuma ação é tomada pelo aplicativo.

Figura 16 – Tela inicial.



Fonte: elaborada pelo autor (2021).

Figura 17 – Tela inicial com menu lateral.



Fonte: elaborada pelo autor (2021).

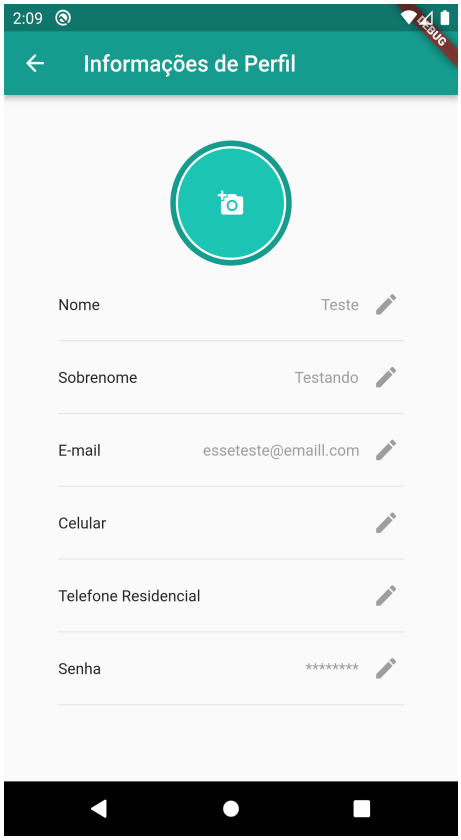
O botão no meio da tela inicial tem uma dupla função. Se o usuário já cadastrou seu endereço no aplicativo, o botão possibilita o acesso à tela que mostra a vizinhança em que o mesmo se enquadra. Por outro lado, caso ainda não tenha cadastrado seu endereço, o botão possibilita o acesso à tela de cadastro de endereço, da mesma forma que a opção listada no menu lateral "Cadastrar Endereço". No menu lateral, temos as informações de nome e *e-mail* do usuário, assim como uma foto de perfil do mesmo. No ícone de engrenagem, o usuário tem acesso às demais informações do usuário e, também, poderá editar elas. Na lista do menu lateral, temos acesso aos "Alertas da Vizinhança", o "Sobre" e uma opção para desconectar do aplicativo no item "Sair".

3.4.6 Informações de perfil do usuário

Nesta tela, temos todas as informações que o usuário já cadastrou no aplicativo, além de informações opcionais que o mesmo pode cadastrar como o número de celular,

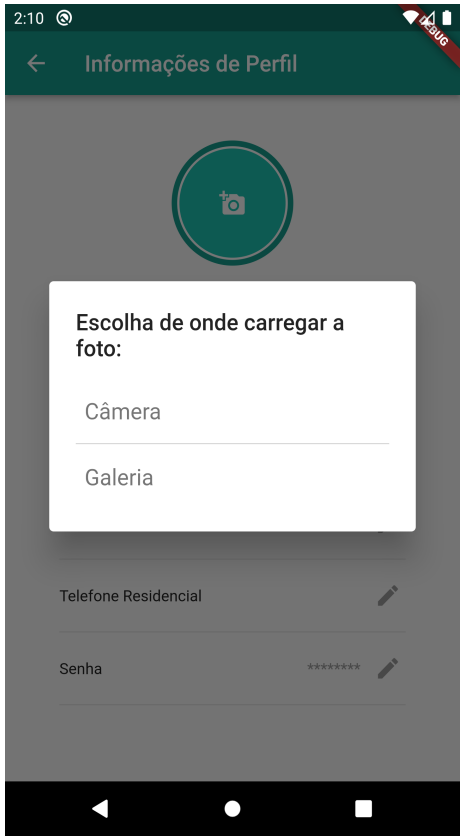
o telefone residencial e uma foto de perfil, conforme representado na Figura 18. Caso o usuário queira carregar uma foto, ele deve clicar no ícone que representa uma câmera fotográfica, abrindo uma tela que sobrepõe a do perfil, conforme representado na Figura 19, para que o mesmo escolha entre tirar uma foto na hora com a câmera ou pegar uma foto do armazenamento interno do aparelho. Esta foto será armazenada em nuvem no Firebase Cloud Storage com o auxilio da biblioteca *Cloud Storage for Flutter*[7].

Figura 18 – Tela de perfil e edição.



Fonte: elaborada pelo autor (2021).

Figura 19 – Escolha entre câmera ou galeria.

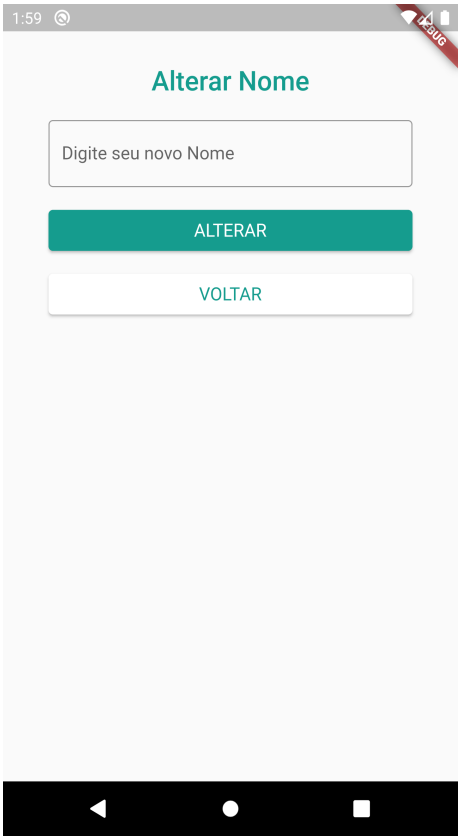


Fonte: elaborada pelo autor (2021).

Caso o usuário queira alterar as demais informações, o mesmo deve pressionar o ícone de lápis ao lado da informação que ele deseja alterar, para então ter acesso a uma tela personalizada de alteração para cada campo conforme representado na Figura 20. Estes campos têm os mesmos validadores encontrados na área de cadastro de usuário, sendo validados quando o usuário clicar no botão Alterar.

[7] Disponível em: https://pub.dev/packages/firebase_storage

Figura 20 – Tela de edição de informação.

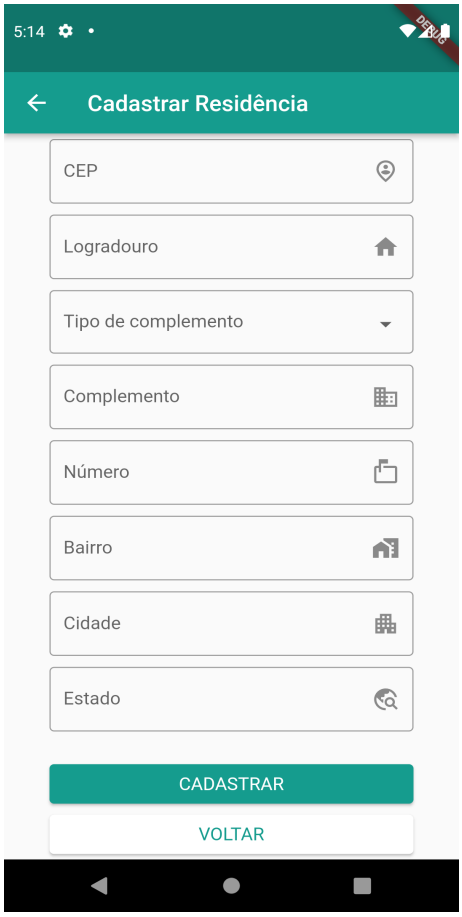


Fonte: elaborada pelo autor (2021).

3.4.7 Cadastrar endereço do usuário

Para o usuário cadastrar seu endereço no aplicativo, o mesmo deve fornecer as informações de CEP, logradouro (Rua, avenida, travessa, etc), tipo de complemento (Casa ou Apartamento), complemento (Vazio caso não possua, A, B, para casas e o número do apartamento, para prédios), número da residência, cidade e estado, conforme representado na Figura 21.

Figura 21 – Tela de cadastro de endereço.



Fonte: elaborada pelo autor (2021).

No momento em que o usuário digita o CEP de sua residência, o aplicativo irá procurar as demais informações de seu endereço com base no CEP digitado. Utilizando a API *Postmon*[8], com o auxílio da biblioteca *search_cep*[9], todas as informações do endereço que este CEP fornecem são carregadas e preenchidas nos devidos campos, cabendo ao usuário digitar as demais informações. Após isso, essas informações são armazenadas no banco de dados do Firebase Realtime Database.

3.4.8 Vizinhança segura

A Vizinhança neste trabalho é definida quando o usuário faz parte de uma grupo de vizinhos delimitados por um raio de proximidade. O raio escolhido foi de quinze metros, onde o centro deste raio é o primeiro morador a cadastrar seu endereço em uma região onde não existia previamente uma Vizinhança, sendo todos os demais vizinhos dentro

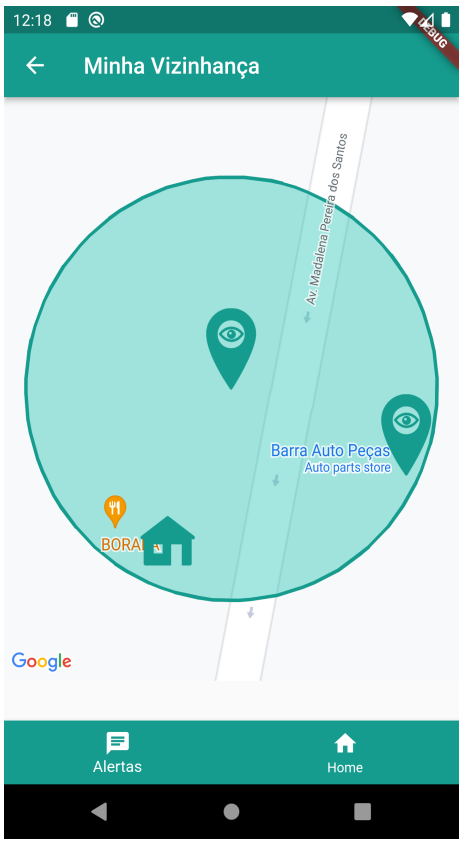
[8] Disponível em: https://postmon.com.br/
[9] Disponível em: https://pub.dev/packages/search_cep

dessa área enquadrados neste grupo de vizinhança. A localização desses usuários é feita através das informações que os mesmos inseriram ao cadastrar seu endereço.

Para fazer esses cálculos, é necessária a localização geográfica de Latitude e Longitude desses endereços. Essas informações são fornecidas pela API *Google Places*[10]. O cálculo utilizado é o de Distância Euclidiana, que mede a distância entre dois pontos, sendo enquadrados na Vizinhança, aqueles que apresentam uma distância ao centro do raio menor que quinze metros. Esses cálculos de distância entre dois pontos são facilitados pela biblioteca *An Android Google Maps Utils original java project port for Flutter*[11] que calcula em metros a distância entre dois pontos geográficos obtidos através da Latitude e Longitude .

Quando necessárias, essas informações são pesquisadas com o uso da API e armazenadas no banco de dados do Firebase Realtime Database, para que no futuro sejam utilizadas sem a necessidade de chamadas de API.

Figura 22 – Tela de vizinhança.



Fonte: elaborada pelo autor (2021).

Quando o usuário aperta o botão “Entrar na vizinhança” na tela inicial, já tendo

[10] Disponível em: https://developers.google.com/maps/documentation/places/web-service/overview
[11] Disponível em: https://pub.dev/packages/google_maps_flutter

seu endereço cadastrado, o aplicativo faz uma verificação e checa se o usuário já pertence a uma Vizinhança. Caso o usuário não faça parte de uma Vizinhança e, no local de seu endereço, já existe uma Vizinhança previamente formada, ele será adicionado nesta. Caso o usuário não faça parte de uma Vizinhança e em seu endereço não exista uma Vizinhança previamente formada, o aplicativo irá criar uma nova, tendo sua posição geográfica como o centro deste novo grupo de vizinhos. Caso pertença a uma Vizinhança, as informações são carregadas e mostradas em tela em uma mapa obtido com o uso do SDK do Maps para Android[12], com o auxílio da biblioteca *Google Maps for Flutter*[13], mostrando neste mapa a localização do endereço do usuário e de seus vizinhos que fazem parte do grupo da vizinhança, conforme representado na Figura 22.

O circulo no mapa indica o raio de quinze metros a partir do primeiro usuário cadastrado na vizinhança. Os marcadores que possuem a marca personalizada do aplicativo, representam os vizinhos da Vizinhança, enquanto o marcador de casa indica a localização do usuário que está atualmente conectado no aplicativo. Os marcadores dos vizinhos também são botões clicáveis utilizados para reportar alguma atividade suspeita que possa estar acontecendo na residência deste vizinho.

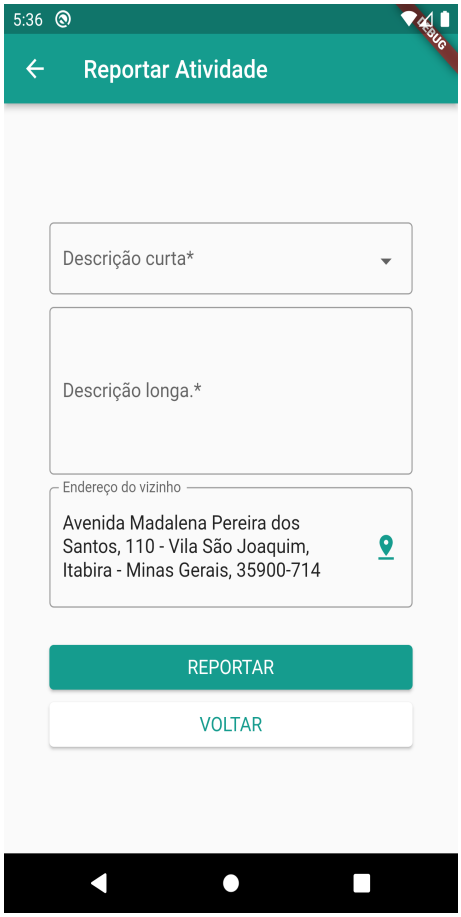
3.4.9 Reportar atividades suspeitas

Esta ferramenta é utilizada para relatar alguma atividade suspeita na residência de um vizinho. Esta tela contém uma descrição curta da atividade, que são algumas opções já pré estabelecidas pelo aplicativo (Atividade Suspeita, Depredação, Furto, Invasão e Outro), área para Descrição longa, onde o usuário relata em texto a atividade que ele está observando e também o endereço deste vizinho, para ajudar na identificação do mesmo. Esta tela está representada na Figura 23.

Quando o usuário aperta o botão “Reportar”, ele registra as informações do alerta no banco de dados Firebase Realtime Database contendo os campos preenchidos acima, juntamente com as informações do id do usuário, o id do vizinho e uma amostra de tempo do momento em que o alerta for enviado. Para enviar a notificação, o aplicativo faz uso do *token* do aplicativo do vizinho. Esse *token*, recuperado do banco de dados Firebase Realtime Database, permite ao aplicativo enviar para esse vizinho a notificação de que algum evento está acontecendo em sua residência, conforme relatado na Figura 24.

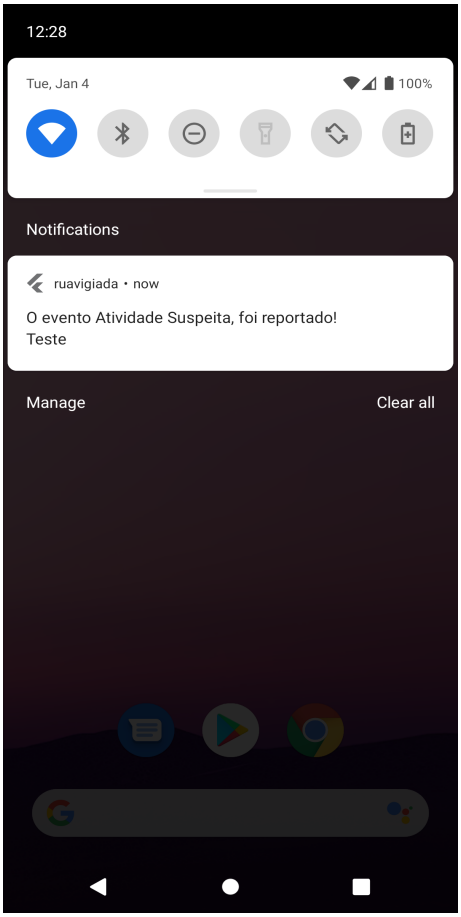
[12] Disponível em: https://developers.google.com/maps/documentation/android-sdk/overview
[13] Disponível em: https://pub.dev/packages/google_maps_flutter

Figura 23 – Tela para relato de atividade suspeita.



Fonte: elaborada pelo autor (2022).

Figura 24 – Notificação recebida pelo vizinho.

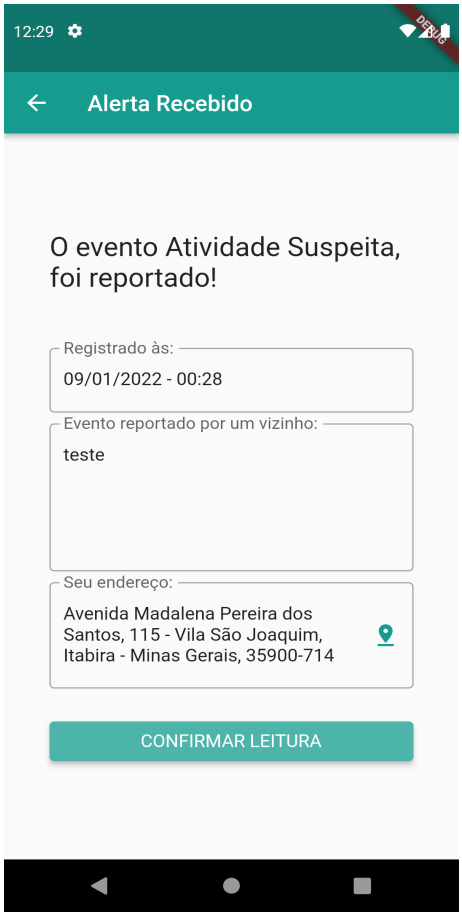


Fonte: elaborada pelo autor (2022).

Com o uso da ferramenta *Firebase Cloud Messaging* e com o auxílio da biblioteca *Firebase Messaging Plugin for Flutter*[14], o aplicativo consegue enviar e receber mensagens do Firebase e de outros usuários. Quando o aplicativo é programado para tomar alguma ação quando o usuário clica em uma notificação, o mesmo pode abrir telas especiais que carregam as informações transportadas por essas notificações. No caso do aplicativo desenvolvido neste trabalho, a notificação abre uma tela, conforme relatado na Figura 25, contendo as informações que um usuário relatou ao outro, que foram registradas no momento do alerta.

[14] Disponível em: https://pub.dev/packages/firebase_messaging

Figura 25 – Alerta recebido pelo vizinho.



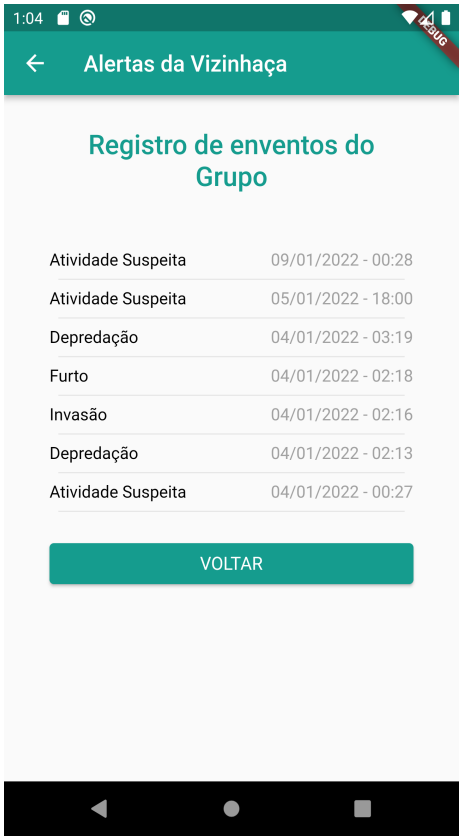
Fonte: elaborada pelo autor (2022).

Ao confirmar a leitura, o usuário afirma ter tomado consciência da notificação de atividade suspeita, sendo registrado posteriormente no banco de dados *Firebase Realtime Database* essa confirmação.

3.4.10 Lista de notificações da Vizinhança

Neste aplicativo é possível ver todo o histórico de atividades suspeitas relatadas pelos usuários dentro de um grupo de vizinhos. As atividades ficam organizadas em uma lista apresentada na "Tela de Alertas da Vizinhança", em duas colunas contendo a descrição curta do relato e a data e hora em que o evento foi notificado, conforme relatado na Figura 26.

Figura 26 – Lista de eventos reportados.



Fonte: elaborada pelo autor (2022).

3.5 Testes

Os testes de um software têm como função mostrar para seu desenvolvedor se existe algum tipo de mal funcionamento na execução de seu aplicativo e se o mesmo atende a todos os requisitos e cumpre sua função planejada. Servem também para que se tente entregar o produto final aos clientes sem nenhum defeito aparente e que seu uso possa ser feito sem interrupções e problemas. Segundo Sommerville (2019), os testes fazem parte de um processo de dois fatores muito comumente confundidos, que são a verificação (o produto está sendo feito de forma correta?) e a validação (o produto construído é o certo?). Esses processos estão associados a conferência do software, para garantir que o mesmo seja construído conforme as especificações levantadas, se o mesmo está atendendo a proposta inicial e se as expectativas do cliente estão sendo atendidas.

3.5.1 Testes no Firebase Test Lab

Para este trabalho, foi utilizada a ferramenta de testes *Firebase Test Lab*, que conta a ferramenta de testes automatizados Robo, que simula de forma automática, as

atividade de um usuário dentro do aplicativo. O Robo registra suas ações de diversas formas, um diagrama que registra de forma sequencial, as ações executadas durante o teste, vídeos e capturas de tela para demostrar o fluxo de ações, detecta e registra qualquer problema com a interface do aplicativo e também aponta qualquer tipo de problema de Acessibilidade encontrado. Ao fim dos testes, são exibidos os resultados, mostrando as informações produzidas, assim como erros e problemas encontrados no decorrer do tempo. No plano gratuito do Firebase, são permitidos realizar testes em até quinze dispositivos por dia, sendo dez execuções em dispositivos virtuais (emulando dispositivos reais), com limite total de sessenta minutos somando o uso de todos esses dispositivos e cinco execuções em dispositivos físicos (dispositivos reais), com limite total de quarenta e cinco minutos, somando o uso de todos esses dispositivos.

Na realização dos testes, o aplicativo foi compilado em código executável para a plataforma Android, também conhecido como Android Application Pack (APK), para, em seguida, ser carregado na plataforma Test Lab para a realização dos testes. Nesta versão do aplicativo compilado, foi modificado a questão da autenticação, sendo preenchido de antemão os campos de "E-mail" e "Senha" para não impedir o acesso das *features* protegidas. Os testes foram feitos em várias amostras, em momentos diferentes, e os mesmos foram registrados pela plataforma, conforme mostrado na Figura 27.

Figura 27 – Captura de tela da plataforma de testes.

Fonte: elaborada pelo autor (2022).

Os possíveis resultados para os teste do Test Lab, são: `Aprovado`, ou seja, sem nenhuma falha encontrada; `Falha`, ou seja, pelo menos uma falha encontrada e `Inconclusivo`, ou seja, os resultados foram incompletos devido à um erro da plataforma. O resultado diferente mostrado na Figura 27, é o ignorado, que quer dizer que o teste não foi reali-

zado pelo aparelho, pois a versão do Android escolhido não é compatível com a mínima demandada pelo aplicativo.

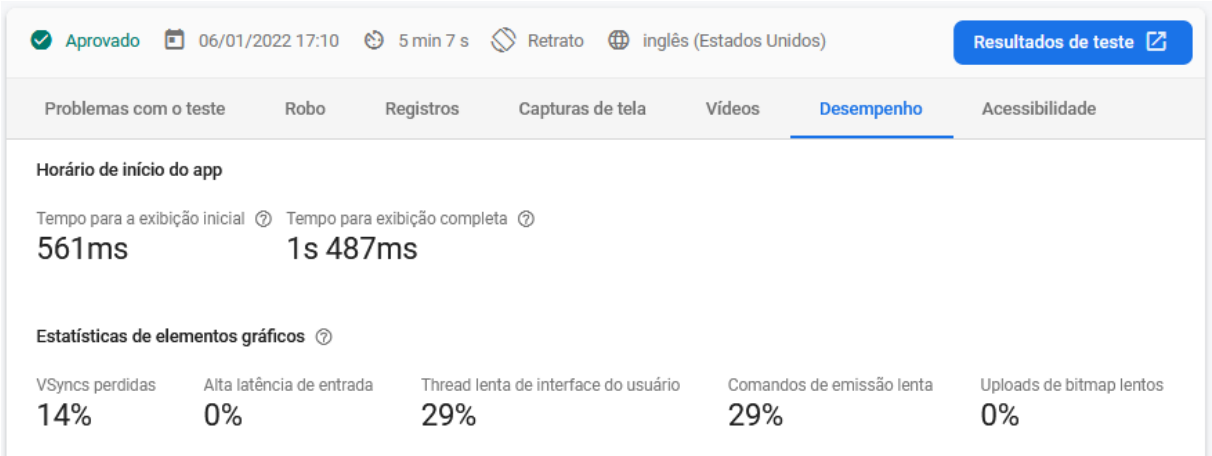
Tabela 2 – Tabela resultado de testes em vários contextos.

Aparelho	Plataforma	Versão	Duração	Resultado
Google Pixel 3	Real	Android 9	5min 6 s	Aprovado
Google Pixel 3	Virtual	Android 11	5 min 14 s	Aprovado
Google Pixel 4	Real	Android 10	5 min 7 s	Aprovado
Google Pixel 3	Real	Android 9	5min 6 s	Aprovado
Google Pixel 2	Virtual	Android 11	5 min 14 s	Aprovado
Moto E5 Play	Real	Android 8.1	5 min 11 s	Aprovado
Google Pixel 2	Virtual	Android 11	2 min 36 s	Aprovado
Google Pixel 3	Real	Android 9	5 min 5 s	Aprovado

Fonte: elaborada pelo autor (2022)

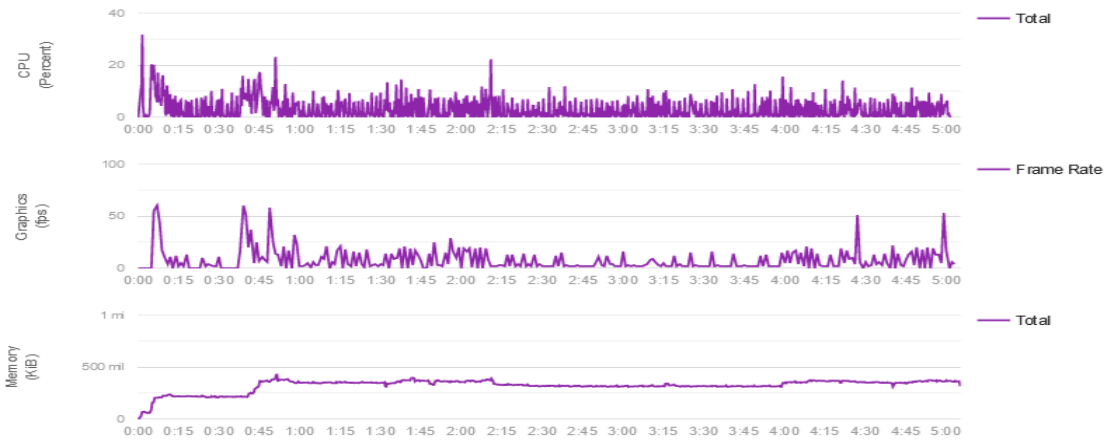
Conforme mostrado na Tabela 3, esses são os aparelhos, tipos de dispositivo e resultado de testes realizados pela plataforma, mostrando que nesses contextos escolhidos, o aplicativo consegue ser executado e efetuar suas ações sem nenhum problema maior. Para maior detalhamento, serão apresentados os resultados obtidos na execução do aplicativo no dispositivo Google Pixel 4. O desempenho geral foi registrado conforme a Figura 28. Na Figura 29, foram apresentados alguns gráficos de desempenho obtidos durante a execução.

Figura 28 – Captura de tela do desempenho.



Fonte: elaborada pelo autor (2022).

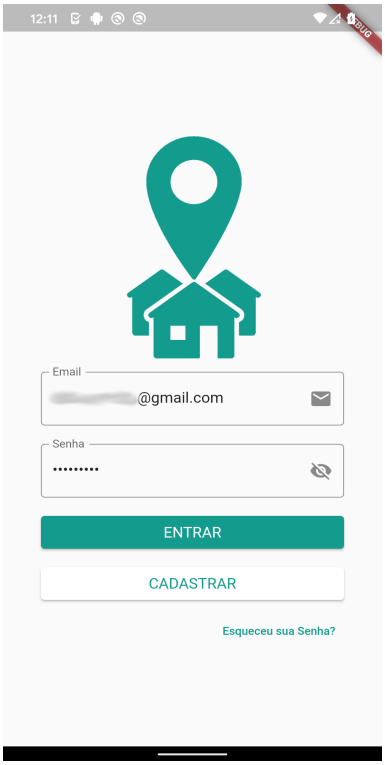
Figura 29 – Gráfico de desempenho.



Fonte: elaborada pelo autor (2022).

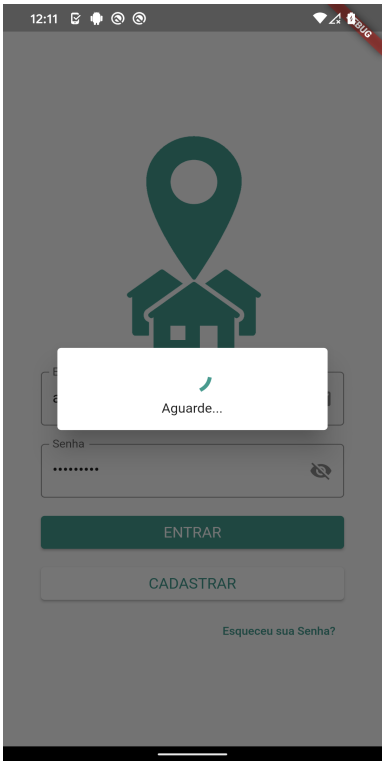
Neste contexto de teste, foram registradas capturas de tela mostrando as interações e telas percorridas pelo Robo. As figuras 30, 31, 32 e 33 mostram algumas destas sequências de telas percorridas.

Figura 30 – Captura de tela dados de acesso.



Fonte: elaborada pelo autor (2022).

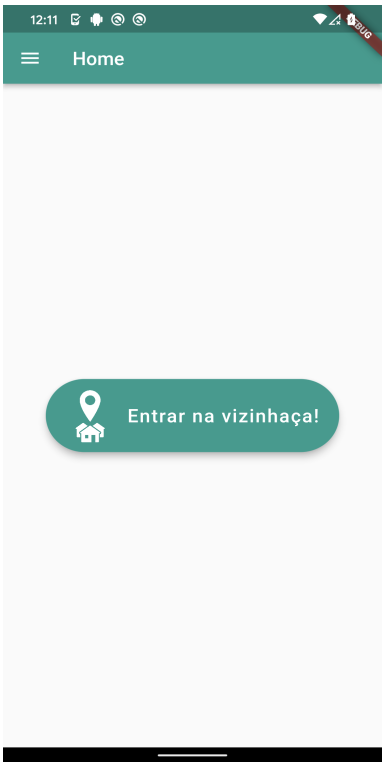
Figura 31 – Captura de tela efetuando login.



Fonte: elaborada pelo autor (2022).

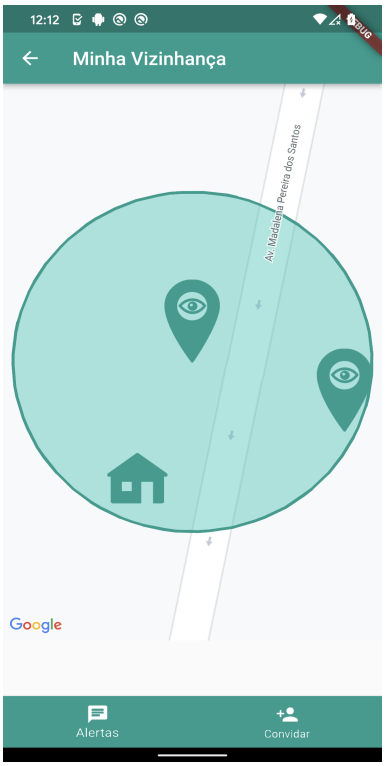
A Figura 30 mostra as informações de "E-mail" e "Senha" previamente preenchidas, sendo censurada a informações de *e-mail*. Em seguida, o botão "Entrar" foi apertado e as informações do aplicativo foram carregadas, pedindo ao usuário para aguardar o fim do carregamento, conforme mostrado na Figura 31.

Figura 32 – Captura de tela inicial.



Fonte: elaborada pelo autor (2022).

Figura 33 – Captura de tela da Vizinhança.



Fonte: elaborada pelo autor (2022).

Em seguida, o Robo interagiu com o botão "Entrar na vizinhança!", conforme mostrado na Figura 32, carregando as informações da Vizinhança armazenadas com o usuário dado para a realização do teste representado na Figura 33.

3.6 Considerações finais

Após o desenvolvimento do aplicativo, foram apresentados na Tabela 3 um comparativo de algumas *features*, presentes nos aplicativos semelhantes, junto ao desenvolvido neste trabalho. O aplicativo `Vizinhança Segura` traz alguns benefícios semelhantes aos demais apresentados na tabela, e outros que podemos ressaltar como:

- A possibilidade de alertar diretamente os vizinhos de atividades suspeitas;

- A presença de *features* semelhantes das apresentados em aplicativos que não estão presentes nacionalmente (formação de grupos com base em posição geográfica);
- Conceder acesso as informações de atividades suspeitas somente para quem deva ter acesso a essa informação.

Tabela 3 – Tabela comparativo de *features* com aplicativos semelhantes.

Nome	My Patrimonial Segurança	Vigilancia Solidária	Neighbors by Ring	Nextdoor	Vizinhança Segura
Multiplataforma	Sim	Sim	Sim	Sim	Sim
F-Droid	Não	Não	Não	Não	Não
Web	Sim	Sim	Sim	Sim	Não
Pago	Sim	Sim	Não	Não	Não
Acesso a Câmeras	Sim	Sim	Não	Não	Não
Controlar Alarme	Sim	Não	Não	Não	Não
Alerta de Vizinho	Não	Não	Não	Não	Sim

Fonte: elaborada pelo autor (2022)

4 Conclusão

Este trabalho apresentou o desenvolvimento de um aplicativo multiplataforma denominado `Vizinhança Segura`.

Após a realização de uma revisão bibliográfica, apresentou-se conceitos de sistemas de informação, a importância de técnicas de desenvolvimento providos pela Engenharia de Software, conceitos de sistemas colaborativos, a relevância do desenvolvimento de aplicativos para dispositivos móveis, os sistemas operacionais mais populares para *smartphones*, tipos de bancos de dados não relacionais e suas características, o que são *frameworks* e algumas aplicações correlatas e uma análise das *features* encontradas nos mesmos.

Na etapa seguinte, foi abordado o desenvolvimento do aplicativo e os resultados. Foram abordadas as tecnologias e ferramentas utilizadas no desenvolvimento do trabalho. Em seguida, foram mostrados os requisitos levantados necessários para a construção do aplicativo. Tendo esses requisitos, foi a vez de fazer a prototipação do *design* e fluxo de telas. Foram apresentadas as telas do aplicativo com figuras, explicando o processo de desenvolvimento e as tecnologias utilizadas em cada uma delas, sendo observado os requisitos levantados, assim como a modelagem do banco de dados apresentada. Por fim, foi realizada a validação de todo aplicativo com os testes realizados em múltiplos aparelhos móveis da plataforma Android com o uso da ferramenta `Test Lab` e apresentação de seus resultados.

Dessa forma, foram alcançados os objetivos apresentados para o desenvolvimento de um aplicativo móvel multiplataforma para a gestão de segurança patrimonial, o `Vizinhança Segura`. Diante da necessidade de aprofundamento e adição de novas *features* ao aplicativo, ficam a seguir algumas propostas de trabalhos futuros.

4.1 Trabalhos futuros

No decorrer do desenvolvimento deste trabalho, foram levantados algumas questões que poderão ser utilizadas em propostas para trabalhos futuros, conforme apresentado abaixo:

- Realização de testes em dispositivos da Apple, no sistema operacional IOS;
- Adição de uma formar de analisar os dados obtidos nos registros das atividades suspeitas dentro das vizinhanças;
- Realizar um estudo de qual o tamanho ideal para definir o raio de uma vizinhança, com base em casos reais;

- Adição de outras funcionalidades ao aplicativo, como adicionar vários moradores em um mesmo endereço através de convites privados e também adicionar uma lista de contatos de segurança para serem notificados no lugar dos moradores, no caso da ausência dos mesmos;
- Adicionar fotos das fachadas das casas (caso queiram os usuários) a fim de facilitar a identificação do endereço do vizinho;
- Adequar o tratamento de dados do aplicativo levando em conta a Lei Geral de Proteção de Dados Pessoais (LGPD), assim como produzir um termo de aceite para o usuário, identificando quais informações deverão ser fornecidas para o uso do aplicativo;
- Avaliação de formas de disponibilização do aplicativo nas principais lojas.

Referências

ANDRADE, A. P. de. *O que é Flutter?* 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-flutter>. Acesso em: 20 dez. 2021. Citado na página 30.

AVER, H. *Por que você deve evitar aplicativos desconhecidos*. 2021. Disponível em: <https://www.kaspersky.com.br/blog/unknown-apps-android/18082/>. Acesso em: 20 dez. 2021. Citado na página 20.

CEGIELSKI, C. G.; RAINER JUNIOR, R. K. *Introdução a Sistemas de Informação*. 5. ed. São Paulo: [s.n.], 2015. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595156166/>. Acesso em: 23 nov. 2021. Citado 2 vezes nas páginas 17 e 19.

EMíDIO, L. M. de S. *Análise dos reflexos na segurança pública com a implementação do Projeto Rede Vizinhos Protegidos pela 133ª Companhia da Polícia Militar do 18º Batalhão da Polícia Militar de Minas Gerais*. Monografia (Curso de Especialização) — Fundação João Pinheiro, 2011. Disponível em: <http://monografias.fjp.mg.gov.br/handle/123456789/1799>. Acesso em: 04 jun. 2021. Citado na página 15.

FRAGA, T. L. *Qual o impacto do crime para as vítimas? Uma análise considerando a influência dos roubos e furtos na percepção de segurança e migração no Brasil*. Dissertação (Mestrado) — Universidade Federal de Viçosa, Viçosa, nov. 2015. Disponível em: <https://www.locus.ufv.br/handle/123456789/9641>. Acesso em: 04 jun. 2021. Citado na página 14.

GARCIA, A. C. B. et al. Groupware 4.0: Avanços e desafios da computação social. *Jornada de Atualização em Informática*, v. 39, n. 4, p. 142–186, 2020. Disponível em: <https://sol.sbc.org.br/livros/index.php/sbc/catalog/book/57>. Acesso em: 18 jan. 2022. Citado na página 18.

LAUDON, J. P.; LAUDON, K. C. *Sistemas de Informação Gerenciais*. 11. ed. São Paulo: [s.n.], 2014. Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/22448/pdf/0>. Acesso em: 23 nov. 2021. Citado na página 20.

MAXIM, B. R.; PRESSMAN, R. S. *Engenharia de software*: uma abordagem profissional. 9. ed. Porto Alegre: [s.n.], 2021. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9786558040118/>. Acesso em: 27 nov. 2021. Citado 2 vezes nas páginas 17 e 18.

PIMENTA, R. *Por que você deve evitar aplicativos desconhecidos*. 2020. Disponível em: <https://geekblog.com.br/o-que-e-ios-saiba-mais-sobre-esse-sistema-da-apple/>. Acesso em: 20 dez. 2021. Citado na página 20.

PIMENTEL, M.; FUKS, H. *Sistemas Colaborativos*. 1. ed. Rio de Janeiro: [s.n.], 2011. Disponível em: <https://sistemascolaborativos.uniriotec.br/>. Acesso em: 18 jan. 2022. Citado na página 18.

SAUVé, J. P. *Tipos de frameworks*. 2021. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/tipos.htm>. Acesso em: 01 dez. 2021. Citado na página 22.

SOMMERVILLE, I. *Engenharia de software*. 10. ed. São Paulo: [s.n.], 2019. Disponível em: <https://plataforma.bvirtual.com.br/Acervo/Publicacao/168127>. Acesso em: 26 dez. 2021. Citado 4 vezes nas páginas 14, 32, 39 e 54.