INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO
DEPARTAMENTO ACADÊMICO DE INFORMAÇÃO E COMUNICAÇÃO
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DANIEL SANTOS DA SILVA

MAPES: APLICATIVO PARA MONITORAMENTO GEOGRÁFICO DE ALUNOS

MANAUS - AMAZONAS
2020

DANIEL SANTOS DA SILVA

MAPES: APLICATIVO PARA MONITORAMENTO GEOGRÁFICO DE ALUNOS

Trabalho de Conclusão de Curso apresentado à banca examinadora Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciências e Tecnologia do Amazonas (IFAM), Campus Manaus Centro, como requisito para o cumprimento da disciplina TCC II – Desenvolvimento de Software

Orientador: Prof. Dr. Jucimar Brito de Souza

MANAUS - AMAZONAS
2020

Biblioteca do IFAM - Campus Manaus Centro

S586m Silva, Daniel Santos da.
MAPES: aplicativo para monitoramento geográfico de alunos / Daniel Santos da Silva. – Manaus, 2020.
135 p. : il. color.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas). – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Centro, 2020.
Orientador: Prof. Dr. Jucimar Brito de Souza.

1. Desenvolvimento de sistemas. 2. Sistema de rastreamento. 3. Monitoramento geográfico. 4. Smartphone. I. Souza, Jucimar Brito de. (Orient). II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 005

Elaborada por Márcia Auzier CRB 11/597

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO MÉDIA E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA - AM.
DEPARTAMENTO ACADÊMICO DE INFORMAÇÃO E COMUNICAÇÃO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA AMAZONAS

Ata de Defesa de Trabalho Final de Graduação do (a) acadêmico (a) **Daniel Santos da Silva**, sobre o tema: "**MAPES: APLICATIVO PARA MONITORAMENTO GEOGRÁFICO DE ALUNOS**".

Ao **quinto** dia do mês de **outubro** de dois mil e vinte, às **16:00** horas, na sala do Google Meet, do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, realizou-se a Defesa Pública de Monografia do (a) formando (a) **Daniel Santos da Silva**, intitulada: "**MAPES: APLICATIVO PARA MONITORAMENTO GEOGRÁFICO DE ALUNOS**". A Banca Examinadora foi constituída pelos professores: **Prof. Dr. Jucimar Brito de Souza**, orientador (a) e presidente da mesma, (a) **Prof. MSc. Antonio Ferreira dos Santos Junior**, examinador (a), e **Prof. MSc. Emmerson Santa Rita da Silva**, examinador (a). A presidência da mesa deu início aos trabalhos, seguindo metodologia apropriada. Após a apresentação, arguição dos examinadores e defesa, a Banca Examinadora se reuniu para análise, deliberação e divulgação de nota atribuída ao trabalho, solicitando inserção e remoção de dados. A sessão foi encerrada às **17:00** horas. Eu, Profa MSc. Jorlene de Souza Marques, lavrei a presente ata, que depois de lida e aprovada, foi assinada por mim, pelos membros da Banca Examinadora e pelo (a) formando (a) **Daniel Santos da Silva**, a quem será conferido, ao final do corrente semestre, o grau de Tecnólogo (a) em Análise e Desenvolvimento de Sistemas.

Manaus, 05 de outubro de 2020.

Orientador(a): _____

Examinador (a): Emmerson Santa Rita da Silva

Examinador (a): _____

Formando (a): Daniel Santos

Secretário (a): Jorlene de Souza Marques

RESUMO

Geolocalização se refere à identificação da localização geográfica de um usuário usando algum dispositivo que disponha de mecanismo de coleta de dados (coordenadas) para estimar sua posição real. Normalmente, a geolocalização se utiliza de tecnologias, como GSM, GPS ou Wi-Fi para determinar esse posicionamento. Na última década, este conceito evoluiu bastante a ponto de tornar-se indispensável em vários sistemas e aplicativos e se tornou uma grande tendência com a popularização dos smartphones. Por causa disso, localizar pessoas utilizando um smartphone acabou se tornando uma alternativa para pais que desejam saber onde seus filhos estão, tendo em vista que essa preocupação é rotineira e angustiante. Como isso é uma situação corriqueira, faz-se necessário implantar soluções que possam auxiliar e tranquilizar os pais sobre o paradeiro de seus filhos. Algumas iniciativas já foram propostas com intuito de minimizar esse problema, como a utilização de microchips nos uniformes dos alunos e catracas biométricas que notificam aos pais se eles já chegaram na escola. Apesar de essas propostas solucionarem a situação, elas acabam sendo de alto investimento para implantá-las. Nessa perspectiva, este trabalho apresenta o desenvolvimento de um aplicativo móvel que permite que os pais possam receber notificações de quando os seus filhos chegam e saem da escola, utilizando os recursos de geolocalização disponíveis nesses dispositivos apoiado com API de mapas e BaaS (*Backend as a Service*) para a utilização dos serviços de *backend*.

Palavras-Chave. Geolocalização; Smartphone; Pais; Alunos; Escola;

ABSTRACT

Geolocation refers to the identification of the geographic location of a user using some device that has a data collection mechanism (coordinates) to estimate its real position. Typically, geolocation uses technologies such as GSM, GPS or Wi-Fi to determine this positioning. In the last decade, this concept has evolved to the point of becoming indispensable in many systems and applications and has become a big trend with the popularization of smartphones. Because of this, locating people by cell phone has become an alternative for parents who want to know where their children are, given that this concern is routine and harrowing. As this is an ordinary situation, it is necessary to implement solutions that can help and reassure parents about the whereabouts of their children. Some initiatives have already been proposed to minimize this problem, such as the use of microchips in students' uniforms and biometric turns that notify parents if they have already arrived at school. Although these proposals solve the situation, they end up being high investment to implement them. In this perspective, this work presents the development of a mobile application that allows parents to receive notifications when their children arrive and leave school, using the location resources available on devices supported with maps API and BaaS (Backend as a Service) for the use of backend services.

Keywords. Geolocation; Smartphone; Parents; Students; School.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API – *Application Programming Interface*

BaaS – *Backend as a Service*

Cell ID – *Cell Identifier*

CGI-TA – *Cell Global Identity-Timing Advance*

CMC – Campus Manaus Centro

CPM – Centro de Posicionamento Móvel

DGPS – *Differential Global Positioning System*

EDGE – *Enhanced Date Rates For GSM Evolution*

ERB – Estação Rádio Base

ERP – *Enterprise Resource Planning*

GPS – *Global Position System*

GSM – *Global System for Mobile*

GSMA – *Global System for Mobile Communications Association*

HLR – *Home Location Register*

IBGE – Instituto Brasileiro de Geografia e Estatística

IEEE – Instituto de Engenheiros, Eletricistas e Eletrônicos

IFAM – Instituto Federal do Amazonas

IHA – Índice de Homicídios na Adolescência

LBS – *Location Based Services*

OMT – *Object Modeling Technique*

OOSE – *Object-Oriented Software Engineering*

SDK – *Software Development Kit*

SGL - *Skia Graphics Library*

SIM – *Subscriber Identity Module*

SMS – *Short Message Service*

TA – *Timing Advance*

WAAS – *Wide Area Augmentation System*

Wi-Fi – *Wireless Fidelity*

WLAN – *Wireless Local Area Network*

SUMÁRIO

1. CONTEXTUALIZAÇÃO E PROBLEMÁTICA

Atualmente o contingente jovem, na faixa de 15 a 19 anos, representa cerca de 7,65% da população brasileira, conforme os dados apresentados em outubro de 2019 pelo IBGE, o que representa uma estatística muito expressiva. Contudo, esse é o grupo etário que mais sofre com os índices de mortalidade por violência. É possível observar em todos os tipos de mídias, notícias que retratam diversos tipos de crimes que acometem os jovens.

E esse é o motivo que faz com que pais e familiares fiquem mais aflitos e preocupados e com sensação de impotência diante da mídia negativa sobre esse assunto. Os alunos adolescentes e jovens estão sujeitos a sofrer qualquer tipo de trauma violento desde quando saem de suas casas até quando chegam no ambiente escolar, aponta um artigo publicado no portal A mente é maravilhosa em 2018.

Em uma pesquisa realizada pelo Instituto de Pesquisa Econômica Aplicada e pelo Fórum Brasileiro de Segurança Pública, o número de homicídios de crianças e jovens brasileiros tem crescido desde a década de 1980. Entre 2016 e 2017, o número de homicídios passou para 35,7 mil casos por ano (ATLAS DA VIOLÊNCIA, 2019). Isso significa que a cada 100 mil jovens brasileiros, quase 70 são assassinados, o que representa um aumento de 37,5% na última década. Outro estudo, o índice de Homicídios na Adolescência (IHA) 2014, aponta que, na região norte, 3,29 adolescentes em cada 1.000 são vítimas da violência letal antes de completarem 19 anos de idade e que aconteceriam mais de 4 mil mortes de pessoas que possuem entre 12 e 18 anos de idade entre os anos de 2015 e 2021. Assim, esse grupo etário está se tornando alvo preferencial dos homicídios quando comparado a outros grupos de idade. Outras causas também ceifam a vida dos jovens, tais como acidentes de trânsito e suicídios. Em 2013, 5,262 jovens morreram vítimas de acidentes de trânsito e outros 788 cometeram suicídios, o que representou um aumento de 10% e 63,5%, respectivamente (WAISELFISZ, 2015).

Além do problema exposto, é possível identificar outro contratempo. Serra (2010) afirma que o número de alunos que "matam aula" vem crescendo continuamente e seus pais nem sabem deste fato. Esse termo "matar aula" está sendo utilizando para caracterizar as ausências dos alunos nas escolas públicas e particulares, cursinhos preparatórios e instituições de ensino superior. O evento

também pode ser considerado quando o aluno está na escola, mas não está dentro da sala durante o horário de aula. Muitos problemas podem estar relacionados a essa evasão como desinteresse dos alunos, falta de estrutura escolar, baixa qualificação dos professores, condições geográficas e socioeconômicas, base familiar inadequada, entre outros (PEZZINI & SZYMANSKI, 2008).

Considerando essa problemática, os pais são os que mais se preocupam com a segurança dos filhos, dentro e fora das escolas, procurando sempre dialogar e alertar seus filhos sobre esse problema social (SILVA, 2014). De acordo com uma notícia publicada no site da revista Exame, além dos eventos relacionados à violência, muitos adolescentes se ausentam desacompanhados da escola, sem o consentimento dos pais. Saboya (2017) afirma que é necessário implantar medidas que possam dar mais tranquilidade aos filhos e aos seus pais.

Logo, existe a necessidade das escolas, como o IFAM, que tem uma área territorial muito grande e com quantidade elevada de alunos adolescentes, terem controle sobre entradas e saídas dos discentes e até mesmo verificar se estes estão dentro das salas no horário das aulas, uma vez que os pais cobram isto da instituição.

Em função dessa situação e dos receios apresentados, surge o seguinte questionamento: Como seria possível monitorar entrada e saída de alunos dentro do IFAM CMC e verificar a posição deles dentro do instituto?

Na busca de solucionar esse problema, propõe-se neste trabalho o desenvolvimento de um aplicativo que poderá ser utilizado pelos pais e/ou instituição para monitorar o acesso e a posição atual dos alunos dentro da escola que tiverem com este aplicativo instalado no celular.

1.1. OBJETIVOS

Os seguintes objetivos foram estabelecidos:

1.1.1. Objetivo geral

Desenvolver um aplicativo para dispositivos móveis que permite que os pais possam receber notificações de quando os seus filhos chegam e saem da escola, utilizando os recursos de geolocalização disponíveis nesses dispositivos apoiado com API de mapas e BaaS (*Backend as a Service*) para a utilização dos serviços de *backend*.

1.1.2. Objetivos específicos

- Analisar as tecnologias de geolocalização utilizadas nos smartphones e as plataformas de serviços de *backend*.
- Identificar tecnologia para monitorar a área do IFAM e que será utilizada pelo recurso de alertas para os responsáveis de que seus filhos chegaram/saíram dessa área rastreada;
- Desenvolver aplicação mobile que permita o envio de notificações de entrada e saída e visualização em tempo real da localização do filho dentro da instituição de ensino através do dispositivo móvel.

1.2. JUSTIFICATIVA

Considerando que os responsáveis demonstram certa aflição relacionada à educação dos seus filhos, que depende da presença deles nas salas de aula, e à segurança, que é definida pela dúvida se os alunos já chegaram à escola (SILVA & SILVA, 2014), as instituições de ensino estão investindo em certos tipos de sistemas de controle de acesso e frequência dos alunos, como uso de carteirinhas e catracas biométricas, que oferecem uma segurança maior para os filhos e seus pais, além de ajudar na gestão da própria instituição, afirma o site Pacto Soluções que desenvolve e implementa esses tipos de sistemas (PACTO SOLUÇÕES, 2017).

O uso de catracas eletrônicas pode ser uma solução. Elas registram o horário de entrada e saída dos alunos que por ela passam através de cartões magnéticos ou digitais (TOPDATA, 2016). Entretanto, essa tecnologia ainda não pode confirmar a presença do aluno na sala. Além disso, as catracas são caras, precisam de manutenção, software de gerenciamento e pessoal treinado para operá-la. Observando a realidade financeira das escolas públicas, esse investimento acaba sendo alto demais, afirma Barbosa (2019).

O desenvolvimento deste projeto pode suprir a necessidade de o os responsáveis quererem saber se o filho já chegou ou saiu da escola e onde ele realmente está, independente da instituição de ensino, uma vez que, usando o aplicativo, o responsável poderá visualizar a posição atual do aluno dentro da instituição.

1.3. METODOLOGIA

De acordo com os objetivos estabelecidos, a execução deles será viabilizada através do cumprimento da seguinte metodologia:

- Levantamento bibliográfico que inclui pesquisas bibliográficas dos assuntos referentes ao tema, bem como em artigos, pesquisas, e trabalhos relacionados;
- Estudo dos trabalhos correlatos que inclui análises preliminares em aplicativos similares com o intuito de entender o funcionamento das técnicas que poderão ser utilizadas no referido trabalho;
- Análise do problema e do contexto que constitui na definição dos requisitos a serem alcançados pelo aplicativo, análise geográfica do IFAM CMC, além de analisar e definir quais as tecnologias que possibilitem a solução proposta;
- Implementação utilizando os métodos e tecnologias selecionadas para o desenvolvimento do aplicativo.
- Testes que serão realizados para assegurar que os requisitos sugeridos ao aplicativo foram alcançados e a realização de eventuais alterações consideradas necessárias e atingir maior eficácia do projeto.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta um levantamento das tecnologias, trabalhos e pesquisas utilizadas no desenvolvimento deste projeto.

2.1. SERVIÇOS BASEADOS EM LOCALIZAÇÃO

Os serviços baseados em localização ou *Location Based Services* (LBS) são serviços ou sistemas que utilizam informações geográficas, que podem ser combinadas ou não com a posição atual do dispositivo móvel, para obter e gerar informações úteis para os usuários (SILVA *et.al.,* 2004). Com isso, o dispositivo do usuário pode determinar sua localização através de informações espacialmente relacionadas além de disponibilizar uma interação dinâmica com algum provedor de conteúdo (FERRARO & AKTIHANOGLU, 2011).

Outra definição relevante que a *GSM Association*[1] utiliza é que um LBS pode usar a localização de um "alvo" para agregar valor a este serviço, em que o "alvo" é a entidade a ser localizada e essa entidade não é necessariamente também o usuário do serviço. É possível citar alguns exemplos de valores agregados: serviços que ajudam a visualizar pontos próximos de interesse, que mostram a localização de um lugar específico em um mapa ou são ativados automaticamente quando a entidade entra ou sai de um local pré-definido (KUPPER, 2005).

Os sistemas que utilizam LBS são geralmente baseados em uma arquitetura cliente-servidor (ALVES *et. al.,* 2012), que integra quatro elementos chaves: o dispositivo móvel, a aplicação servidora ou provedor de conteúdo, infraestrutura da rede de comunicação e a tecnologia de localização (EGGEA, 2013), conforme apresentada na Figura 1.

[1] GSM Association: também referenciada como GSMA ou *Global System for Mobile Communications Association* A GSMA representa um conglomerado de companhias e operadoras móveis em todo o mundo, incluindo fabricantes de aparelhos e dispositivos, empresas de software, fornecedores de equipamentos e empresas de Internet, além de organizações em setores industriais adjacentes.

Figura 1 – Elementos chaves de um serviço baseado em localização

Fonte: FERRARO & AKTIHANOGLU, 2011. Adaptada.

A seguir, serão descritos os elementos que compõe os sistemas baseados em localização.

2.1.1. Dispositivo Móvel

Quando se fala em dispositivos móveis, está se referindo aos dispositivos eletrônicos capazes de se conectarem a uma rede móvel, utilizando cartão de Módulo de identidade do assinante – ou *Subscriber Identity Module* (SIM) - e realiza transferência de voz e/ou dados (fazendo uma ligação ou download de arquivos, por exemplo). Esses dispositivos, tais como smartphones e tablets, tiveram uma grande importância para a popularização no uso de aplicações LBS, dada as suas configurações de hardware cada vez melhores e a inserção quase universal de tecnologias de geolocalização (como o *Global Positioning System* - GPS) (ALVES *et. al.,* 2012).

2.1.2. Provedor de Conteúdo

O provedor de conteúdo é a entidade que utiliza as informações geográficas ou cria o conteúdo baseado nessas informações e que é fornecida diretamente aos dispositivos móveis ou através de terceiros. Um provedor LBS não cria ou armazena

o conteúdo gerado e os dados que podem ser utilizados pelo usuário no dispositivo móvel. Um exemplo disso são os mapas, que podem ser fornecidos através do *Google Maps* em um dispositivo Android. Os dados são disponibilizados de forma separada, através destes provedores, que podem ser filtrados pelo usuário em um serviço. Por isso o papel destes fornecedores de conteúdo tem-se tornado cada vez mais relevante, visto que eles integram a maioria dos serviços LBS atualmente (EGGEA, 2013).

2.1.3. Infraestrutura de Comunicação

Trata-se da infraestrutura de comunicação sem fio que o desenvolvedor do aplicativo LBS pode utilizar para gerenciar o tráfego de informações compartilhadas entre o cliente e o servidor e que são utilizadas pelo aplicativo, maximizando o desempenho da transferência. Podem ser usadas diversas tecnologias desde a rede de telefonia celular até redes sem fio como a IEEE[2] 802.11 (FERRARO & AKTIHANOGLU, 2011).

2.1.4. Tecnologia de localização

A tecnologia de localização permite obter as informações espaciais do dispositivo além de poder repassar essas informações para a aplicação. Há certo grau de escolha entre as tecnologias a serem utilizadas, que podem ser, por exemplo, *Cell ID*, navegação por satélite (GPS) ou por *Wi-Fi*. Para os dispositivos que possuem mais de uma tecnologia de localização integrada, é possível a utilização híbrida de posicionamento que vem se tornando cada vez mais utilizada, justamente para minimizar as desvantagens de uma ou outra tecnologia (EGGEA, 2013).

O conceito de serviços baseado em localização e os recursos utilizados por esses serviços, tais como GPS, *Cell ID* e *Wi-Fi* constituem a base tecnológica do projeto que será desenvolvido, pois a partir desses recursos é possível determinar

[2] IEEE: o Instituto de Engenheiros Eletricistas e Eletrônicos é uma organização sem fins lucrativos que fomenta os conhecimentos na área da engenharia elétrica, eletrônica e da computação. Algumas de suas funções mais relevantes são a criação, aprovação e divulgação de normas técnicas dentro dessas áreas e definições de padrões que são utilizados em dispositivos eletrônicos e computadores.

automaticamente a localização do smartphone e combinar com mapas e áreas georreferenciadas. Esses recursos serão abordados à frente (FERRARO & AKTIHANOGLU, 2011).

2.1.4.1 Cell ID

Este sistema utiliza a localização das células de estações rádio bases (ERBs) e a potência do sinal para medir a distância do dispositivo até a fonte de origem do sinal. É utilizada, por exemplo, em redes GSM que utilizam a ERB para identificar a posição dos dispositivos dentro de sua área de cobertura (MONTEIRO, 2005).

Essa tecnologia apresenta uma resposta rápida, pois não é necessário realizar cálculos complexos para localizar o dispositivo. Além disso, quando o usuário realiza uma chamada, as informações de localização do dispositivo são atualizadas em tempo real. Entretanto, se dispositivo se encontrar inativo, a última localização registrada é armazenada pela rede em um sistema chamado *Home Location Register* (HLR). Para atualizar a localização do aparelho, a rede pode solicitar a monitorização da qualidade do sinal recebido pelas diversas estações adjacentes, para então poder identificar a célula da rede mais próxima ao dispositivo e solicitar a localização atualizada (JOHNSON, 2007).

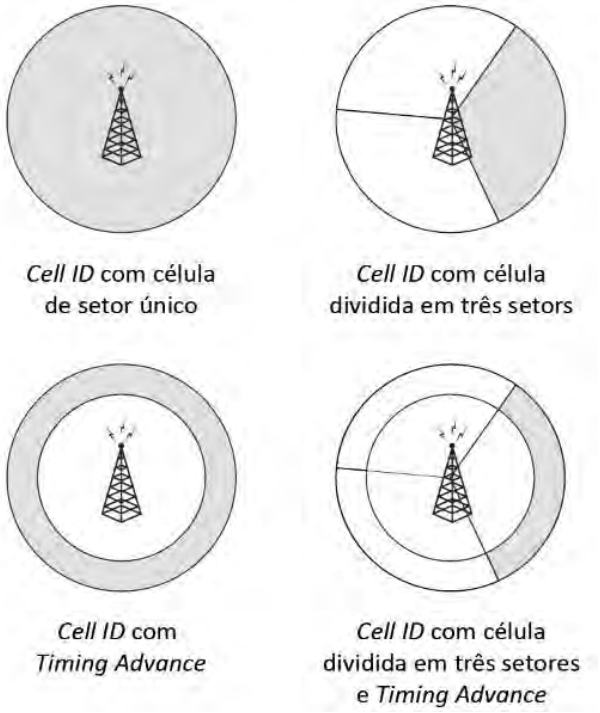
A característica negativa dessa tecnologia é que as estações rádio bases possuem áreas de cobertura de tamanhos distintos e que podem se sobrepor e por isso não é possível determinar a posição exata de um objeto. Portanto, a precisão da localização depende do tamanho da área de cobertura da estação, que pode variar de 150 m até 20 km, que quando usada em aplicações que não precisam de alta precisão, pode ser aceitável. Porém para serviços de emergência ou de rastreamento, tal abordagem não fornece informações satisfatórias (MONTEIRO, 2005).

Existem algumas técnicas que podem ser usadas para melhorar a precisão da *Cell ID.* A área de cobertura de algumas células é dividida em seções, reduzindo a área total da possível localização do objeto. Para obter uma leitura ainda mais precisa, a técnica chamada *Timing Advance* (TA) pode ser usada para calcular a distância que o dispositivo está da célula ERB através de informações detalhadas obtidas em um Centro de Posicionamento Móvel (CPM). Através da combinação destes métodos, que

é referenciada como *Cell Global Identity-Timing Advance* (CGI-TA), é possível produzir resultados com precisão dentro de 100 a 200 metros, sendo que esta precisão é melhor das cidades do que nas áreas rurais devido a maior concentração de estações base nas áreas urbanas (MALLICK, 2003).

A figura 2 mostra a precisão de várias técnicas de posicionamento que usam *Cell ID* e *TA*; as áreas sombreadas representam as possíveis localizações do dispositivo móvel.

Figura 2 - Áreas de posicionamento *Cell ID* e *Timing Advance*



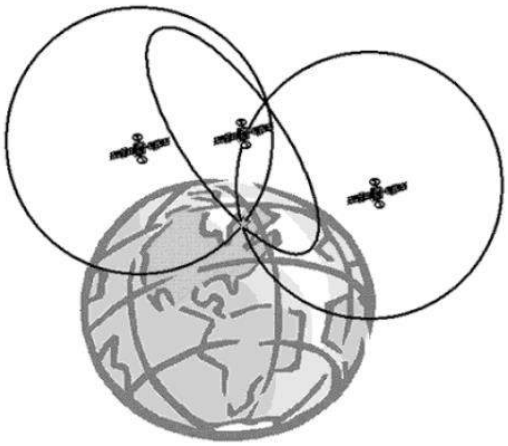
Fonte: MALLICK, 2003.

2.1.4.2 GPS (Global Position System)

O sistema de posicionamento global baseado em satélites, que foi desenvolvido pelo departamento de Defesa dos Estados Unidos em 1973, apresenta alto grau de precisão e funciona praticamente em qualquer parte da superfície terrestre (CORREIA, 2004). Para que o sistema possa estimar a localização de um dispositivo, é necessário utilizar sinais enviados por alguns satélites. Por consequência, este sistema não funciona corretamente dentro de ambientes fechados como prédios ou casas ou mesmo em lugares cercados de construções muito altas (MOURA, 2007).

O GPS funciona através do receptor do dispositivo do usuário que recebe o sinal de pelo menos três satélites realizando a triangulação e calcula o tempo de viagem do sinal de rádio. Através desse tempo e da posição do satélite no espaço, é possível calcular a área do arco na superfície da Terra de cada satélite. A partir disso, é possível gerar interseções nos arcos que aumentam a precisão de localização do objeto. Além de conseguir realizar cálculos para posicionamento, esse método pode ser usado para calcular tempo, velocidade e altitude (HJELM, 2003). É possível visualizar a triangulação dos satélites na Figura 3.

Figura 3 - Triangulação dos satélites



Fonte: HJELM, 2003

É uma tecnologia que oferece alta precisão com erro de estimativa máximo de 10 metros em áreas abertas. Ainda é possível reduzir os problemas de posicionamento e aumentar o grau de precisão para ordem de centímetros através de sistemas que utilizam estações de referência em terra com coordenadas conhecidas. Alguns desses sistemas são o *Differential Global Positioning System* (DGPS) e o *Wide Area Augmentation System* (WAAS). O DGPS usa pontos de referência próximos ao usuário que transmitem as informações de correção diretamente aos dispositivos. Já o WAAS, transmissão os dados através de satélites geoestacionários (CORREIA, 2004).

2.1.4.3 Wi-Fi

Um dos primeiros sistemas de posicionamento baseado em tecnologia de radiofrequência foi o Ethernet sem fio, usualmente chamado de Wi-Fi, que usada para construir redes locais *wireless* (WLAN – *Wireless Local Area Networks*). Esta tecnologia é utilizada em larga escala atualmente e está presente em diversas escolas e campi universitários, sendo uma opção muito atrativa para sistemas de localização (RODRIGUES, 2011).

Vários estudos sobre essa tecnologia geraram sistemas que podiam realizar a localização em ambientes internos, como por exemplo, o RADAR (BAHL & PADMANABHAN, 2000) que se baseia em medidas de força de sinal empírico e em um modelo de propagação de sinal. Embora o método de propagação de sinal seja inferior em termos de precisão, ele torna a implantação mais fácil e exige menos esforço manual. O método que utiliza força de sinais empírica consegue alcançar exatidão média de 2 a 3 metros, dentro de uma área do tamanho de uma sala de escritório comum.

3. TRABALHOS RELACIONADOS

Nesta seção, serão apresentados alguns trabalhos relacionados que propõem funcionalidades semelhantes ao do aplicativo que será desenvolvido.

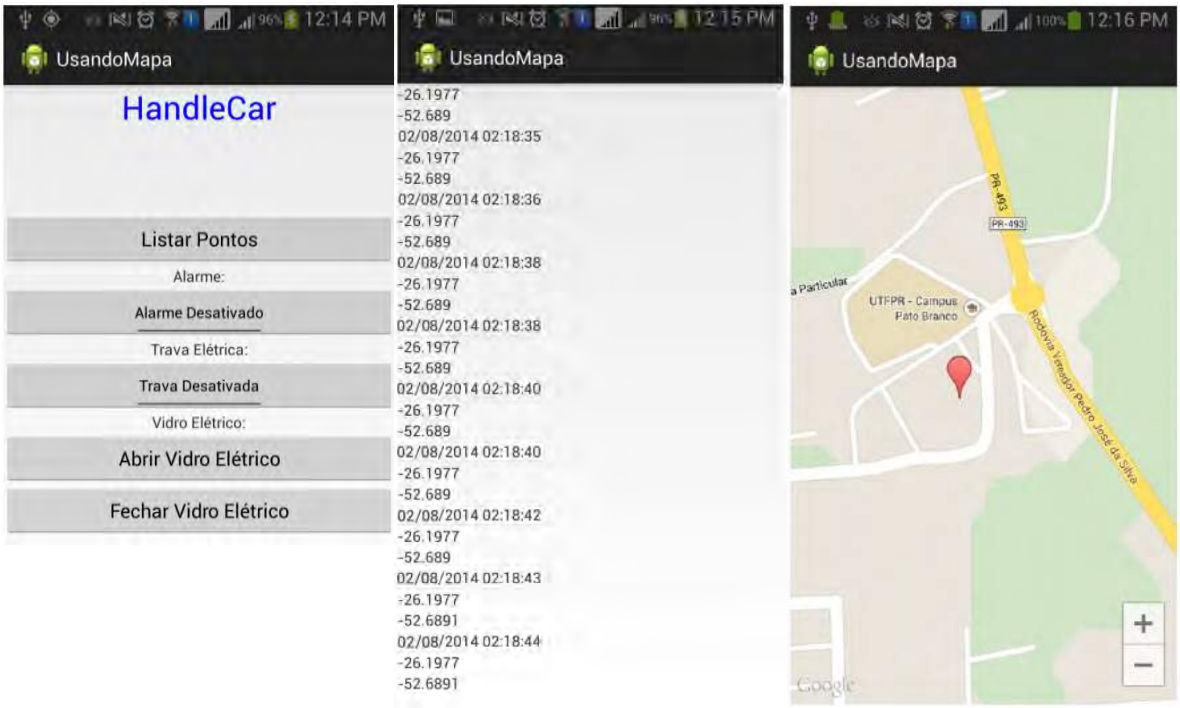
3.1. SISTEMA DE RASTREAMENTO E CONTROLE DE RECURSOS DE UM VEÍCULO UTILIZANDO UM SMARTPHONE ANDROID

Em seu trabalho de conclusão de curso, Galon (2014) desenvolveu um aplicativo para o sistema Android que permitia o controle dos recursos de um automóvel à distância, como alarme, travas elétricas e vidros elétricos das portas, assim como recuperação de dados deste, utilizando smartphone, um microcontrolador e a rede da operadora de telefonia. Além disso, era possível receber uma mensagem de texto (SMS) informando caso o alarme do automóvel fosse disparado, bem como verificar a posição atual do veículo e consultar o histórico através do GPS.

O funcionamento do sistema se dá através de um módulo GSM, um módulo GPS e um microcontrolador localizado no interior do veículo e que fornece os dados de posicionamento através de envios de SMS e recebe comandos para executar a ação desejada relacionada ao alarme, vidros elétricos e travas elétricas do veículo através do aplicativo instalado no smartphone do usuário.

É efetuada, constantemente, a coleta de dados referente ao posicionamento do veículo através do módulo GPS e a cada intervalo de tempo, o usuário recebe estas informações para que possa visualizar a posição atual do automóvel em tempo real. O sistema se torna inoperante caso não seja possível obter as coordenadas geográficas, como, por exemplo, se o veículo estiver dentro de túneis ou em garagens cobertas e/ou se o smartphone estiver em um local onde não haja a cobertura da rede da operadora para receber as mensagens de texto.

Figura 4 – Interfaces do aplicativo: (a) Tela Inicial; (b) lista de posições; (c) Posição do veículo



Fonte: GALON, 2014.

Analisando esse trabalho e o projeto em questão é possível verificar algumas similaridades: é feita a coleta de dados referente ao posicionamento, no qual o primeiro sistema obtém essas informações em um intervalo de tempo estabelecido, além de poder visualizar a posição atual do dispositivo móvel utilizando o Google Maps;

3.2. COMPASS

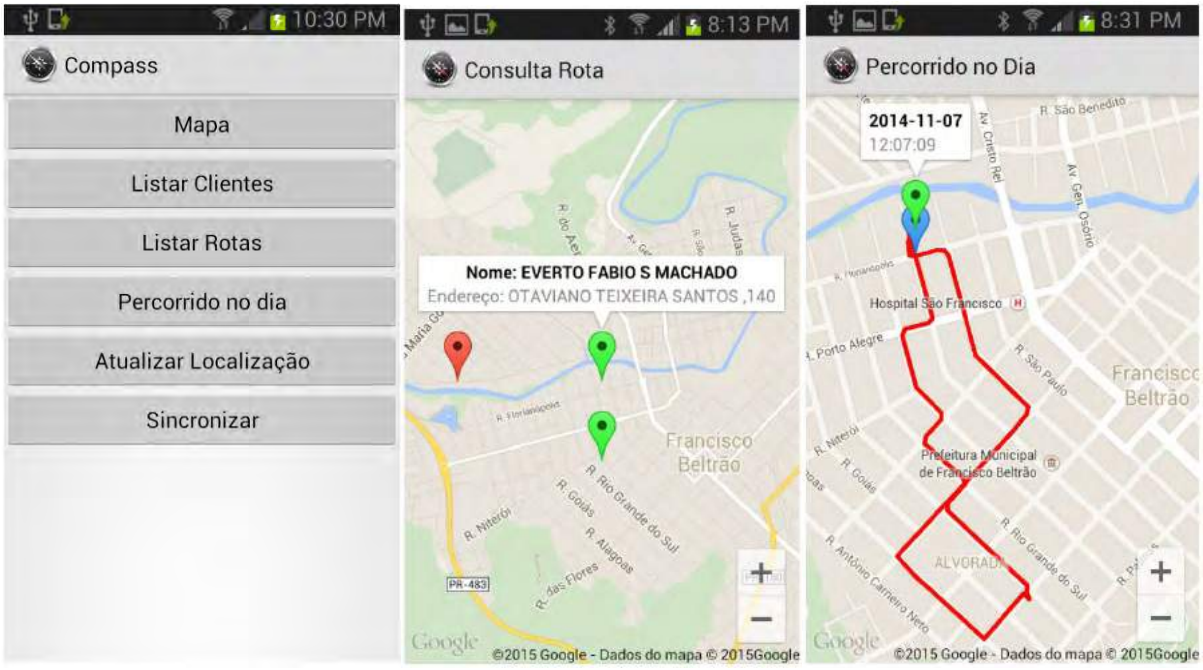
Machado (2015) apresentou um projeto que implementava o mapeamento de todos os clientes de uma empresa de logística, a fim de construir um banco de dados capaz de realizar o controle das visitas dos representantes comerciais e traçar rotas a serem cumpridas de maneira mais eficaz e eficiente. Através de um aplicativo desenvolvido para a plataforma Android, integrado ao servidor socket, ao ERP[3] (*Enterprise Resource Planning*) e ao banco de dados, foi possível fornecer dados dos clientes e rotas a serem seguidas pelos representantes de vendas.

[3] ERP: é um sistema que permite gerenciar os dados de uma empresa de forma integrada e confiável, e que é usado para auxiliar nas tomadas de decisões para melhorar o desempenho e a eficiência da companhia.

Com o aplicativo, o usuário registra as informações relativas ao endereço de cada cliente que as envia ao servidor socket, onde serão inseridas no banco de dados do ERP. O servidor socket compartilha os dados com todos os dispositivos móveis integrados ao sistema. Quando precisar fazer a rota, o representante usa o app para carregar as informações dos clientes que ele deverá visitar e traça as rotas, exibindo-as através do Google Maps. Após cumprir o percurso, o usuário registra essa informação através do aplicativo, que atualiza dos dados no servidor com os novos dados relativos à localização do dispositivo.

A localização do dispositivo é atualizada sempre que haver movimentação de 20 metros ou mais e essa é gravada no banco de dados do celular, para que posteriormente seja possível sincronizar essas informações com o banco de dados do servidor e poder identificar os pontos já visitados e construir o percurso realizado e para as demais rotinas que necessitarem desta informação.

Figura 5 – Interfaces do aplicativo: (a) Tela Inicial; (b) consulta de rota; (c) Caminho percorrido



(a) (b) (c)

Fonte: MACHADO, 2015.

O COMPASS apresenta algumas funcionalidades semelhantes com o aplicativo proposto, tais como: registro da posição do dispositivo a cada movimentação de pelo menos 20 metros a fim de criar um histórico da rota realizada,

tornado possível a visualização desse caminho utilizando o Google Maps, além de poder consultar a posição atual e os próximos clientes a serem visitados.

3.3. MARISTA VIRTUAL

O Marista Virtual é um aplicativo gratuito, desenvolvido pela iCentury, que pode ser usado nos colégios da Rede Marista presentes em Rio Grande do Sul e Brasília. O aplicativo oferece diversas funcionalidades, dentre elas o controle da entrada e saída dos alunos que notifica os pais quando os alunos chegam ou saem da escola. Um aviso é enviado ao usuário do aplicativo assim que o aluno passa pela catraca da portaria.

Figura 6 – App Marista Vistual

Fonte: APPS4BUSINESS

O ponto semelhante em relação ao aplicativo MARISTA VIRTUAL é o uso de notificações para avisar aos pais quando os alunos chegam ou saem da escola, funcionalidade essa considerada muito importante para o projeto. Entretanto, o projeto que será desenvolvido não utilizará a catraca para envio de mensagens e sim a área demarcada pelo *geofence* da API do *Google Fences*.

Analisando os trabalhos relacionados, é possível sintetizar na Tabela 1 as principais funcionalidades dos aplicativos desenvolvidos comparados ao aplicativo proposto neste trabalho, o MAPES.

Tabela 1 – Comparação de funcionalidades dos aplicativos relacionados e o MAPES

	SISTEMA DE RASTREAMENTO E CONTROLE DE RECURSOS DE UM VEÍCULO	COMPASS	MARISTA VIRTUAL	MAPES
Coleta de dados referente ao posicionamento	X	X		X
Visualização em tempo real utilizando API do Google Maps	X	X		X
Criação de rotas		X		X
Notificações de entrada e saída			X	X

Fonte: próprio autor

4. PROCESSO E TECNOLOGIAS A SEREM UTILIZADOS NO DESENVOLVIMENTO DA APLICAÇÃO MOBILE

4.1. PLATAFORMA ANDROID

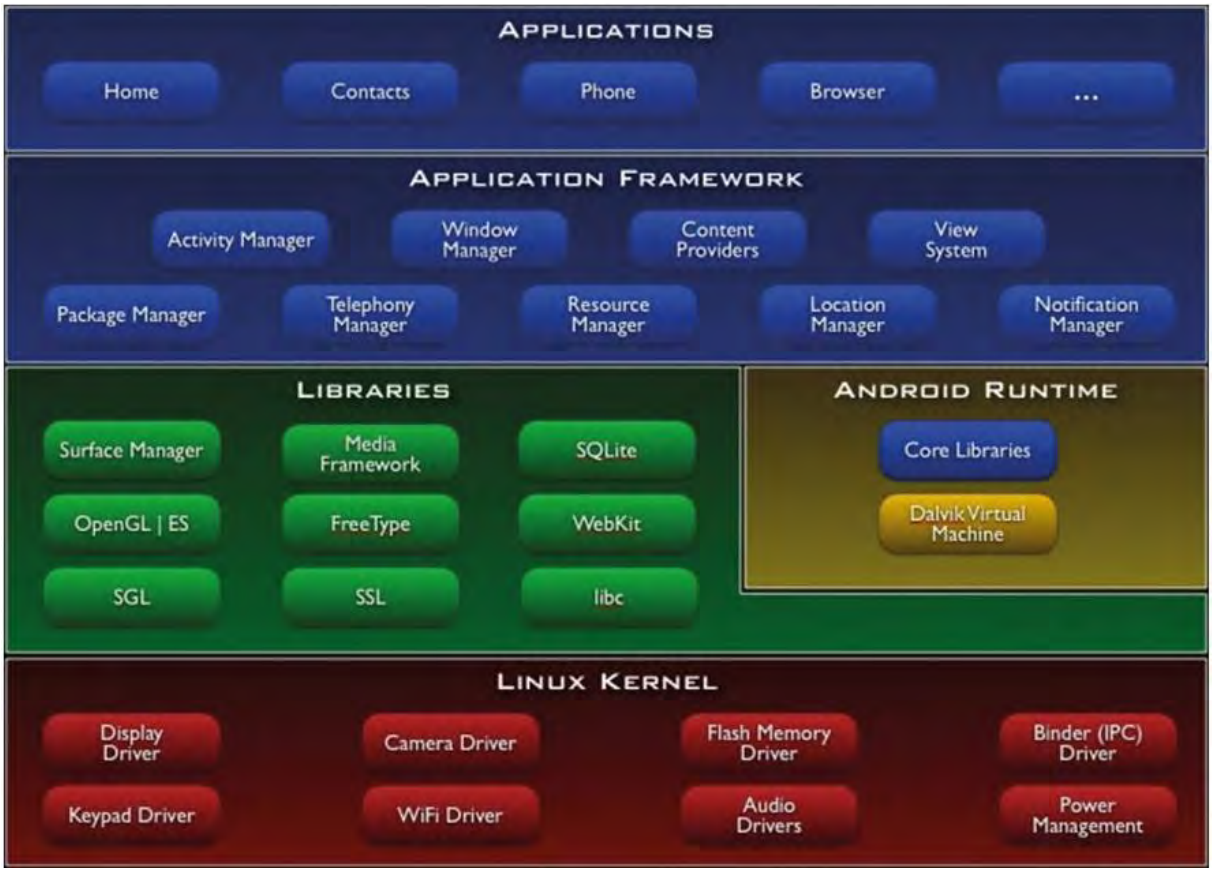
Android é a primeira plataforma de aplicativos para dispositivos que possui o código-fonte livre e que está revolucionando o mercado mundial através de seu uso em *smartphones*, tablets e em outros aparelhos. Criado pelo Google, juntamente com a *Open Handset Alliance*, que é uma aliança de dezenas de organizações que estão com intuito de fazer um dispositivo “melhor” e mais “aberto” (ABLESON, KING & SEN, 2012).

4.1.1. Arquitetura Android

O Android é baseado no sistema operacional Linux, que é responsável por gerenciar a memória, os processos e threads em execução, segurança dos arquivos e pastas, além dos componentes de redes e drivers (LECHETA, 2016). Alguns recursos encontrados na plataforma são: framework de aplicação que permite reuso de componentes, Máquina Virtual Dalvik otimizada para dispositivos móveis; gráficos otimizados 2D e 3D; banco de dados relacional nativo (SQLite); suporte a tecnologias GSM, EDGE, 3G, 4G e Wi-Fi (dependente de hardware); bluetooth, câmera, GPS, bússola e acelerômetro (dependente de hardware) (SILVA, 2015).

A arquitetura da plataforma é dividida em várias camadas: *Applications*, *Application Framework*, *Libraries, Android Runtime* e *Linux Kernel* (RABELLO, 2013). A Figura 7 apresenta cada uma dessas camadas.

Figura 7 – Arquitetura geral da plataforma



Fonte: SILVA, 2015.

Na camada *Applications*, está localizado um conjunto de aplicações nativas como gerenciador de contatos, cliente de e-mail, navegador, aplicativo de SMS, mapas, calendário, entre outros, além das funções mais básicas como fazer chamadas telefônicas, todos desenvolvidos na linguagem de programação Java. É nesta camada que o usuário interage com as interfaces dos aplicativos (MACK, 2010).

A camada *Application Framework* apresenta os componentes que permitem a reutilização de código permitindo que esses componentes sejam utilizados para futuras aplicações. Alguns exemplos desses componentes são os componentes gráficos, provedores de conteúdo, gerenciador de recursos, gerenciador de notificação e gerenciador de *activities* (RABELLO, 2013).

A camada seguinte é subdivida no grupo das bibliotecas (*Libraries*) e ambiente de execução (*runtime*) que é composta pelas bibliotecas padrão escritas em C/C++, dentre elas: biblioteca de sistema C, bibliotecas de mídias (áudio, vídeo e imagens),

gerenciador de superfície, LibWebCore, SGL, bibliotecas 3D, FreeType e SQLite; e pela máquina virtual Dalvik onde toda aplicação Android é executada (SILVA, 2015).

A última camada, *Linux Kernel*, fornece serviços do núcleo do sistema, como gerenciamento de memória, gerenciamento de processos, segurança, pilhas de redes, drivers dos componentes etc. (EGGEA, 2013).

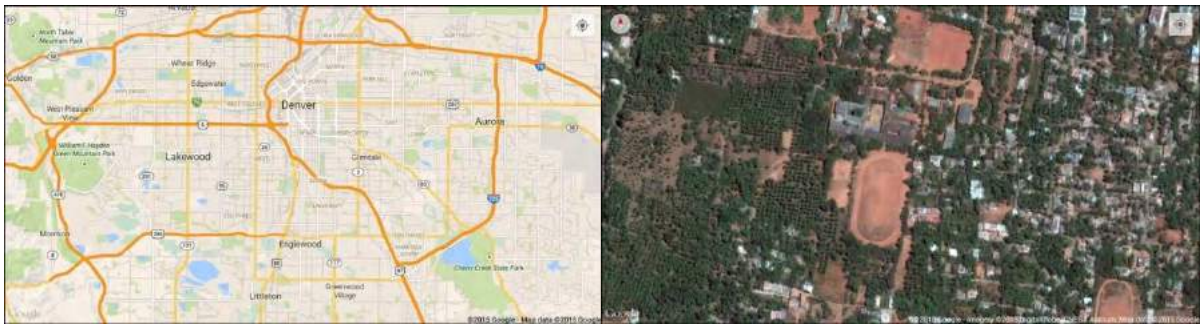
4.2. GOOGLE MAPS API

Um dos serviços mais conhecidos do Google é o de mapas, conhecido como *Google Maps*, que permite, de forma gratuita, a visualização de mapas e imagens de satélite, além de pesquisa de lugares, contatos, rotas e outros recursos (OLIVEIRA, 2012). Além de poder visualizar a localização do aparelho, é possível obter e prover informações relevantes e pertinentes ao local em que o dispositivo se encontra (LEAL, 2016).

O serviço, que foi anunciado em 2005, apresenta e atualiza de forma assíncrona, mapas interativos com carregamento rápido, exibindo interface simples e amigável para os usuários, e ainda se transformou em uma API pública rica em recursos que pode ser facilmente integrada aos sites e aplicativos, permitindo personalização e customização dos mapas (GIBSON & ERLE, 2006).

O Google Maps evoluiu ao longo dos anos e adicionou novas funcionalidades, como rotas por vários tipos de transportes, outros tipos de mapas (Figura 8), visualização 3D de ruas e edificações (*street view*), bem como informações sobre o trânsito, tempos de viagens e transporte público (SCHIMITT, 2013).

Figura 8 – Tipos de mapas: (a) normal, (b) satélite, (c) híbrido e (d) relevo



(a) (b)

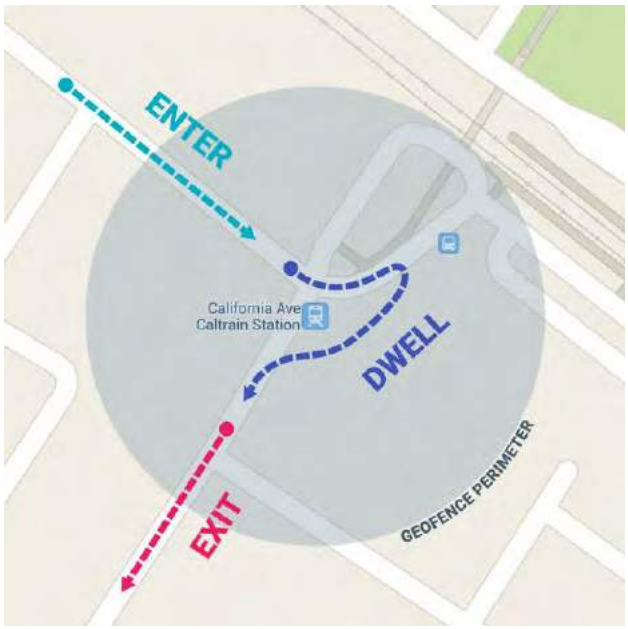
(c) (d)

Fonte: AMAL W., 2015.

4.2.1. Google Fences API

O conceito de *"fences"* é retirado de um termo conhecido como *"geofencing"*, que é definido com uma área virtual demarcada sobre uma região geográfica através de coordenadas de GPS (Figura 9). Geralmente utilizadas em aplicativos para dispositivos móveis com intuito de monitorar perímetros e notificar quando usuários entram ou saem dessa área (VINHAL, 2015).

Figura 9 – *Geofencing* de uma área demarcada

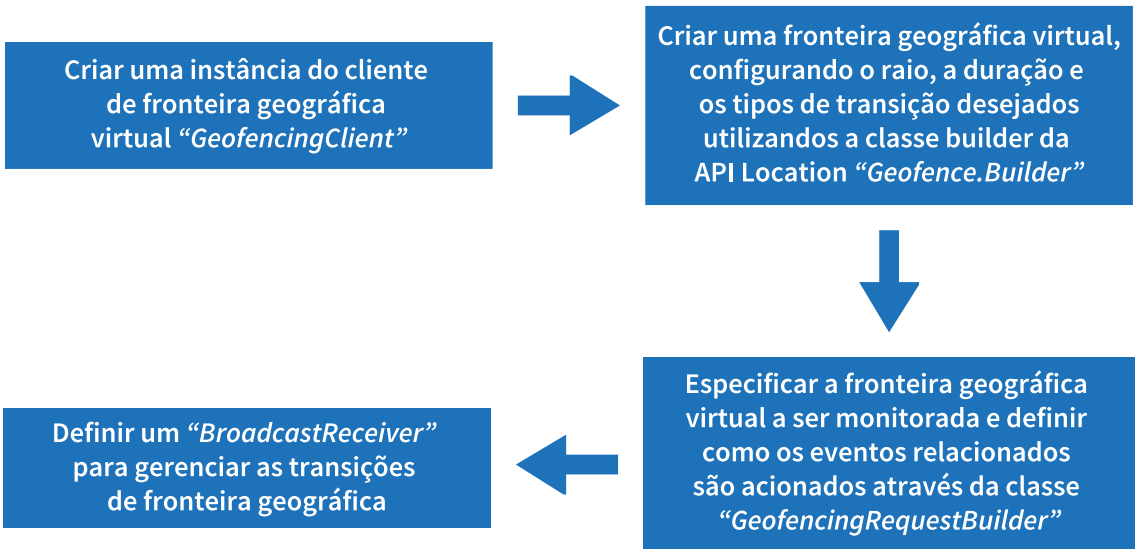


Fonte: Android Developers

Usando a *Fences* API do Google, é possível definir áreas (também conhecidas como *geofences*) que podem obter sinais de contexto, como por exemplo, a localização atual do dispositivo (latitude e longitude), ou alertas quando o usuário sai de uma *geofence* antes de 12h. Essa área é criada especificando a latitude, a

longitude e o raio e que definem a fronteira geográfica virtual no local de interesse. É possível ter até 100 áreas georreferenciadas por usuário do dispositivo e para cada uma delas, é possível utilizar os serviços de localização para tratar os eventos de entrada e saída, além de limitar a duração que essa área ficará ativa (GOOGLE, 2019). Esta API será de grande valia para o aplicativo, visto que a área do IFAM-CMC será demarcada e todas as vezes que o aluno entrar ou sair desta área, o aplicativo enviará uma notificação ao responsável. A figura 10 apresenta as etapas para criar e monitorar uma fronteira virtual.

Figura 10 – Etapas para criar uma fronteira virtual em Android/Java



Fonte: próprio autor

As principais classes dessa API são *GeofencingClient*, *Geofence.Builder* e *GeofencingRequest.Builder.* A classe *GeofencingClient* é o principal ponto de entrada para interagir com a *Fences* API e utiliza o método *addGeofences()* para adicionar uma área virtual que deve ser monitorada e um *BroadcastReceiver* para gerenciar a transição nessa área. Para criar as áreas geográficas virtuais utiliza a classe *builder Geofence.Builder* para criar os objetos de fronteira virtual, configurando o raio, a duração e os tipos de transição desejados para ela. Os tipos de transição definem os acionadores que serão utilizados no monitoramento, podendo ser quando o dispositivo entra e ou sai de uma área rastreada. A classe *builder GeofencingRequest.Builder* é utilizada para criar um objeto da classe *GeofecingRequest* que especifica as fronteiras geográficas virtuais a serem monitoradas e como as notificações de geofence devem ser relatadas (ANDROID DEVELOPERS, 2019).

4.2.2. Google Indoor Maps

Um dos recursos mais interessante habilitado no Google Maps disponibilizado no Brasil durante a Copa do Mundo de 2014, é o *Indoor Maps*, que permite que os usuários consigam visualizar a planta baixa de alguns lugares facilitando a busca por locais específicos e descobrindo novos pontos de interesse.

O recurso é habilitado quando o usuário aplica o zoom em um local no mapa e, se o serviço estiver disponível para aquele lugar, será possível navegar de andar em andar através dos mapas internos. A *feature* aumenta ainda mais precisão da localização do dispositivo, permitindo que o usuário visualize no mapa as lojas de um shopping, por exemplo, usando símbolos que são facilmente reconhecíveis e que representam diferentes pontos de interesse no interior dos locais (GOOGLE), conforme a Figura 11.

Figura 11 – *Indoor Map* do Shopping JK Iguatemi

Fonte: GOOGLE

Atualmente, os mapas internos estão disponíveis em alguns locais específicos. No Brasil, apenas alguns aeroportos, shoppings, estádios e teatros disponibilizam este recurso. Em Manaus, o serviço está disponível no Aeroporto Internacional Eduardo

Gomes, Manauara Shopping, Amazonas Shopping, Shopping Ponta Negra e Arena da Amazônia.

4.2.2.1 Upload de planta baixa no Google Indoor Maps

Para que o aplicativo desenvolvido possa ter suas funcionalidades implementadas adequadamente, é necessário carregar a planta baixa do local pelo computador. Neste projeto, foi carregada a planta do IFAM CMC.

Segundo a Google, alguns requisitos devem ser obedecidos antes de realizar upload da planta, tais como:

- Ter permissão ou licença para criar ou fazer upload do seu conteúdo, incluindo todas as informações relevantes a esse arquivo.
- Obedecer às leis vigentes, incluindo instruções e restrições do proprietário em relação ao upload da planta baixa.
- Respeitar a privacidade das pessoas com relação ao conteúdo que está sendo enviado.
- Não é permitido fazer upload de plantas baixas de construções não públicas, conteúdo de defesa nacional, conteúdo secreto, imagem de pessoas, conteúdo de marca registrada, conteúdo com direitos autorais e conteúdo ilegal ou impróprio.

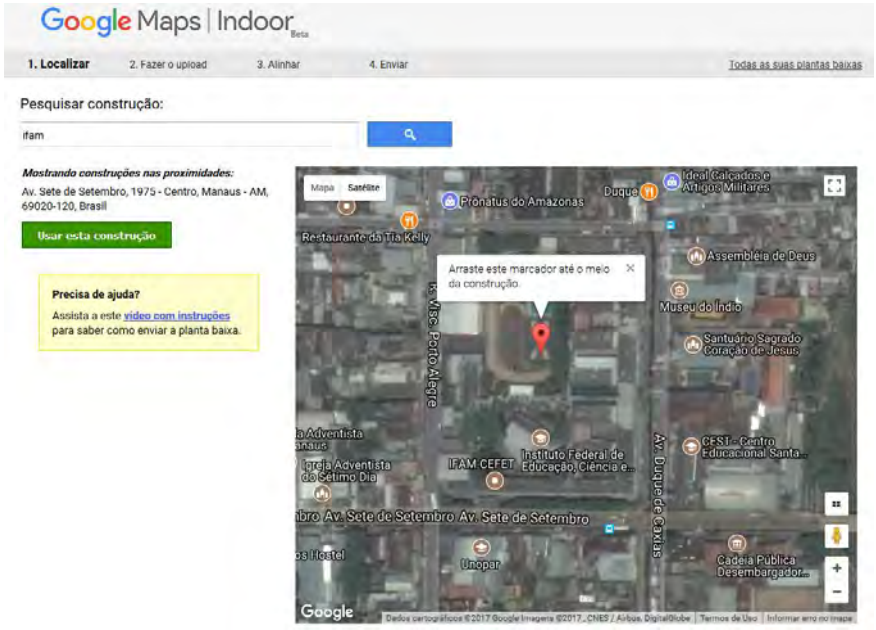
Os tipos de arquivos devem estar salvos como .JPG, .PNG, .PDF, .BMP ou .GIF, antes de carregar a planta baixa.

1. Acessar o navegador da Web com o link http://maps.google.com/floorplans/find.

2. Pesquisar a construção ou prédio.

3. Posicionar o marcador até o centro do prédio e selecionar **Usar esta construção.**

Figura 12 – Localizar prédio ou construção



Fonte: GOOGLE MAPS INDOOR

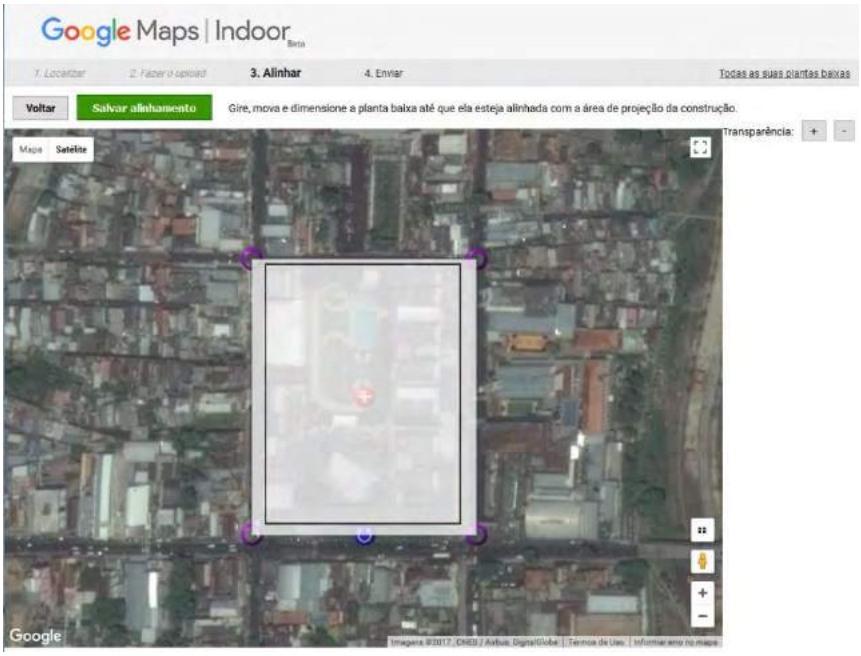
4. Inserir as informações relevantes, selecionar o arquivo da planta baixa e clicar em **Fazer o upload do mapa para este andar.**

Figura 13 – Fazer o upload

Fonte: GOOGLE MAPS INDOOR

5. É possível girar, mover e redimensionar a planta para que possa alinhá-la com as imagens de satélite. Após isso, selecionar **Salvar alinhamento.**

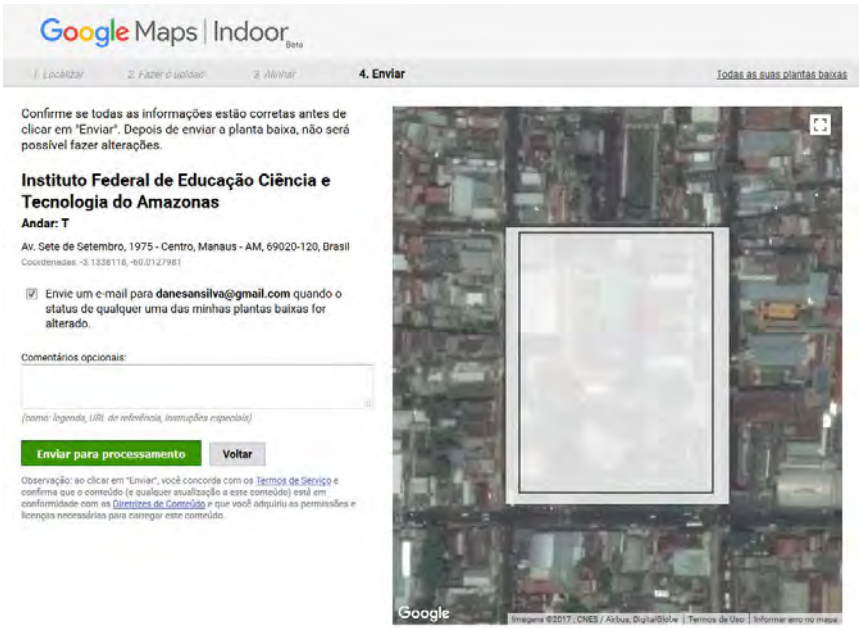
Figura 14 – Alinhar a planta



Fonte: GOOGLE MAPS INDOOR

6. Em sua última etapa, conferir as informações se estão corretas e inserir, opcionalmente, algum comentário. Para finalizar, clicar em **Enviar para processamento.**

Figura 15 – Enviar para processamento



Fonte: GOOGLE MAPS INDOOR

Após enviar o arquivo, ele é revisado, para então ser aprovado e ficar disponível no Google Maps.

4.3.FIREBASE

A computação em nuvem é um modelo utilizado que permite o acesso universal, vantajoso e sob demanda de recursos de redes compartilhados (servidores, armazenamento, aplicações e serviços) que podem ser fornecidos e disponibilizados com menor esforço de gerenciamento ou interação do provedor desses serviços (MELL; GRANCE et al., 2011). Assim, vários recursos de computação em nuvem podem ser adquiridos pelos consumidores através de serviços de forma elástica, sob demanda e a um baixo custo, que são alugados e nenhum outro *hardware* ou *software* é comprado diretamente pelo consumidor (WILLIAMS, 2010).

Esses serviços podem ser enquadrados em uma ou mais das seguintes modalidades, segundo Costa (2015):

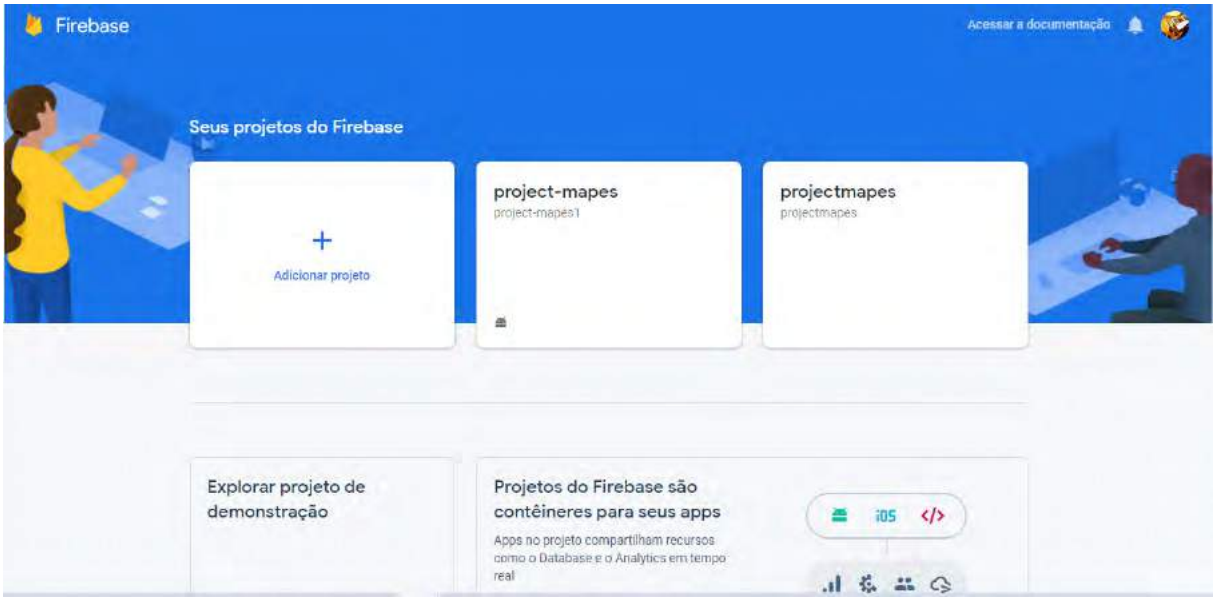
- *Software as a Service* (SaaS): fornece um conjunto de softwares já hospedados, dispensando a necessidade de desenvolver ou programar, apenas configurá-los conforme a necessidade.
- *Plataform as a Service* (PaaS): neste modelo é fornecido um ambiente para criar, hospedar e gerenciar o próprio software, onde o dimensionamento dos recursos é transparente e realiza uma abstração de máquinas virtuais e seus componentes.
- *Infrastructure as a Service* (IaaS): oferece o hardware (servidor, armazenamento e rede) associado a um software como um serviço. Logo, utiliza-se da virtualização dos recursos para dividir a infraestrutura de acordo com a capacidade dos recursos contratados. Trata-se, por exemplo, da mudança dos servidores físicos locais para os servidores em nuvem.

Os modelos descritos anteriormente são os principais e os mais conhecidos. Porém, existem outros modelos em ascensão, como por exemplo, o *Desktop as a Service* (DaaS), *Data Storage as a Service* (DSaaS), e o *Backend as a Service* (BaaS) (COSTA, 2015), que é o serviço utilizado nesse projeto.

Backend as a Service é o modelo de serviço que dispõe de SDKs (*Software Development Kit*) e APIs (*Application Programming Interface*) para que desenvolvedores possam conectar seus aplicativos aos serviços em nuvem. Ele fornece o *backend* com os recursos já configurados como armazenamento, processamento, integração com banco de dados, sistema de *push notification*[4], autenticação, entre outros, além de abstrair as complexidades de implantação e gerenciamento, fazendo com que os desenvolvedores foquem somente na construção das aplicações (PAIVA; LEAL; QUEIRÓS, 2016).

Sendo assim, o Firebase é um serviço do Google do tipo BaaS (*Backend as a Service*) baseado em computação em nuvem desenvolvido para aplicações mobile e web que foi lançado em 2004 e se tornou uma das principais ferramentas que auxiliam e aceleram o desenvolvimento dos projetos, sem a necessidade de gerenciar a infraestrutura (ORLANDI, 2018; FIREBASE, 2020). O gerenciamento dos projetos é realizado por meio do console do Firebase, o qual disponibiliza uma série de recursos para implantação e visualização dos mesmos, conforme as Figura 16 e 17.

Figura 16 – Tela principal dos projetos do Firebase



Fonte: FIREBASE CONSOLE (console.firebase.google.com)

[4]*Push Notification* é uma mensagem de dados enviada automaticamente pelo servidor para dispositivo do usuário sem que este faça uma solicitação explícita (LECHETA, 2016).

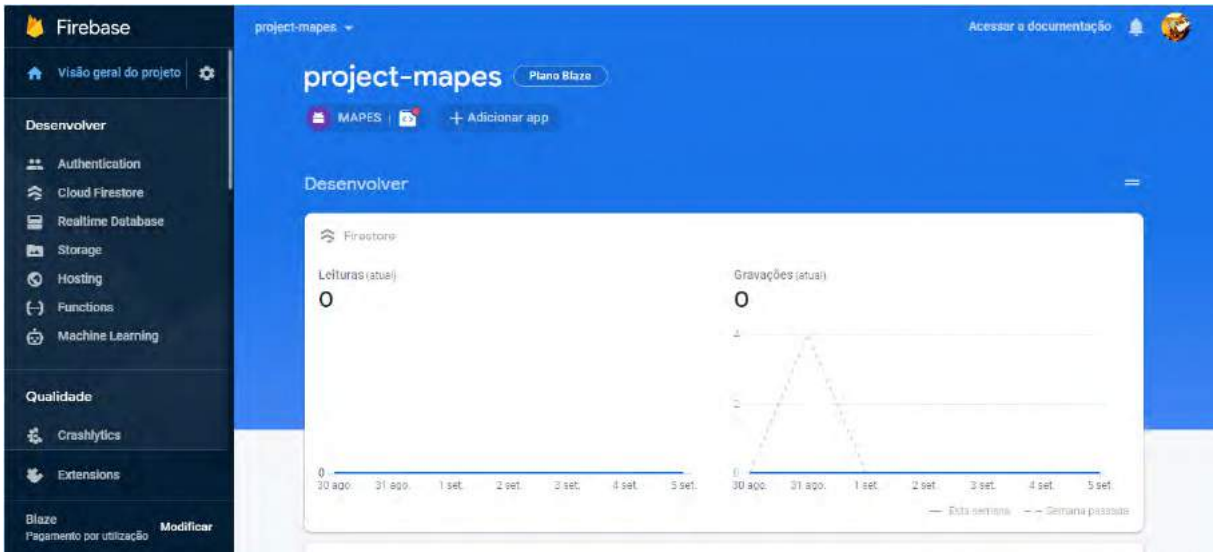
Para utilizar os recursos do Firebase, é necessário criar um projeto de acordo com esses passos (FIREBASE, 2020):

1. No **Console do Firebase**, clicar em **Adicionar projeto** e, em seguida, selecionar ou inserir um **Nome de projeto**.

2. *(Opcional)* Editar o **ID do projeto**. O Firebase atribui automaticamente um ID exclusivo ao projeto. **Não será possível alterar o ID do projeto depois que o Firebase provisionar os recursos para o projeto.**

3. Seguir as demais etapas de configuração no Console do Firebase e clicar em **Criar projeto**. O Firebase provisiona recursos automaticamente para o projeto. Quando o processo for concluído, será direcionado para a página de visão geral do do projeto no Console do Firebase.

Figura 17 – Console do projeto “project-mapes” no Firebase



Fonte: FIREBASE CONSOLE (console.firebase.google.com)

Os serviços que foram utilizados na implementação deste projeto foram (FIREBASE, 2020):

- *Authentication*: serviço para autenticar usuários na aplicação por meio de senhas, números de telefone e provedores de identidades federadas, tais como: Google, Facebook, Twitter entre outros. É possível fazer login no app utilizando a FirebaseUI, que é uma solução de autenticação simples e completa, ou o SDK do Firebase Authentication que permite utilizar os outros

métodos de login de forma manual. O ponto de entrada do *Firebase Authentication* é a classe *FirebaseAuth*. Primeiramente, é importante obter uma instância dessa classe através do método *getInstance()* e então para criar uma nova conta de usuário enviando o endereço de e-mail e a senha basta utilizar o método *createUserWithEmailAndPasswor*d*()* passando o e-mail e senha como parâmetros. E para que um usuário faça login com e-mail e senha, utiliza-se o método *signInWithEmailAndPasswor*d*()* passando os respectivos valores como parâmetros.

- *Cloud Firestore*: é um banco de dados de documentos NoSQL flexível e escalonável hospedado na nuvem que armazena e sincroniza os dados do app em tempo real. Ela mantém as informações atualizadas sem ter que recuperar todo o banco de dados nos diferentes aplicativos por meio de *listeners* em tempo real e oferece suporte offline para o desenvolvimento flexível de aplicações responsivas que funcionem independentemente da latência da rede ou da disponibilidade da Internet. O Firestore contém os dados estruturados em documentos que são pares de valores-chaves e em coleções que são os conjuntos de documentos. Ainda é possível criar subcoleções dentro dos documentos e definir uma hierarquia escalonável de estrutura de dados. Esses documentos podem conter diversos tipos de dados diferentes, desde *strings* e números a objetos complexos e aninhados. Com consultas expressivas, eficientes e flexíveis, é possível recuperar dados no nível do documento sem recuperar a coleção inteira ou qualquer subcoleção dentro dele. Permite classificação, filtros e limites às consultas e cursores para paginar os resultados. Além disso, utiliza *listeners* para manter os dados atualizados nos aplicativos, notificando com um instantâneo somente dos dados que foram alterados. A classe *FirebaseFirestore* representa um banco de dados e é o ponto de entrada para todas as operações do *Cloud Firestore.* Para inicializar uma instância que será responsável por executar as funções no banco de dados, utiliza-se o método *getInstance()*. Para navegar entre as coleções, utiliza-se o método *collection()*, passando como parâmetro o caminho. Para adicionar um documento, basta utilizar o método *set()* com o parâmetro do objeto que deseja armazenar. Utiliza-se o método *get()* junto com os métodos que atendam a uma determinada condição, como o *where()*, para consultar

coleções ou documentos. E para excluir documentos, utiliza-se o método *delete().*

- *Storage*: é um serviço para armazenamento de objetos (como imagens, vídeos e áudios), criado para veicular conteúdo gerado pelo usuário. Os arquivos são armazenados em um repositório do Google Cloud Storage e são acessados através do SDK do Firebase que permite lidar com perdas de conexão ao enviar ou baixar os arquivos e retornar de onde parou, utilizando a segurança do Google (*Authentication*). Além disso, oferece operações robustas e alta escalabilidade. Para iniciar é importante criar uma instância da classe *FirebaseStorage* através do método *getInstance().* Para fazer o upload de um arquivo, deve-se recuperar a referência completa do arquivo no Cloud Storage utilizando os métodos *getReference()* e *child()* que inclui o nome dele, e utilizar o método de upload *putBytes()*.

- *Cloud Functions*: serviço utilizado para integrar códigos que respondem a eventos dos recursos do Firebase e/ou solicitações HTTPS e executam alguma função específica. As funções foram codificadas em JavaScript (Node.js), armazenadas e executadas no Google Cloud Plataform, mas pode-se utilizar Java, Python, Go e C#. Os gatilhos ocorrem no Firebase, por exemplo, quando há uma mudança nos dados no Firestore e o servidor responde utilizando a função criada, por exemplo, para enviar uma notificação para a aplicação cliente. Nesse projeto, foi utilizado o *Functions SDK Admin* no ambiente do Node.js para hospedar o código da função com ajuda de ferramentas CLI (*Command Line Interface* – Interface de Linhas de Comando). O principal módulo que será utilizado do SDK será o *firebase-admin* que será responsável para enviar a mensagem com os dados úteis para a criação da notificação no dispositivo do responsável através do método *send()*.

4.4.NODE.JS

Node.js é um ambiente de execução JavaScript baseado em uma arquitetura *non-blocking thread* (não bloqueante), ou seja, executa processamento de várias conexões podem ser controladas ao mesmo tempo no servidor, evitando o

enfileiramento de requisições ociosas. Essa é uma plataforma assíncrona altamente escalável de baixo nível orientada a objetos, pois as funções são executadas em paralelo e aguardam o retorno por meio das funções de *callback*. Além disso, não sofrem de *dead-locks* e trabalham apenas em single-thread (uma instância por processo) e utiliza o *event-loop* que é o agente responsável por escutar e emitir os eventos no sistema em loop infinito (NODEJS.ORG, 2020; PEREIRA, 2015).

Esse ambiente foi utilizado para responder aos eventos do Geofence e criar os *push notifications* no aplicativo.

4.5.UML (UNIFIED MODELING LANGUAGE)

A UML, também conhecida como linguagem de modelagem unificada orientada a objetos, é uma linguagem visual que é usada para modelagem de projetos de softwares e que auxilia os analistas de sistemas a definirem as características e funcionalidades dos sistemas, tais como requisitos, ações, estrutura lógica, fluxo do processo, personagens, entre outros, de maneira que todas as pessoas envolvidas no processo de desenvolvimento possam compreendê-las. Essa modelagem utiliza notações gráficas, como diagramas, para representar essas características e funcionalidades e que podem ser criadas por softwares. (GUEDES, 2014)

Essa linguagem auxilia nos processos de especificação, visualização, documentação e desenvolvimentos de software utilizando o paradigma de orientação a objetos. Através de seus variados tipos de diagramas, é possível representar projetos de software sob diversas óticas sintetizando os processos e relacionamentos envolvidos e ao mesmo tempo ignorar detalhes que não são relevantes naquele momento. (VARGAS, 2008)

A UML surgiu em 1996 oriunda da união das metodologias Booch de Grady Booch, da OMT (*Object Modeling Technique)* de Ivar Jacbson e da OOSE (*Object-Oriented Software Engineering*) James Rumbaugh, que na época eram as mais populares em modelagem de orientação a objetos, e do apoio da empresa Rational Software Coporation. No ano seguinte, várias empresas contribuíram com o projeto, como a Hewlett-Packard, IBM, Microsoft, Oracle, ICON Computing, Eletronic Data

Services, ObjectTime, Platinum, Reich, Softeam, entre outras, e com isso, a OMG[5] (*Object Management Group*) adotou a UML como a linguagem padrão de modelagem de software. Em julho de 2005, a versão 2.0 da linguagem foi lançada oficialmente, estando essa atualmente na versão 2.5 (SILVA & VIDEIRA, 2001).

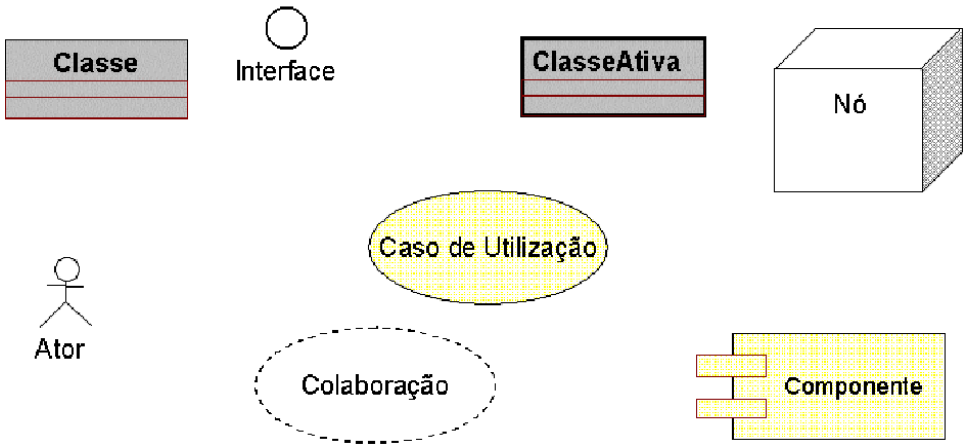
4.5.1. Tipos de Elementos e Diagramas UML

Os diagramas da UML modelam aspectos estruturais e comportamentais do software a ser desenvolvido, ou seja, pode especificar as regras do negócio, relacionamentos, estados, sequência de atividades e de colaborações. Esses diagramas provêm entendimento e a utilização por humanos e máquinas, através dos elementos gráficos de notações padronizados e de um mecanismo para mapeamento entre a representação gráfica do diagrama e a textual em XML (*Extensible Markup Language*), que é uma linguagem extensível de marcação largamente utilizada, o que possibilita a exportação desses dados para outras ferramentas (PEREIRA, 2011).

Cada elemento gráfico dessa linguagem possui uma sintaxe e uma semântica, ou seja, uma notação padronizada que possui um significado e o objetivo de sua representação, os quais podem ser aplicados em definições de problemas variados e a diferentes níveis de abstração. Cada notação gráfica pode ser considerada básico, que definem os modelos; de relacionamento, que interligam os elementos básicos; e diagramas, que são utilizados para agrupar os elementos. Essas notações são organizadas de acordo com a sua funcionalidade ou responsabilidade. Portanto, existem elementos de estrutura, comportamento, agrupamento e de anotação (SILVA & VIDEIRA, 2001). As figuras 18 e 19 ilustram os principais elementos anteriormente mencionados.

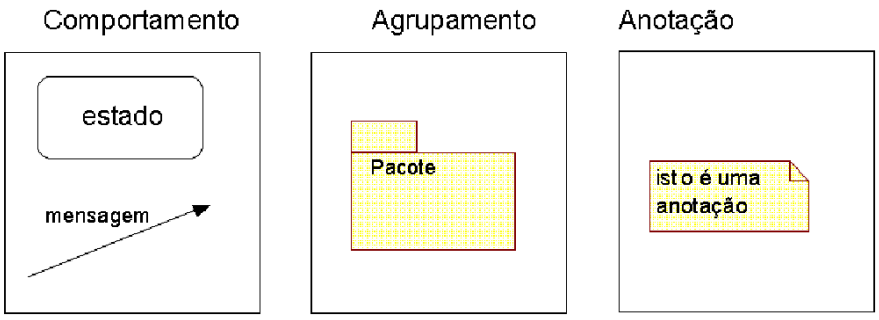
[5] OMG: também conhecido como Grupo de Gerenciamento de Objetos, é um consórcio internacional de empresas que define e ratifica padrões na área de orientação a objetos.

Figura 18 – Resumo dos elementos de estrutura



Fonte: SILVA & VIDEIRA, 2001

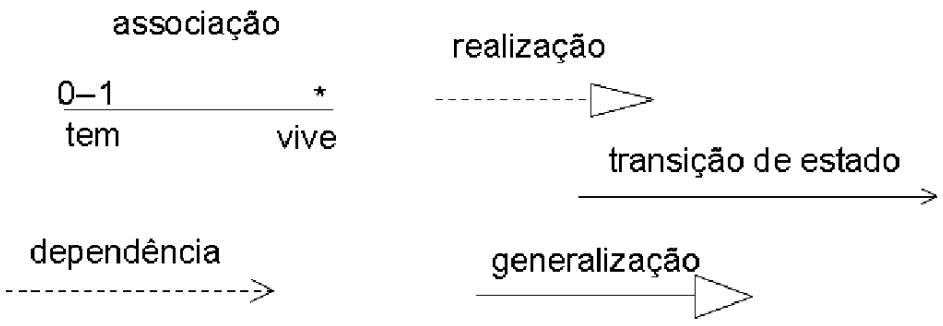
Figura 19 – Resumo dos elementos de comportamento, agrupamento e anotação



Fonte: SILVA & VIDEIRA, 2001

Outros elementos gráficos importantes são os que representam a relação de interdependência entre os elementos básicos e que apresentam uma sintaxe e uma semântica bem definida. A figura 20 ilustra os principais tipos de relação da UML, como associação, realização, dependência, transição de estado e generalização (SILVA & VIDEIRA, 2001).

Figura 20 – Resumo dos tipos de relações

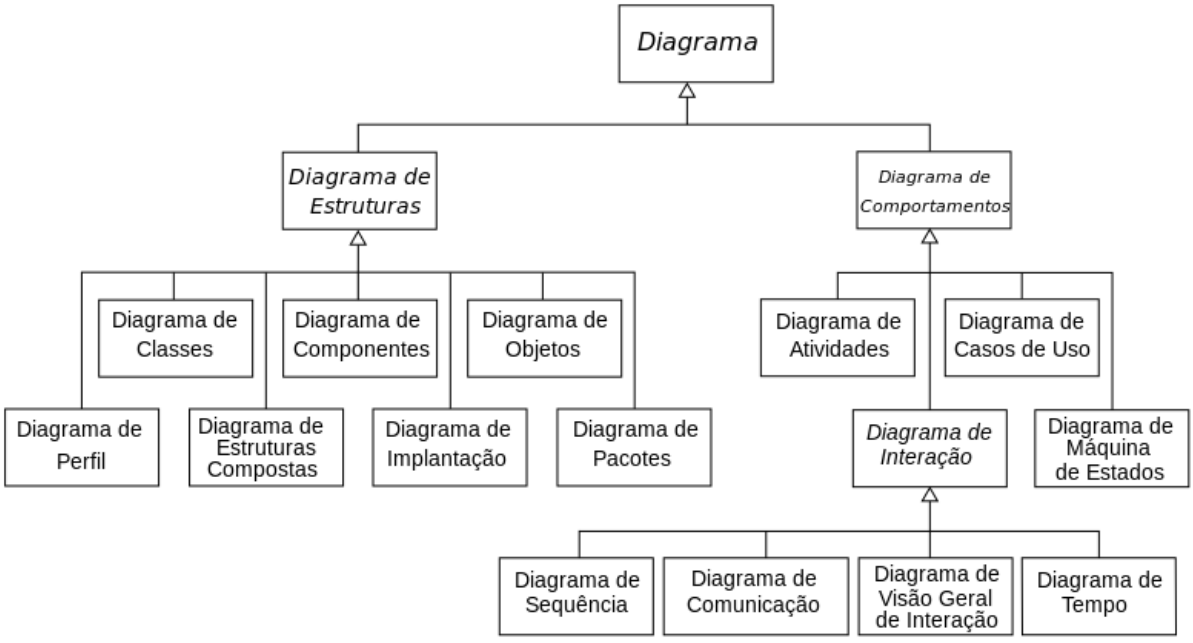


Fonte: SILVA & VIDEIRA, 2001

A possibilidade de agrupar elementos básicos e suas relações é traduzida em diagrama. A UML dispõe de diferentes tipos de diagramas. Cada um deles utiliza um subconjunto de elementos básicos que analisa o sistema, ou parte dele, e fornece uma determinada visão do projeto; alguns diagramas focam na perspectiva de forma mais geral e outros se preocupam em demonstrar um viés mais técnico ou até mesmo, visualizar profundamente uma característica específica do sistema ou de um processo. Assim, é possível utilizar vários diagramas que se complementam para compreender o projeto como um todo, além de identificar erros, incompatibilidades e possibilitando tomadas de decisão (GUEDES, 2014).

Os diagramas UML são divididos em duas categorias: estrutural e estática. A parte estática de um projeto de software pode ser representada pelos diagramas de classes, objetos, componentes e implantação, onde é possível visualizar as classes, interfaces, componentes, nós e outros; já a parte dinâmica, consideradas comportamentais, podem ser representadas pelos diagramas de casos de uso, sequência, colaboração, estados e atividades, que indicam as partes do sistema que sofre algum tipo de alteração (ROSSONI, 2011). A figura 21 representa a hierarquia dos diagramas.

Figura 21 – Hierarquia dos Diagramas UML



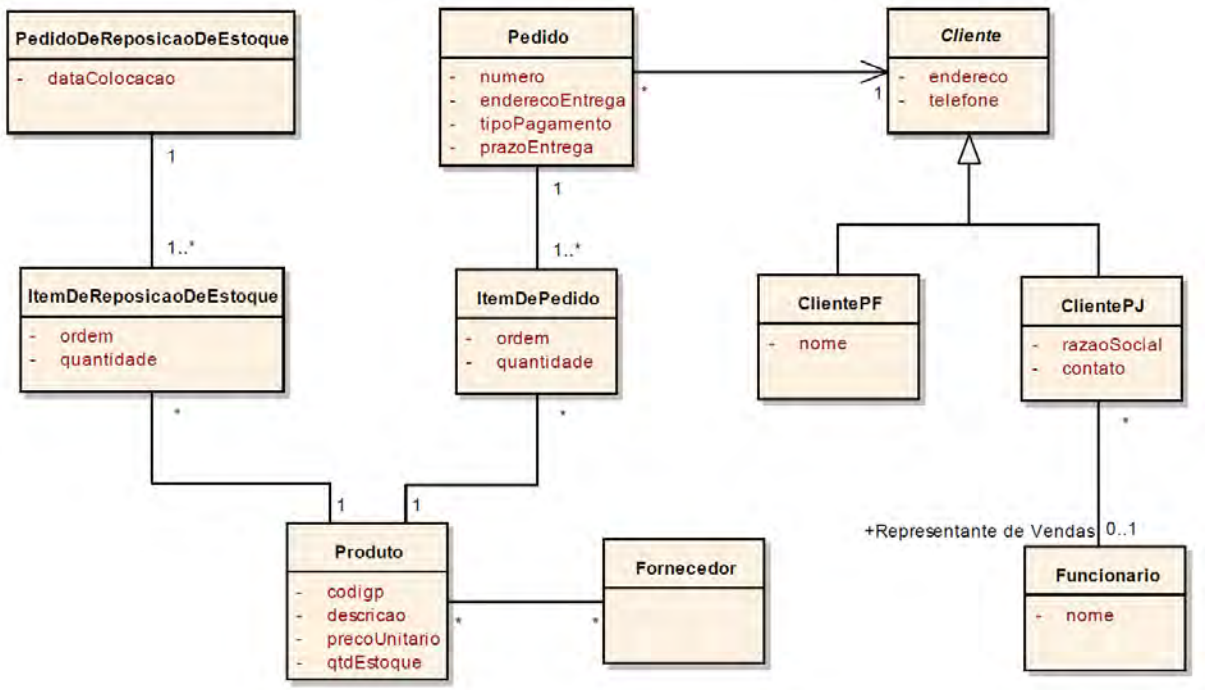
Fonte: Adaptado de OMG Unified Modeling Language™, *Superstructure 2.4.1, Figura A.5*

A modelagem do projeto em questão será representada por diagramas de classes, casos de uso e sequência e por isso, apenas esses diagramas UML serão descritos.

4.5.2. Diagrama de Classes e seus componentes

Um diagrama de classe apresenta um conjunto de classes, interfaces, colaborações e seus relacionamentos. Esses diagramas abrangem uma visão estática da estrutura do sistema que é direcionada pelas classes e normalmente são utilizados para construir a modelagem orientados a objetos. As classes são blocos de construção que descrevem um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica (BOOCH, RUMBAUGH & JACOBSON, 2000). É possível visualizar um exemplo do diagrama de classes na Figura 22.

Figura 22 – Exemplo de Diagrama de Classes

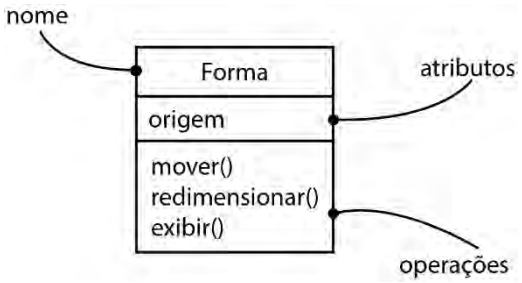


Fonte: PEREIRA, 2011

É possível utilizar esse diagrama para providenciar uma abstração definida a partir do vocabulário do domínio do sistema que está em desenvolvimento, agrega um conjunto restrito de e definido de responsabilidades, separa claramente a

especificação abstrata da implementação, além de representar a relação entre itens do software, hardware e até itens considerados puramente conceituais (BOOCH, RUMBAUGH & JACOBSON, 2000). A notação gráfica de uma classe e seus componentes mais relevantes é apresentada na Figura 23.

Figura 23 – Notação gráfica UML de uma Classe

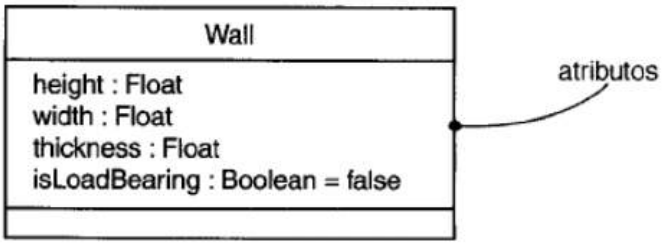


Fonte: BOOCH, RUMBAUGH & JACOBSON, 2000

Cada classe deve possuir um nome único diferente de todas as outras classes que estão no mesmo pacote. Na prática, esses nomes são substantivos ou expressões breves compostas por caracteres textuais e especiais (exceto dois pontos) que são definidos a partir do vocabulário do sistema. Tipicamente, o primeiro caractere de cada palavra existente no nome da classe aparece como maiúsculo, como em *Cliente* ou *SensorTemperatura* (BOOCH, RUMBAUGH & JACOBSON, 2000).

Na segunda secção da representação da classe, é possível identificar os atributos que são as propriedades compartilhadas por todos os objetos de uma classe. Essa classe pode ter nenhum ou qualquer número de atributos possíveis. O nome de um atributo pode ser definido como o nome das classes e normalmente, o primeiro caractere de cada palavra existente no nome do atributo aparece em maiúsculo, exceto a primeira letra, como em *nome* ou *dataNascimento.* Na definição dos atributos, é possível visualizar apenas o seu nome e opcionalmente os respectivos tipos, a visibilidade (se é um atributo público, privado, protegido), multiplicidade, valor inicial ou outras qualificações (se é constante) (SILVA & VIDEIRA, 2001). Na Figura 24, é possível visualizar o atributo *isLoadBearing* da classe *Wall* representado pelo seu nome, tipo e valor inicial, respectivamente.

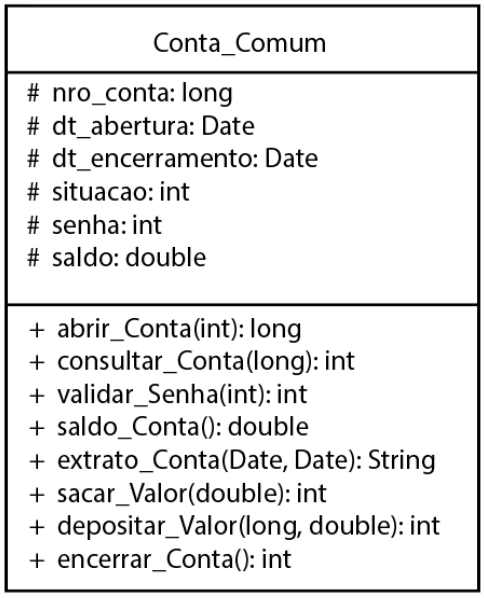
Figura 24 – Atributos da classe *Wall*



Fonte: BOOCH, RUMBAUGH & JACOBSON, 2000

Na definição das operações (ou métodos) é possível visualizar apenas seus nomes, ou adicionalmente as respectivas assinaturas com seus parâmetros e tipos de retorno, ou outras qualificações, como por exemplo a visibilidade do método, assim como a do atributo, que pode ser representado pelos símbolos de sustenido (#), de adição (+) de subtração (-) e de til (~) para definir a visibilidade privada, protegida, pública e pacote, respectivamente (GUEDES, 2014). A classe *Conta_Comum* é representada com seus atributos e métodos mais detalhados na Figura 25.

Figura 25 – Detalhamento dos atributos e assinaturas dos métodos



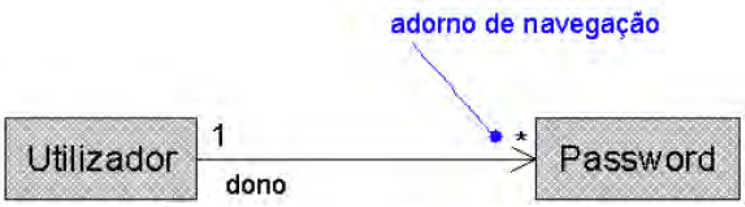
Fonte: GUEDES, 2014

As classes são conectadas através de um relacionamento, que permite que elas compartilhem informações entre si e auxiliem na execução dos processos do sistema. Em uma modelagem orientada a objetos, os três tipos de relacionamentos importantes são as associações, as dependências e as generalizações, que são representados graficamente por diferentes tipos de linhas (BOOCH, RUMBAUGH & JACOBSON, 2000), como as apresentadas anteriormente na Figura 22.

A associação é um relacionamento estrutural que descreve um vínculo que ocorre entre objetos de uma ou mais classes e o fluxo de navegação entre eles. Esse tipo de relacionamento pode ter nome ou título para auxiliar a compreensão do tipo de vínculo existente entre os objetos das classes envolvidas. Quando a associação retorna à mesma classe, é conhecida como associação unária, ou seja, existe o vínculo entre os objetos da mesma classe. A associação estabelecida entre duas classes distintas é chamada de binária. A associação ternária ou n-árias conectam mais de duas classes, que é representada por um losango para conectar as ligações da associação (GUEDES, 2014).

As associações podem representar outras características, tais como multiplicidade e navegação. A multiplicidade indica o número de instâncias de uma classe que pode se relacionar com uma única instância da outra classe envolvida, por exemplo, multiplicidade muitos (*), um ou mais (1..*), exatamente um (1), zero ou um (0..1) e outros. A navegação traduz o fluxo da interação entre as instâncias de uma classe em relação às instâncias de outra classe. Por omissão, o a navegação é bidirecional, contudo, existem situações que é necessário representar uma associação unidirecional (SILVA & VIDEIRA, 2001). A Figura 26 exemplifica o uso da multiplicidade e da navegação na associação.

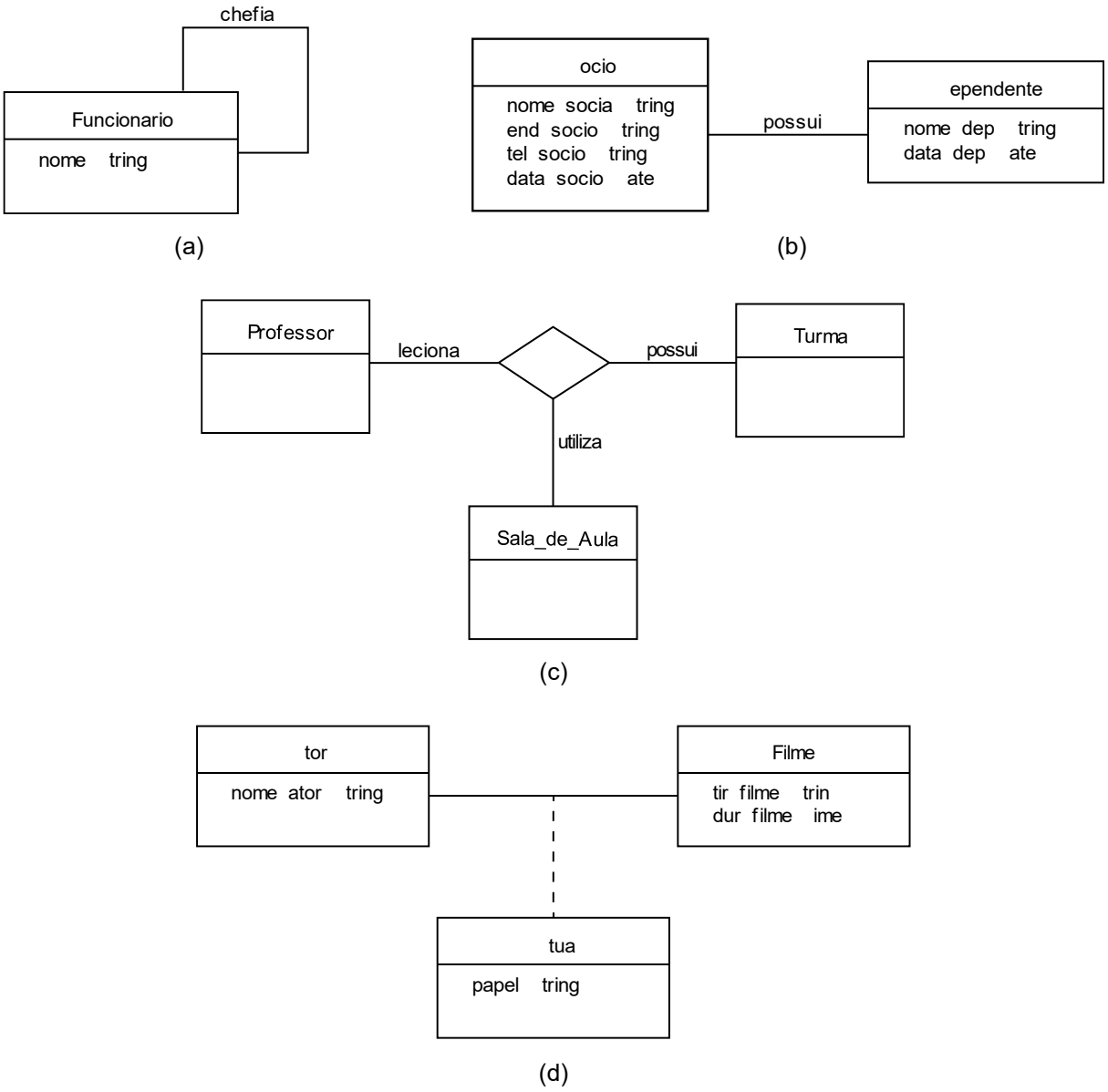
Figura 26 – Multiplicidade e navegação da associação



Fonte: SILVA & VIDEIRA, 2001

Existe ainda, a classe associativa que são produzidas baseadas na associação entre outras classes. Essas classes associativas contêm atributos que não podem ser armazenados por nenhuma das classes envolvidas (GUEDES, 2014). A Figura 27 exemplifica os três tipos de associações e a classe associativa.

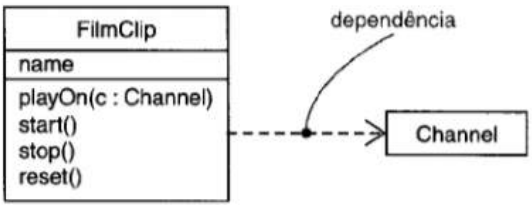
Figura 27 – Exemplos de Associação Unária (a), Binária (b), Ternária (c), Classe Associativa (d), respectivamente



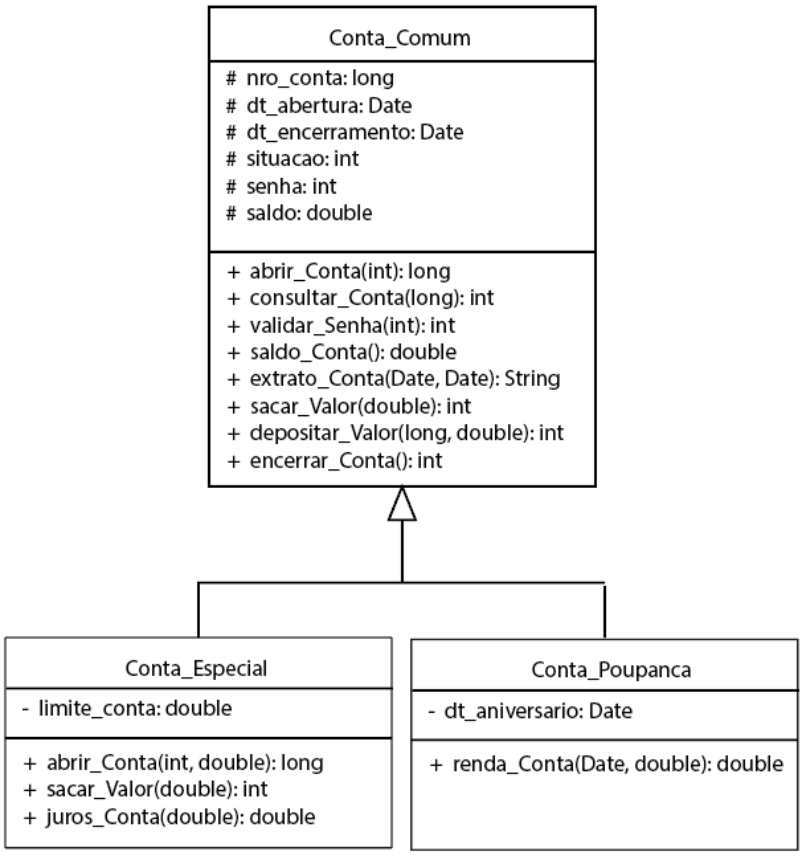
Fonte: GUEDES, 2014

Já a dependência é um relacionamento utilizado para explicitar o grau de dependência de uma classe em relação à outra e que determina as modificações na especificação de uma classe. Enquanto a generalização ou especificação é um relacionamento entre classes denominadas superclasses ou classes-mãe e tipos mais específicos chamadas subclasses ou classes-filha, demonstrando os atributos comuns, a hierarquia entre as classes e a possibilidade de utilização dos métodos polimórficos nas classes especializadas (BOOCH, RUMBAUGH & JACOBSON, 2000; GUEDES 2014). Exemplos de Associação de dependência e generalização estão representados na Figura 28.

Figura 28 – Exemplos de Associação de dependência (a) e de generalização (b)



(a)



(b)

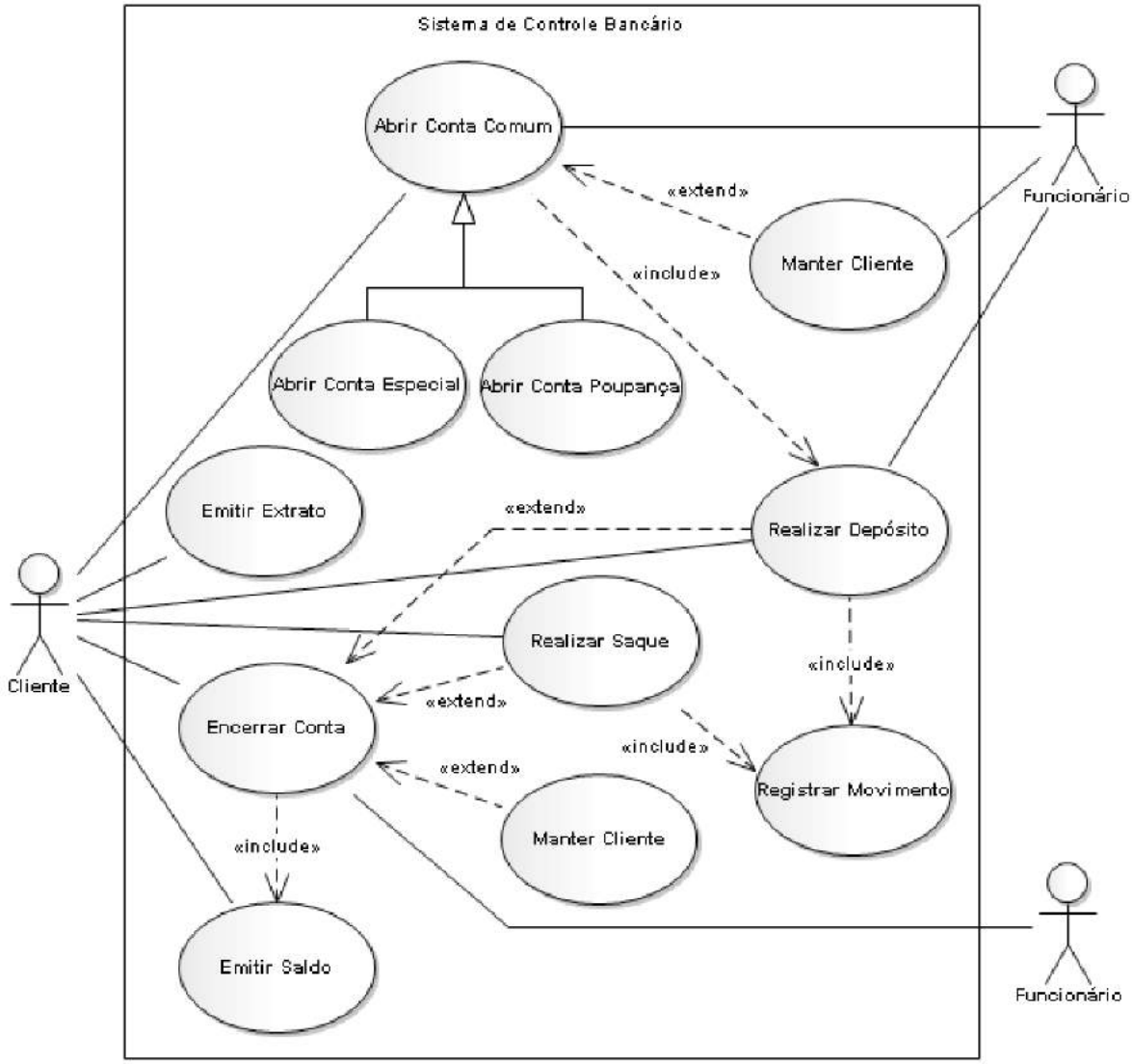
Fonte: BOOCH, RUMBAUGH & JACOBSON, 2000; GUEDES, 2014

4.5.3. Diagrama de Caos de Uso e seus componentes

Esse diagrama é o mais geral e informal da UML, utilizado, normalmente, na fase inicial do projeto, que é a de levantamento e análise de requisitos do sistema e possui o objetivo de identificar o maior número de recursos e serviços que o sistema poderá executar, sem estar preocupado em detalhar informações de implementação. Ele representa uma visão sistêmica e de alto nível do software e possui uma linguagem simples e de fácil compreensão para os *stakeholders*, além de servir de

base para a produção de outros diagramas complementares. Os elementos básicos desse diagrama são os atores que utilizarão o software, que representam os usuários, outros sistemas ou algum hardware especial, e as funcionalidades que o sistema executará, conhecidas como casos de uso nesse diagrama (GUEDES, 2014). A Figura 29 demonstra um exemplo do diagrama.

Figura 29 – Exemplo de Diagrama de Casos de Uso



Fonte: GUEDES, 2014

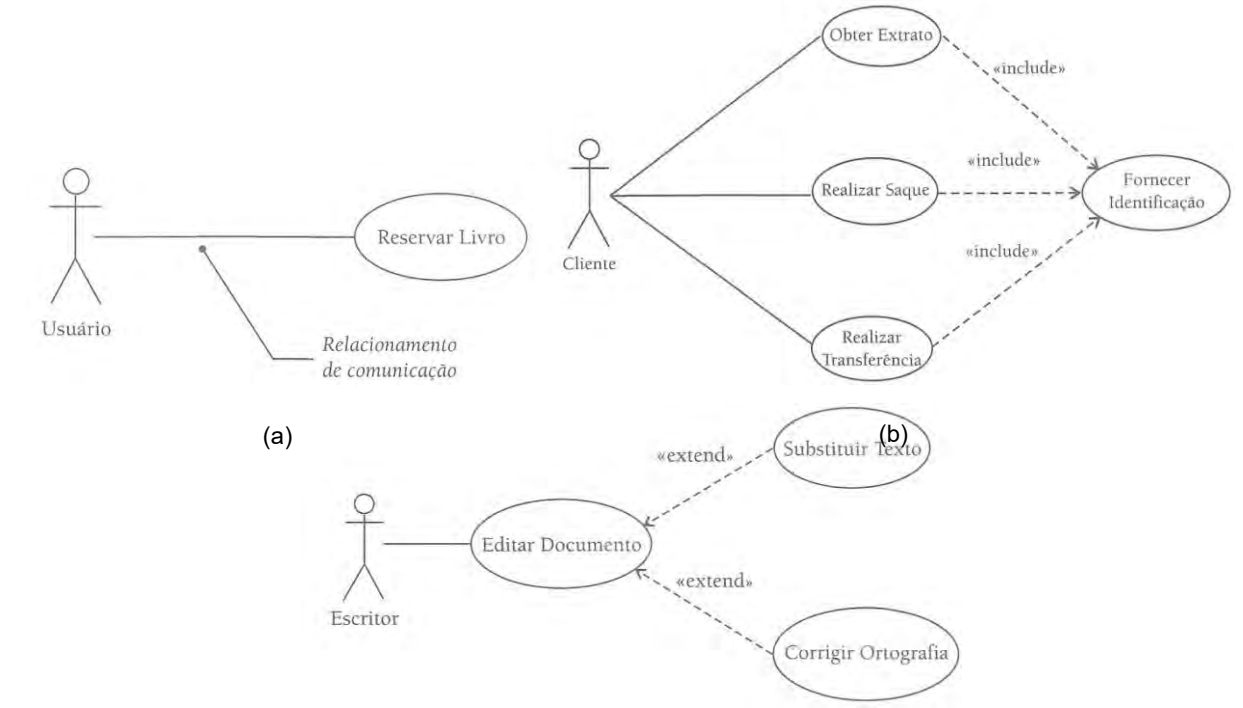
Os atores representam o papel que qualquer elemento externo desempenha quando executa alguma ação no sistema que será desenvolvido. Esses atores podem ser seres humanos assim como dispositivos eletrônicos ou outros sistemas com quem podem se relacionar. É uma boa prática de modelagem nomear o ator com o nome do papel desempenhado dentro do contexto. Além disso, um ator pode participar de

vários casos de uso (PEREIRA, 2011), como no exemplo apresentado anteriormente na Figura 19, onde o ator Cliente participa de 5 casos de uso.

Os casos de uso correspondem às ações executadas pelos atores interagindo com o sistema através da interação entre as funções do sistema e os usuários dentro de um cenário principal e alternativo. No diagrama, o caso de usos é representado graficamente por uma elipse com oração que expressa uma determinada ação. Portanto, essa frase desse ser definida na voz ativa e começar com um verbo no infinitivo (SILVA & VIDEIRA, 2001). Exemplos deles podem ser visualizados na Figura 29.

O terceiro componente desse diagrama é o relacionamento, que tem a função de relacionar os atores aos casos de uso. Podem existir relacionamentos entre atores e mais de um caso de uso, entre casos de uso e até entre atores. Os seguintes relacionamentos são definidos na linguagem: comunicação, inclusão, extensão e generalização. Os relacionamentos de comunicação informam a associação entre o ator e o caso de uso e a troca de informações entre eles e é o mais utilizado; os relacionamentos de inclusão associam somente os casos de uso e é utilizado para representar uma sequência de interação entre eles, o que pode incluir o comportamento de um caso de uso comum; os relacionamentos de extensão são utilizados para modelar comportamentos eventuais que ocorrem sob certas condições ou que a realização depende de uma decisão do ator e que não deve necessariamente existir dependência de quaisquer casos de uso que estendam o primeiro; e os relacionamentos de generalização podem existir entre dois casos de uso ou entre dois atores e são utilizados para representar a herança de características de um caso de uso ou ator mais genérico (BEZERRA, 2007). É possível observar alguns exemplos dos tipos de relacionamentos mencionados na Figura 29.

Figura 30 – Exemplos de relacionamentos: (a) comunicação; (b) inclusão; (c) extensão; e (d) generalização



Fonte: BEZERRA, 2007

4.5.3.1 Cenários e especificações de casos de uso

A descrição textual dos casos de uso especifica as ações na ordem em que elas devem ser executadas e deve ser produzida em um formato padronizado para o projeto. Essa etapa é conhecida como especificação ou descrição dos casos de uso.

Essa especificação valida a interação entre os atores e os casos de uso que foram destacados no diagrama, seguindo a ordem por ele descrita (PEREIRA, 2011).

O cenário ilustra exatamente essa sequência de ações que está relacionada ao comportamento do sistema. É a descrição textual de um ou mais fluxos de ações de forma clara e concisa para que todos possam entender sem dificuldades. Essa descrição deve conter: Início e fim do caso de uso, em que momento acontece a interação com os atores, quais informações são trocadas, definição do fluxo de ações principal e alternativo. O fluxo principal corresponde a sequência normal das ações. Já o fluxo alternativo corresponde a uma exceção ou um caminho alternativo. Outras informações complementares a essa especificação são as pré e pós condições, atores que interagem com o caso de uso e que são influenciados por ele, utilização de diagramas de sequência, e outros (SILVA & VIDEIRA, 2001). Como exemplo, a Figura 31 ilustra a especificação do caso de uso “Validar Utilizador”

Figura 31 – Especificação textual do caso de uso “Validar Utilizado”

Nome: Validar Utilizador

Cenário Principal

O caso de utilização inicia-se quando o sistema apresenta um ecrã a pedir ao cliente o seu cartão electrónico. O cliente introduz o seu cartão MB e, através de um pequeno teclado, o seu PIN. Note-se que o cliente pode limpar a introdução do seu PIN inúmeras vezes e reintroduzir um novo número antes de activar o botão “Entrar”. O cliente activa o botão “Entrar” para confirmar. O sistema lê o PIN e a respectiva identificação do cartão MB, e verifica se é válido. Se o PIN for válido o sistema aceita a entrada e o caso de utilização termina.

Cenário Alternativo 1 (Cliente cancela operação)

O cliente pode cancelar a transação em qualquer momento activando o botão “Cancelar”, implicando a reinicialização do caso de utilização. Não é realizada qualquer alteração à conta do cliente.

Cenário Alternativo 2 (PIN inválido)

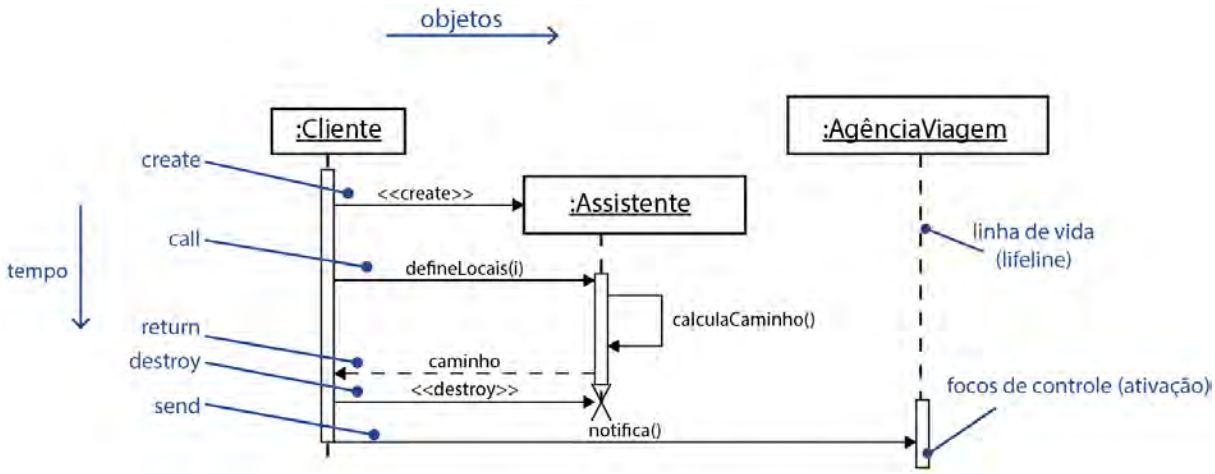
Se o cliente introduz um PIN inválido, o cartão MB é ejectado e o caso de utilização reinicializado. Se tal ocorrer 3 vezes consecutivas, o sistema cativa (i.e., “recolhe”) o cartão MB e cancela a transação; não permitindo qualquer interacção nos 2 minutos seguintes.

Fonte: SILVA & VIDEIRA, 2001

4.5.4. Diagrama de Sequência e seus componentes

O diagrama de sequência ilustra a interação e a colaboração entre objetos em um sistema baseada em uma perspectiva temporal. Esse diagrama é representado através de duas dimensões: a horizontal que representa o conjunto de objetos atuantes (atores, classes e objetos); e a vertical que representa a progressão do tempo. Existem algumas notações particulares desse diagrama, tais como: linhas de vida, envio de mensagens, ocorrências de execução, criação e destruição de objetos (BEZERRA, 2007). Um exemplo do diagrama de sequência é ilustrado na Figura 32.

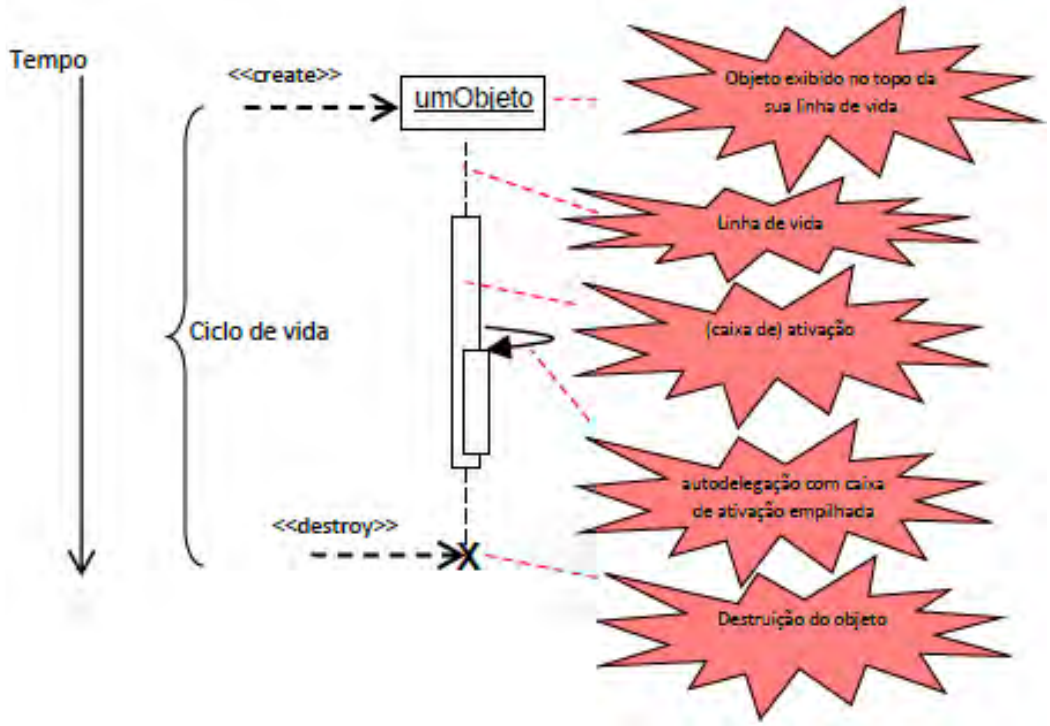
Figura 32 – Exemplo de um diagrama de sequência



Fonte: SILVA & VIDEIRA, 2001

A linha de vida é utilizada para representar a participação dos itens que realizam alguma ação. Representam também o foco do controle, ou seja, a ativação ou a desativação de objetos ou a delegação a outro objeto. Essa linha de vida é composta por duas partes: a cabeça que é a representação do objeto atuante; e a cauda que corresponde a uma linha vertical tracejada indo até o final do diagrama ou até a sua remoção (PEREIRA, 2011). A Figura 33 ilustra ciclo de vida de um objeto em um diagrama de sequência.

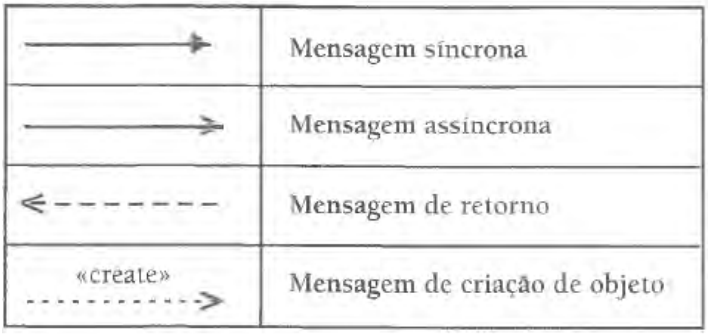
Figura 33 – O ciclo de vida de um objeto representado em um diagrama de sequência



Fonte: PEREIRA, 2011

A interação entre os objetos do diagrama é realizada através de mensagens. Uma mensagem é a especificação dos papéis que os objetos emissor e receptor devem acordar para que determinada ação seja realizada, onde o objeto emissor solicita a execução de uma operação definida no objeto receptor. A notação para uma mensagem, geralmente, é uma flecha desenhada na horizontal que liga uma linha de vida a outra, partindo do objeto remetente em direção ao objeto receptor, além de ser rotulada com, pelo menos, o nome da função e os valores de seus parâmetros (BEZERRA, 2007). Os formatos possíveis dos tipos de mensagens possíveis estão ilustrados na Figura 34.

Figura 34 – Mensagens utilizadas no Diagrama de Sequências



Fonte: BEZERRA, 2007

Uma mensagem síncrona indica que o objeto remetente espera o processamento da mensagem pelo objeto receptor antes de recomeçar o seu processamento. Usando uma mensagem assíncrona, o objeto remetente não precisa aguardar término do processamento pelo objeto receptor. A mensagem de retorno especifica o retorno do processamento de uma mensagem enviada anteriormente. Uma mensagem de sinal é usada para enviar um sinal, como por exemplo, a criação ou a destruição de um objeto. É possível que um objeto possa enviar uma mensagem para ativar uma operação em sua própria classe, enviando uma mensagem reflexiva (PEREIRA, 2011; BEZERRA 2007).

5. IMPLEMENTAÇÃO DA SOLUÇÃO

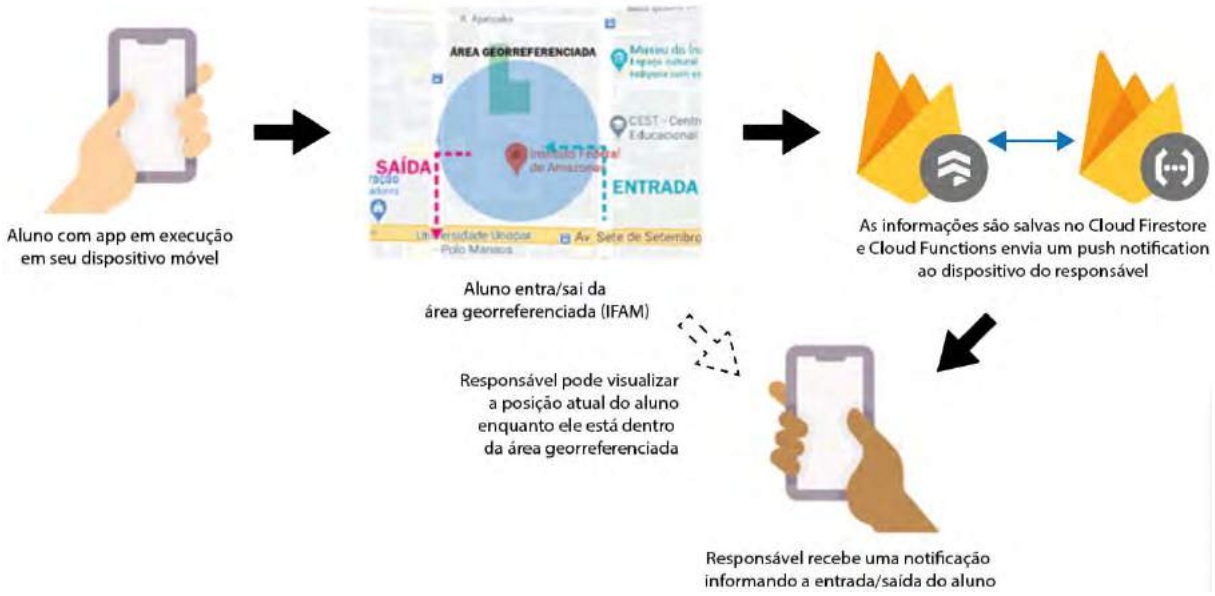
Neste capítulo, será apresentada as diversas etapas para o desenvolvimento da solução denominada MAPES para o problema definido no Capítulo 1. Portanto, será descrita, em maiores detalhes, a visão geral do aplicativo, a definição dos requisitos – funcionais e não funcionais –, a sua modelagem UML e finalmente a implementação do app para dispositivo móvel Android.

5.1. VISÃO GERAL

Este trabalho descreve o desenvolvimento de um aplicativo capaz de enviar notificações aos usuários responsáveis quando seus filhos entrarem ou saírem das dependências do IFAM, além de permitir a visualização, em tempo real, da localização e da movimentação de seus filhos dentro da instituição. Foi dado o nome a este projeto de MAPES visto que ele monitora o aluno considerando a posição atual, a entrada e a saída.

O aplicativo mobile foi desenvolvido utilizando a linguagem Java para Android integrada com os recursos *Authentication, Cloud Firestore, Cloud Storage* e *Cloud Functions* do Firebase. A Figura 35 apresenta uma visão geral do app.

Figura 35 – Visão geral do aplicativo



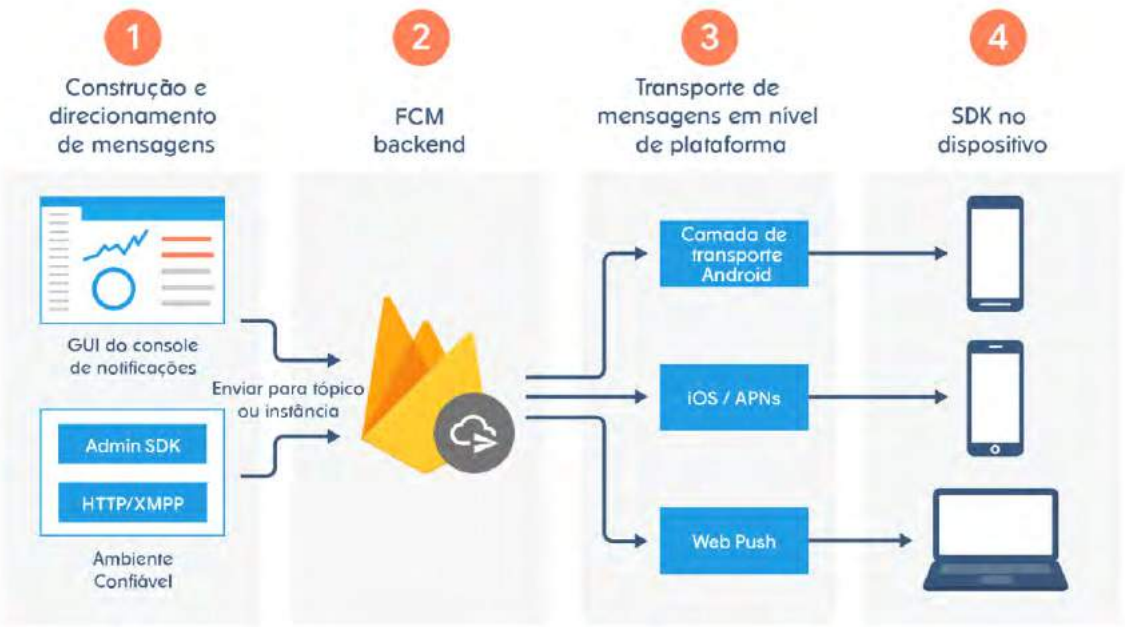
Fonte: próprio autor

O aplicativo MAPES utiliza perfis de usuários para definir suas funcionalidades: perfil do *aluno* e perfil do *responsável*. Através do perfil do aluno, o app que está em execução em um dispositivo móvel detecta quando o usuário entra ou sai da área georreferenciada, que no projeto proposto é a área do IFAM. Após esse evento, o aplicativo salva as informações sobre a chegada ou a saída no *Cloud Firestore* do Firebase e que por sua vez, o *Cloud Functions* detecta esse evento e dispara um *push notification* para o aplicativo que está sendo executado no dispositivo do responsável. Ademais, controla a ativação e desativação dos recursos de localização do smartphone que são utilizados para coleta de suas coordenadas geográficas (longitude e latitude) através do intervalo de tempo configurado pelo usuário. Usando o perfil de *responsável*, o usuário poderá visualizar as notificações de entrada e saída do aluno, a localização atual de seu filho e a movimentação dele dentro da escola.

O principal recurso que foi utilizado é o *Google Fences API* que cria fronteiras geográficas virtuais combinando a localização atual do usuário com a proximidade em relação a locais de interesse que podem ser monitorados. Essas fronteiras são marcadas através da especificação da latitude, longitude e o raio que criam uma área circular ao redor do local de interesse. O aplicativo recebe *callbacks* quando um usuário entra ou sai da região do *geofence*.

Para implementar o envio e recebimentos de notificações, foi utilizado o serviço de envio de mensagens FCM (*Firebase Cloud Messaging*) que utiliza os protocolos do servidor FCM por meio do ambiente do *Cloud Functions* para Firebase ou do servidor de apps próprio utilizado para criar, segmentar e enviar mensagens. A visão geral da arquitetura do FCM é apresentada na Figura 36.

Figura 36 – Visão geral da arquitetura do FCM



Fonte: GOOGLE FIREBASE (adaptado)

As solicitações de mensagens podem ser criadas usando o Editor de Notificações baseado em GUI do Firebase ou um ambiente de servidor confiável compatível com o SDK Admin do Firebase ou os protocolos do Servidor FCM. Esse ambiente pode ser o próprio servidor de aplicativos, o *Google App Engine* ou o *Cloud Functions* para o Firebase (ANDROID DEVELOPERS, 2019), sendo este o que foi escolhido no desenvolvimento do aplicativo.

Após a criação da mensagem e o envio ao backend do FCM, este recebe a solicitação, gera um código de mensagem e outros metadados e os envia à camada de transporte específica da plataforma que está sendo utilizada. Essa camada, por sua vez, encaminha a mensagem para o dispositivo de destino quando este estiver online, processa a entrega e aplica a configuração específica para cada plataforma. Essa camada de transporte inclui:

- Camada de transporte do Android (ATL) para dispositivos Android com o Google Play Services
  Serviço de notificação *push* da Apple (APNs) para dispositivos iOS
- Protocolo de push da Web para aplicativos da Web

No dispositivo final, o app cliente recebe a mensagem ou a notificação (GOOGLE FIREBASE, 2019)

5.2. REQUISITOS DO APLICATIVO

O gerenciamento de requisitos é uma atividade valiosa e fundamental durante o processo de desenvolvimento de software/aplicativo. Essa atividade tem por objetivo designar os atributos para atender os requisitos do software que estabelecem as funções que deve executar e as restrições aplicáveis à sua operação, além de definir formas de priorizá-los, rastreá-los e visualizá-los. Essa etapa facilita a compreensão dos serviços de desempenho, restrição de hardware, dentro outros (MENDES, COSTA & LORENSO, 2015).

Os atributos dos requisitos podem ser classificados como requisitos funcionais ou requisitos não funcionais (também conhecido como requisitos de qualidade). Os requisitos funcionais são as funções que o software deve executar, as suas reações relacionadas a entradas específicas e os comportamentos em determinadas situações. Os requisitos não funcionais referem-se aos critérios qualificadores dos requisitos funcionais que trarão ao usuário melhor experiência no uso do software ou às restrições sobre serviços ou funções executados pela aplicação (PAULA FILHO, 2000).

Nesse contexto, partindo do levantamento realizado sobre as funcionalidades dos trabalhos relacionados, os requisitos funcionais (RF) aplicativo MAPES são:

Tabela 2 – Requisitos Funcionais

Código	Descrição	Requisitos Relacionados
RF01	Cadastrar os perfis de usuários, opcionalmente utilizando a integração com as contas do Google ou do Facebook	-
RF02	Executar a autenticação através de e-mail e senha	[RF01]
RF03	No dispositivo do filho (aluno), capturar as informações geográficas da posição atual e salvar no banco de dados em um intervalo de tempo estabelecido	[RF02]
RF04	Utilizando o perfil Responsável, adicionar vínculo com o Perfil Aluno estabelecendo uma senha para aceitação	[RF01] e [RF02]

RF05	Visualizar em tempo real a posição atual do filho (Aluno) através do dispositivo móvel do pai (Responsável) utilizando API do Google Maps	[RF02] e [RF04]
RF06	No dispositivo do pai (Responsável), receber notificações de entrada e saída da área georreferenciada do aluno	[RF02] e [RF04]
RF07	Utilizando o perfil Responsável, consultar as notificações de entrada e saída durante um intervalo de datas	[RF02] e [RF04]
RF08	Visualizar histórico de rotas dentro da área georreferenciada através do perfil Responsável	[RF02], [RF03] e [RF04]

Os requisitos não funcionais (RNF) são:

Tabela 3 – Requisitos Não Funcionais

Código	Descrição
RNF01	Permitir a configuração do tempo de captura dos dados geográficos no dispositivo do filho (aluno) para que não comprometa a eficiência da bateria.
RNF02	Suportar o cadastro de mais de um perfil Aluno associado ao perfil Responsável, assim como mais de um perfil Responsável associado a um perfil Aluno
RNF03	Permitir que somente um perfil (Responsável/Aluno) seja configurado em um dispositivo móvel.
RNF04	O aplicativo será executado em sistema operacional Android

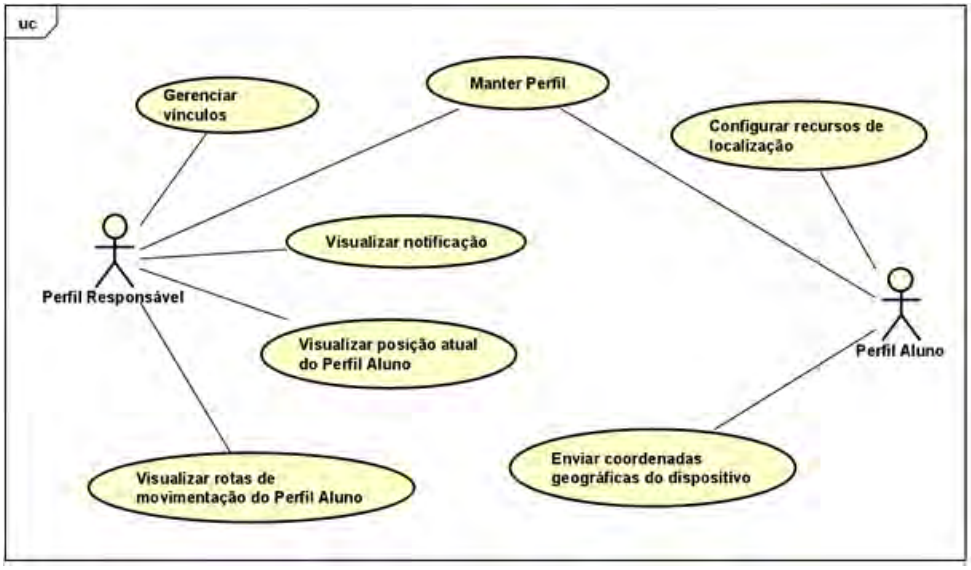
5.2.1. Modelagem

Para a modelagem do aplicativo MAPES, foram utilizados os seguintes diagramas UML: Diagrama de Casos de Uso, Diagrama de Classe e Diagrama de Sequência; e foram elaborados utilizando-se a ferramenta *Astah Community*, desenvolvida em linguagem Java.

5.2.1.1 Diagrama de Casos de Uso

Segundo Sommerville (2011), os digramas de Casos de uso são utilizados para auxiliar durante a fase de descoberta de requisitos, pois são documentos gráficos de alto nível que representam os todas as possíveis interações e os atores envolvidos e que podem incluir informações que descrevem essa interação As principais funcionalidades do aplicativo MAPES estão ilustradas na Figura 37 por meio do diagrama de casos de uso e a descrição textual detalhada destes casos de uso está apresentada no Apêndice 1.

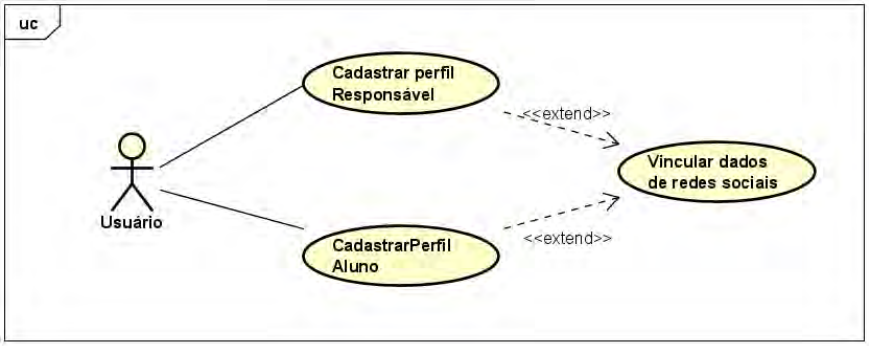
Figura 37 – Diagrama de Casos de Uso do aplicativo MAPES



Fonte: próprio autor

O primeiro passo para poder utilizar o MAPES, é cadastrar o perfil do usuário (Aluno/Responsável) com as informações de login, e-mail, senha, nome completo e, opcionalmente, carregar uma foto para o perfil ou vincular esses dados aos de uma rede social, como Google ou Facebook. A Figura 38 ilustra esse cenário.

Figura 38 – Cadastro de perfis Responsável e Aluno

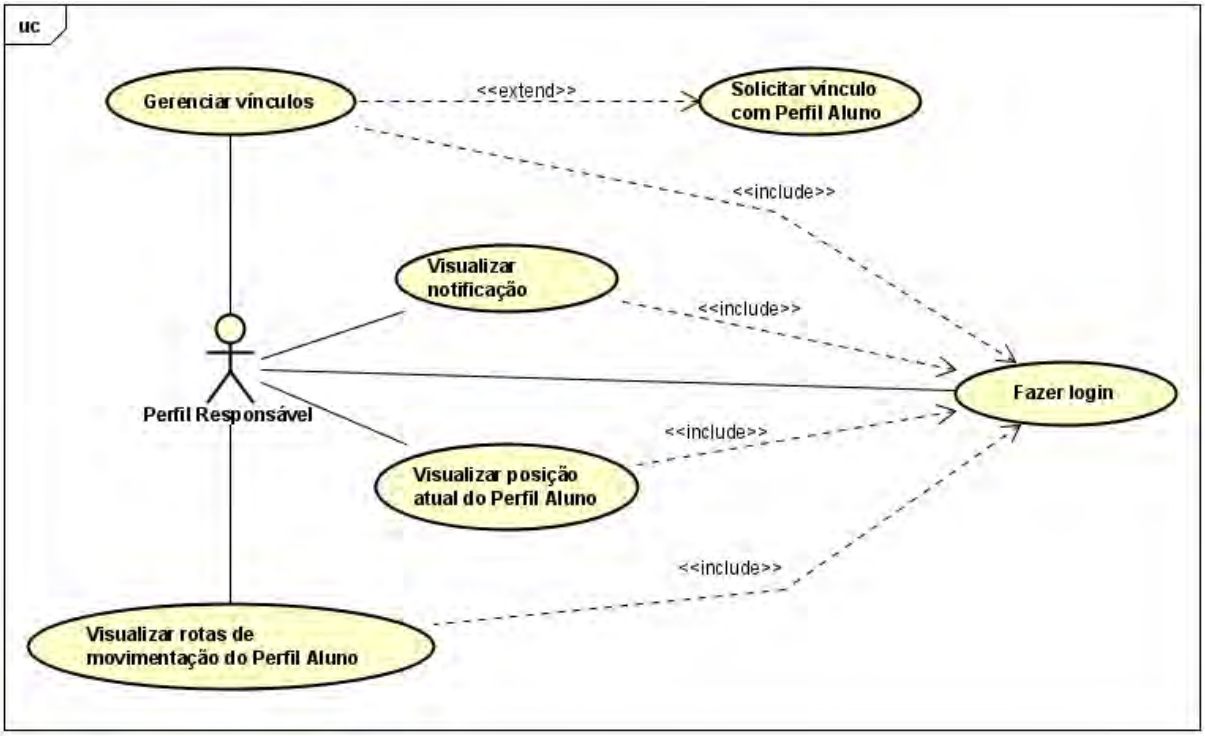


Fonte: próprio autor

Após essa etapa, o usuário deve se autenticar com seu e-mail e senha cadastrados para visualizar as funcionalidades específicas para o seu perfil. É permitido que os usuários alterem as informações dos seus perfis, com exceção do e-mail.

Com o perfil Responsável, o usuário poderá visualizar, na tela inicial, a posição atual do aluno e as últimas notificações de entrada e saída. É possível ainda, consultar as rotas e outras notificações de movimentação dentro da área sobre o Perfil Aluno com o qual mantém vínculo. Para tanto, é necessário que o Responsável adicione o Aluno para habilitar essas funcionalidades. Ao enviar o pedido de vínculo com um perfil Aluno, o usuário responsável deve criar uma senha para ser validada pelo aluno para aceitar o vínculo solicitado. As funcionalidades disponíveis para o perfil Responsável estão representadas na Figura 39.

Figura 39 – Funcionalidades do Perfil Responsável

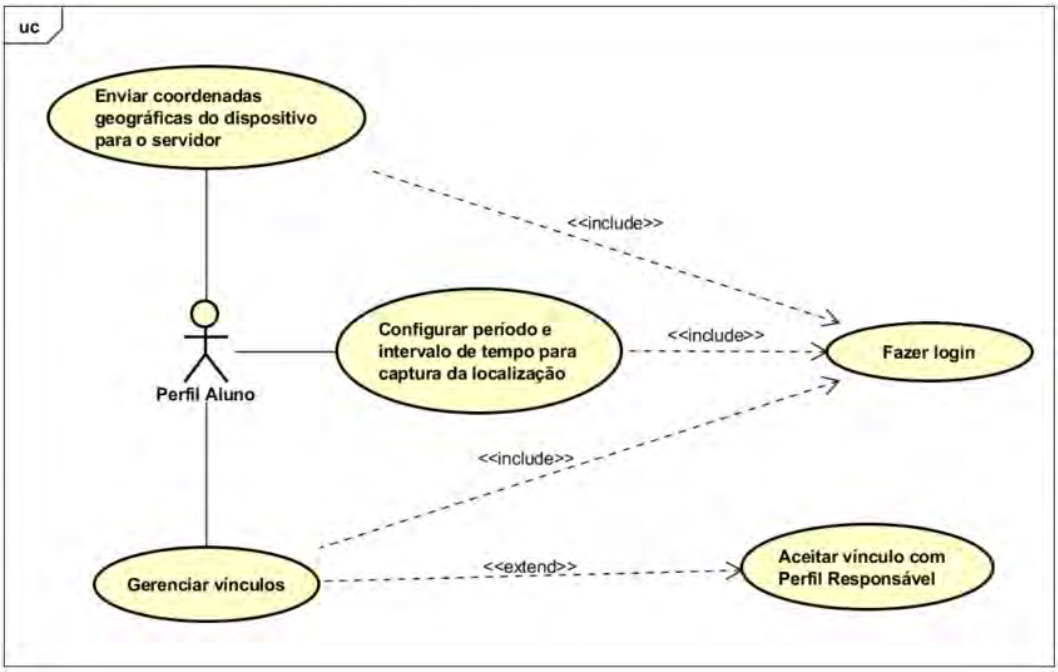


Fonte: próprio autor

Para o perfil Aluno, o MAPES disporá de funcionalidades para visualizar a sua posição atual capturada por seu dispositivo e configurar os recursos de localização, onde poderá ajustar a frequência com que o aplicativo captura as informações de latitude e longitude. Utilizando esse perfil, o app enviará as informações de horário de entrada e saída da área georreferenciada aos módulos do Firebase que enviará, por

sua vez, ao dispositivo que está logado com o perfil Responsável com quem mantem vínculo. Esse cenário é representado na Figura 40.

Figura 40 – Funcionalidades do Perfil Aluno

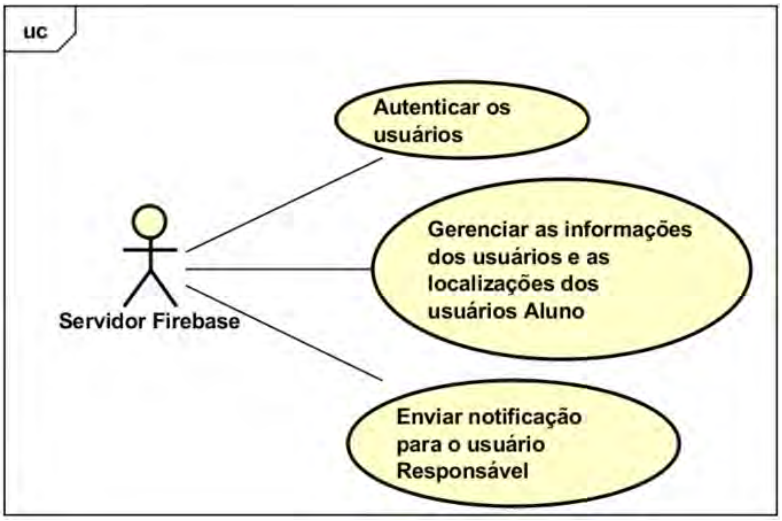


Fonte: próprio autor

Plataforma Firebase

Esta seção detalhará a aplicação que executará na plataforma Firebase e que dará suporte ao aplicativo MAPES, tendo seus casos de uso representados na Figura 41.

Figura 41 – Diagrama de Casos de Uso da aplicação web



Fonte: próprio autor

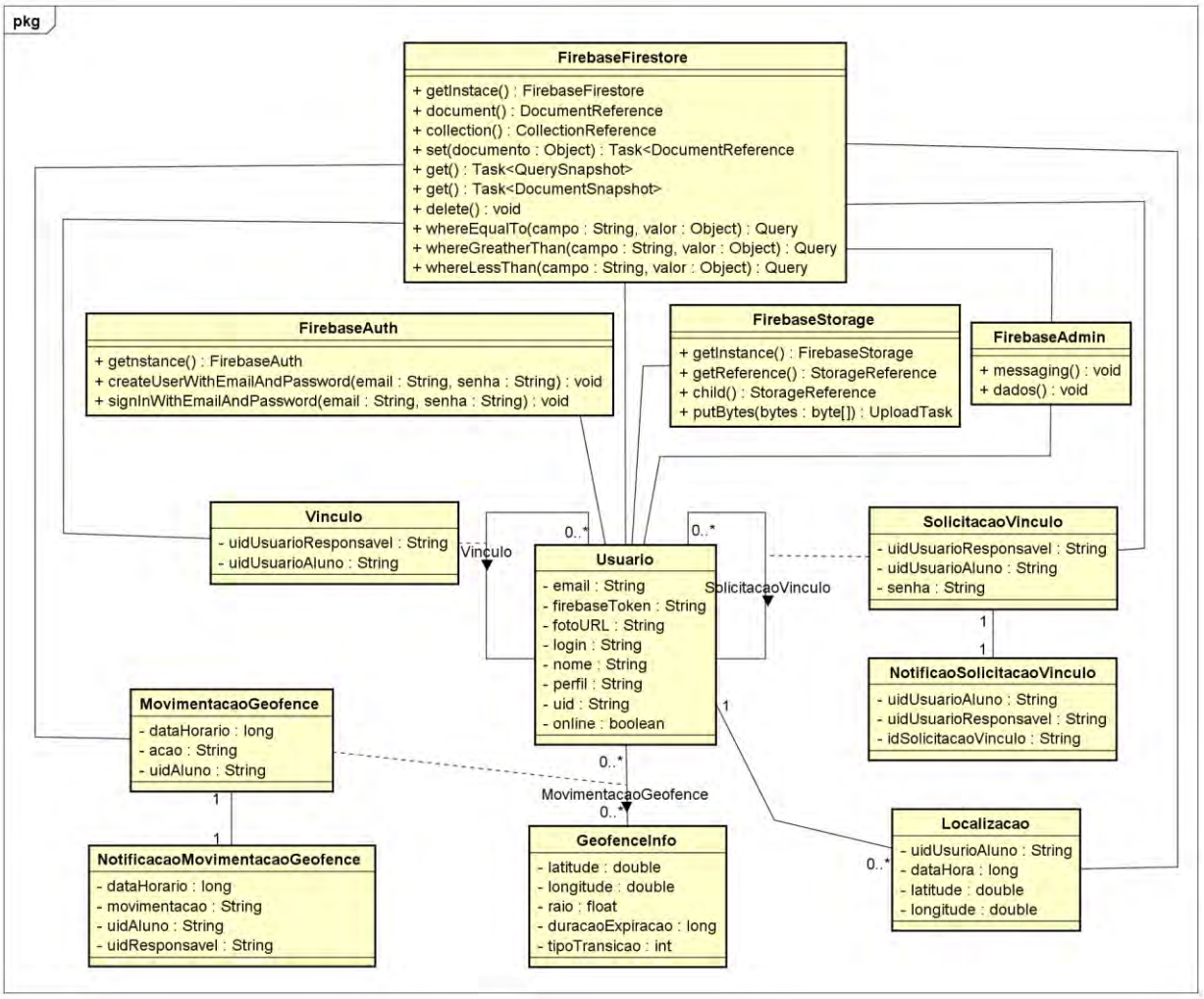
A plataforma Firebase ficará responsável intermediando as ações do aplicativo MAPES baseado nos perfis dos usuários. Quando o responsável ou o aluno fizerem o cadastro ou alterarem seus perfis, esse efetivará essas ações no banco de dados. Através do módulo do *Cloud Firestore,* o Firebase recebe as coordenadas geográficas atuais do dispositivo que está logado com o Perfil Aluno e as salva momentaneamente no banco de dados e as sincroniza enviando-as ao dispositivo que está logado com o Perfil Responsável, para que seu responsável possa visualizar a posição em tempo real do aluno.

Quando o dispositivo logado com o Perfil Aluno entrar ou sair da área georreferenciada, o módulo *Cloud Functions* detecta que novas informações de entrada ou saída foram adicionadas e repassa esses dados por meio de *push notification* ao dispositivo que está logado com o Perfil Responsável.

5.2.1.2 Diagrama de Classes

O diagrama de classes é utilizado para representar de forma estática o conjunto de objetos presentes em uma aplicação contendo seus atributos e seus métodos que serão aplicados para efetuar a manipulação, os relacionamentos e as colaborações que ocorrem entre as classes definidas (PRESSMAN, 2011). A Figura 42 mostra o Diagrama de classes do aplicativo representando as classes e seus relacionamentos.

Figura 42 – Diagrama de Classes do aplicativo MAPES

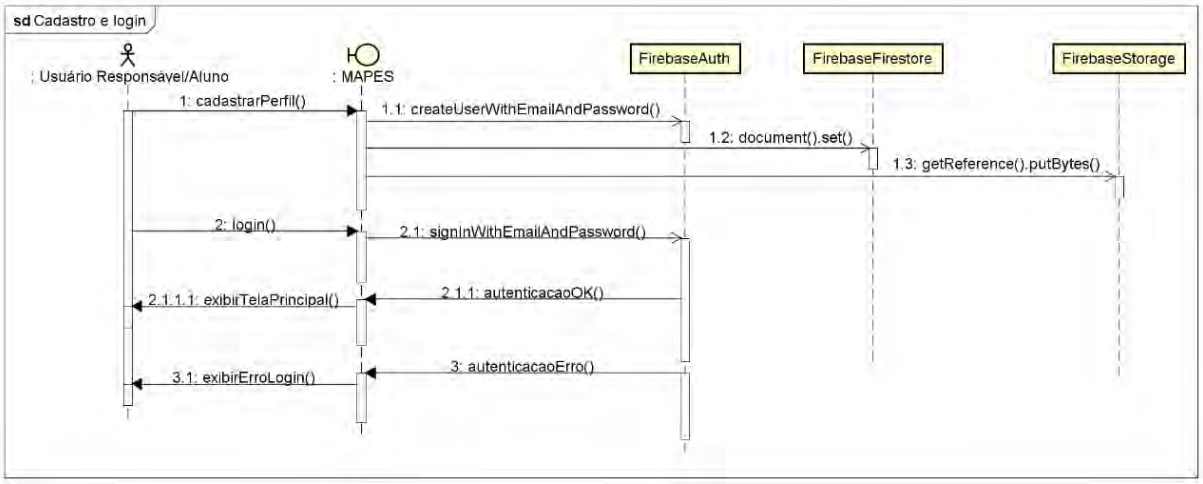


Fonte: próprio autor

5.2.1.3 Diagrama de Sequência

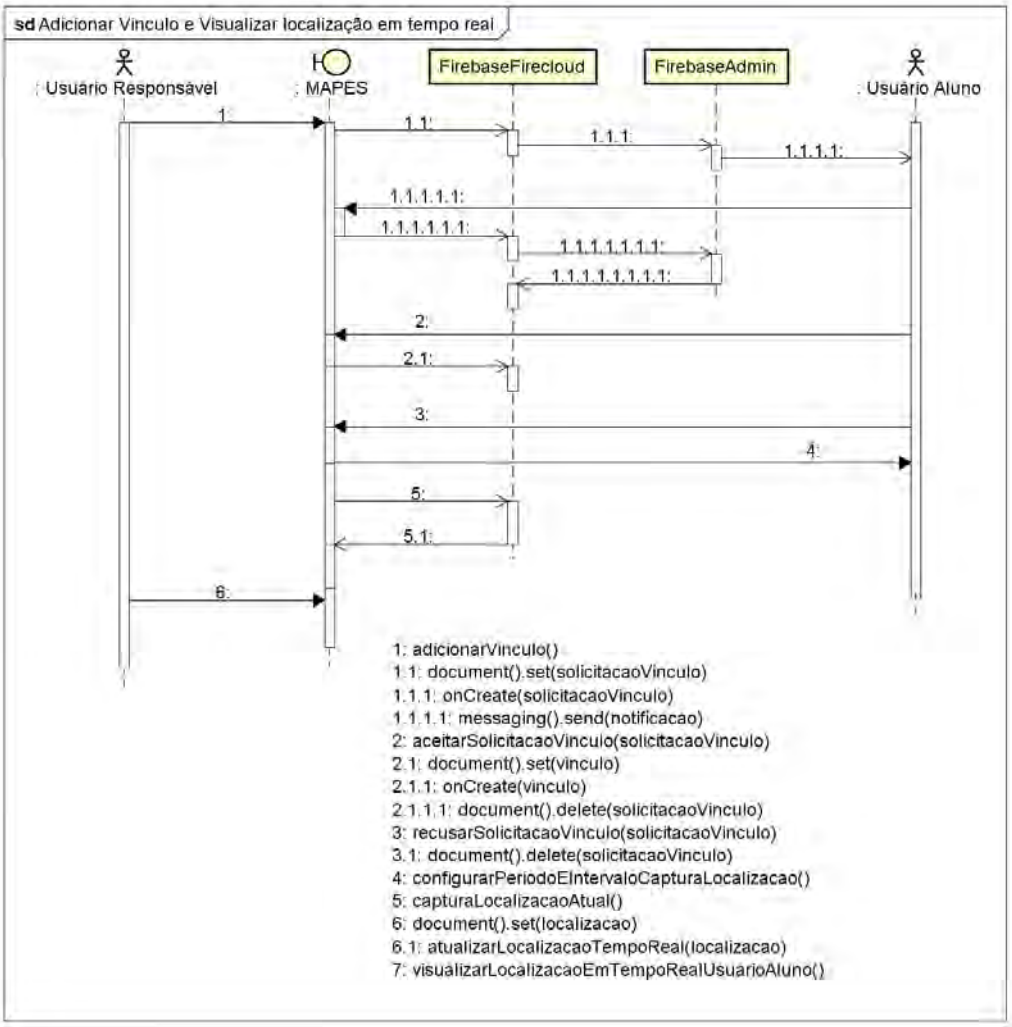
Esse diagrama é utilizado para representar comportamentos, indicando como os objetos/atores se comunicam em função do tempo através dos eventos iniciados pelos métodos. Além disso, o diagrama de sequência também detalha a sequência das interações envolvidas em um caso de uso em particular de alto nível (SOMMERVILLE, 2011; PRESSMAN, 2011). As Figuras 43, 44 e 45 ilustram os diagramas para as principais funcionalidades do aplicativo baseado nos casos de uso.

Figura 43 – Diagramas de Sequência para cadastro e login



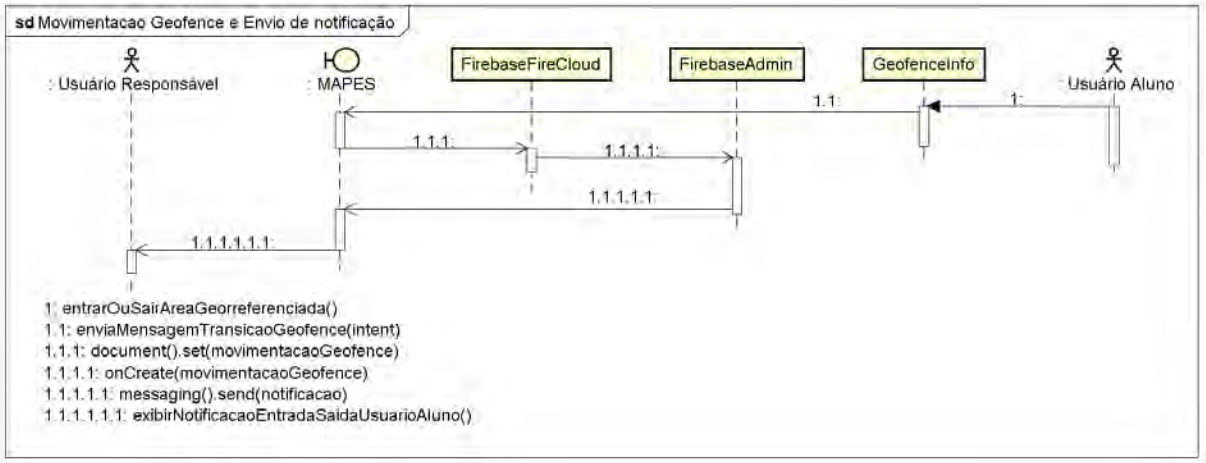
Fonte: próprio autor

Figura 44 – Diagrama de Sequência para adicionar vínculo e visualizar a localização em tempo real



Fonte: próprio autor

Figura 45 – Diagrama de Sequência para detectar movimentação na área georreferenciar e envio de notificações



Fonte: próprio autor

5.3. DETALHAMENTOS DA APLICAÇÃO

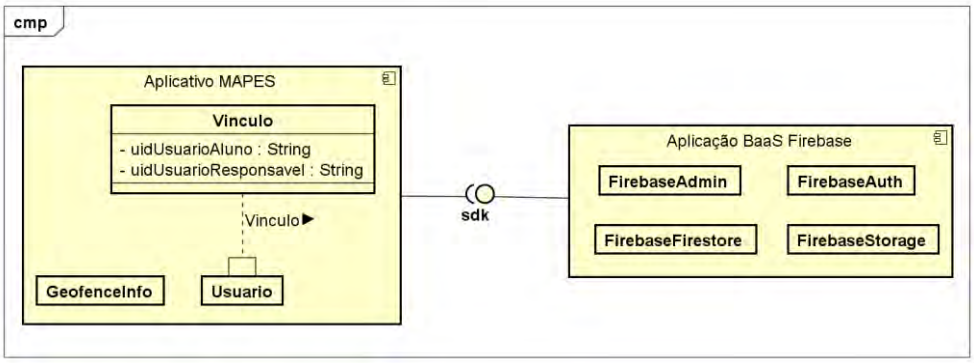
Para funcionamento adequado do aplicativo, foi realizado o desenvolvimento de 2 módulos (componentes) principais:

- **Aplicativo mobile:** corresponde ao aplicativo instalado no smartphone que irá disponibilizar as funcionalidades específicas para cada perfil. Esse componente gerencia os vínculos entre os usuários a geofence, responsável por responder aos eventos de transição na área georreferenciada;

- **Aplicação BaaS no Firebase:** ficará responsável por salvar as coordenadas passadas pelo aplicativo mobile e enviar as *push notifications* de entrada e saída do aluno. Essa aplicação executa as funções de backend sendo as de autenticação, banco de dados, repositório de arquivos e ambiente para desenvolvimento de funções orientadas a eventos.

Para realizar a integração dos produtos do Firebase ao app MAPES, é importante adicionar os SDKs que utilizam as bibliotecas necessárias para que cada módulo funcione adequadamente. Essa etapa, será demonstrada a seguir no item 5.3.1. A Figura 46 representa o diagrama da interação entre esses componentes.

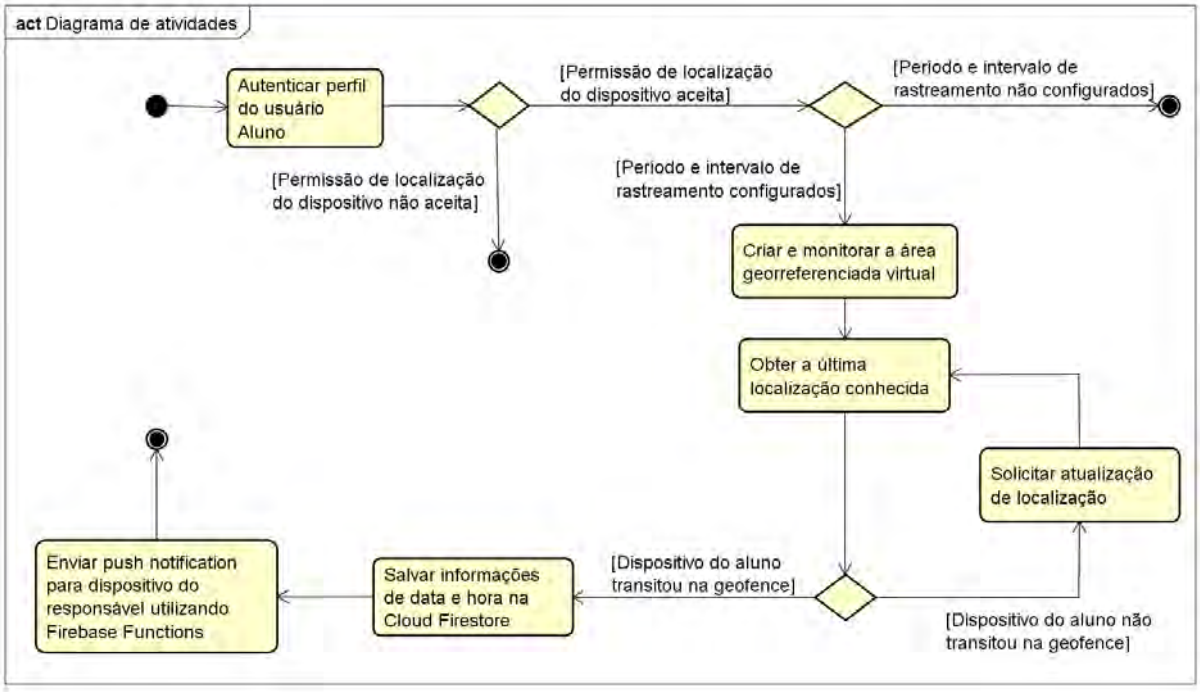
Figura 46 – Diagrama de componentes da Arquitetura MAPES



Fonte: próprio autor

A visão geral dos passos realizados para monitorar do dispositivo do aluno executados pelo algoritmo da aplicação mobile e a aplicação BaaS Firebase que utiliza os recursos *Authentication, Storage, Cloud Firestore* e *Cloud Functions* é representado no diagrama de atividades da Figura 47.

Figura 47 – Diagrama de Atividades



Fonte: próprio autor

O protótipo do aplicativo MAPES foi desenvolvido na linguagem Java/Android utilizando a IDE Android Studio 3.6 que é o ambiente de desenvolvimento oficial para desenvolvimento de aplicativos Android que é mantido pela Google e foi testado em dois dispositivos emulados *(AVD – Android Virtual Device)* integrados a IDE. As versões dos sistemas operacionais instalados nos dispositivos foram a Nougat (7.1) e

a Oreo (8.0). Já a aplicação Firebase foi instalada e configurada conforme mostrado no item 4.3.

A seguir, será descrito o detalhamento da implementação das principais funcionalidades do aplicativo.

5.3.1. Adicionar SDKs do Firebase ao aplicativo

Para adicionar os SDKs dos módulos do Firebase, é necessário adicionar as dependências no arquivo Graddle (app/build.gradle) do módulo do aplicativo. O código para adicionar as dependências é demonstrado na Figura 48.

Figura 48 – Código para adicionar as dependências dos produtos Firebase

```
01  dependencies {
02      implementation fileTree(dir: 'libs', include: ['*.jar'])
03
04      implementation 'com.google.firebase:firebase-analytics:17.2.0'
05      implementation 'com.google.firebase:firebase-core:17.2.0'
06      implementation 'com.google.firebase:firebase-auth:19.1.0'
07      implementation 'com.google.firebase:firebase-database:19.1.0'
08      implementation 'com.google.firebase:firebase-storage:19.1.0'
09      implementation 'com.google.firebase:firebase-firestore:20.0.0'
10      implementation 'com.google.firebase:firebase-messaging:20.0.0'
11  }
```

Fonte: próprio autor

Na linha 10, é adicionado a dependência necessária para utilizar os recursos do FCM. Nas linhas anteriores, são adicionadas as outras dependências necessárias para utilizar os recursos de *Analytics*, *Authentication*, *Storage* e *Cloud Firestore*.

5.3.2. Cadastrar o perfil do usuário

Para cadastrar o perfil do usuário foi utilizado os recursos *Authentication* e *Storage* do Firebase, que implementam essa função e a de salvar as imagens dos perfis cadastrados, respectivamente. O código implementado para o cadastro do perfil do usuário está representado na Figura 49.

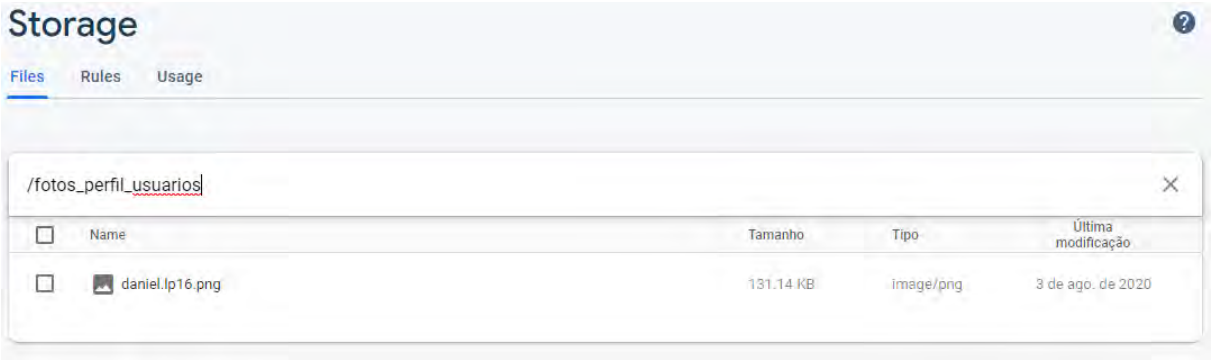
Figura 49 – Código para cadastrar o usuário

```
01  firebaseAuth.createUserWithEmailAndPassword(usuarioQuerSeraCadastrado.getEmail(), senha)
02          .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
03              @Override
04              public void onSuccess(AuthResult authResult) {
05                  firebaseUser = authResult.getUser();
06                  if (firebaseUser != null) {
07                      final StorageReference imagemPerfilUsuarioStrageReference = usuarioDAO
08                              .getFotosPerfilUsuarioStorageReference(
09                                      usuarioQuerSeraCadastrado.getLogin());
10                      imagemPerfilUsuarioStrageReference
11                              .putBytes(cadastrarImagemFragment.getFotoPerfilDoImageView())
12                              .addOnSuccessListener(
13                                      new OnSuccessListener<UploadTask.TaskSnapshot>() {
14                                  @Override
15                                  public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
16                                      imagemPerfilUsuarioStrageReference.getDownloadUrl()
17                                              .addOnSuccessListener(new OnSuccessListener<Uri>() {
18                                                  @Override
19                                                  public void onSuccess(Uri uri) {
20                                                      usuarioQuerSeraCadastrado
21                                                              .setFotoURL(uri.toString());
22                                                      usuarioQuerSeraCadastrado
23                                                              .setUid(firebaseUser.getUid());
24                                                      usuarioDAO.cadastrarUsuario(usuarioQuerSeraCadastrado,
25                                                              firebaseUser);
26                                                      Toast.makeText(getApplicationContext(),
27                                                              text: "Usuário cadastrado com sucesso!",
28                                                              Toast.LENGTH_LONG).show();
```

Fonte: próprio autor

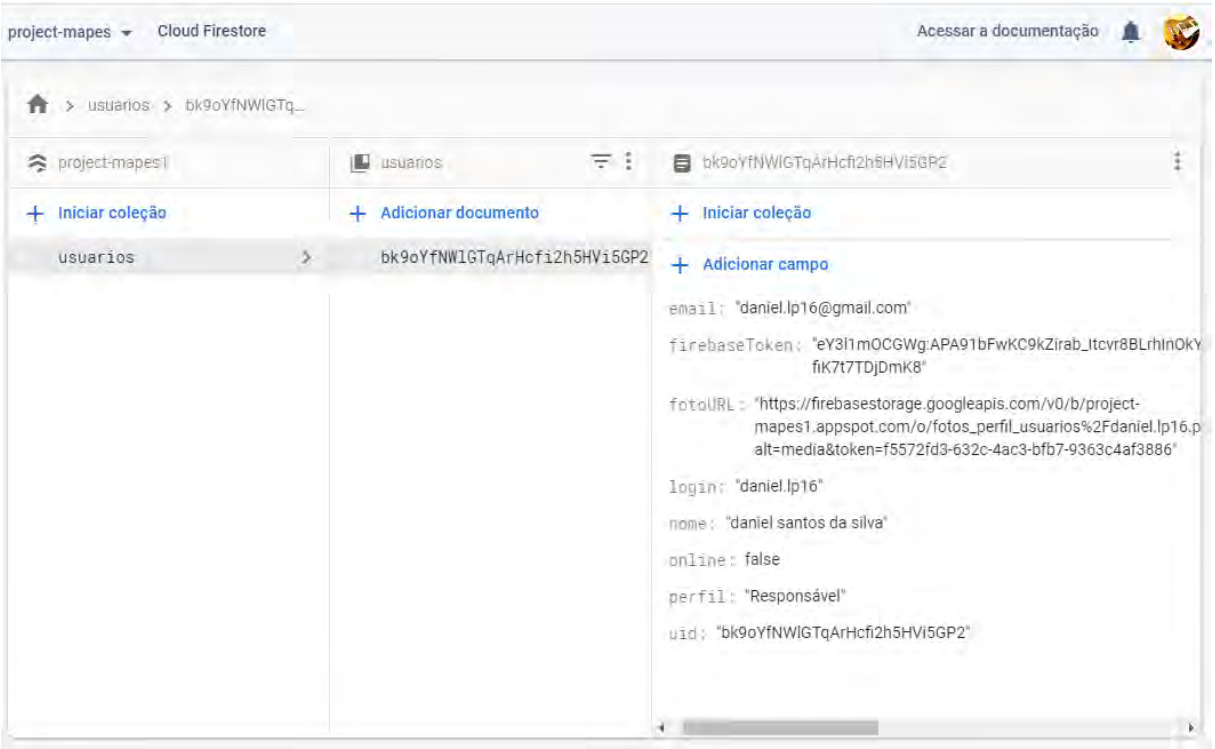
Na linha 1, o método *createUserWithEmailAndPassword* do objeto instanciado de *FirebaseAuth* cria um usuário no Firebase com dois parâmetros passados sendo email e senha, respectivamente. Na linha 5 dentro do *listener* de *callback onSucessListener*, é recuperado o usuário criado e autenticado (*FirebaseUser*) através da instância de *AuthResult*. Da linha 7 a 11, é criado uma instância de *StorageReference* que contém o caminho onde será armazenada a imagem de perfil do usuário (utilizando o método *getFotosPerfilUsuarioStorageReference*) e a própria imagem (utilizando o método *putBytes*). Nas linhas 20 e 22, são passados para a instância do usuário o caminho da foto do perfil armazenada em *Storage* e o *User Id,* que é o identificador único de autenticação para o Firebase. Por fim, na linha 24, o perfil do usuário é cadastrado no banco de dados do *Cloud Firestore.* As Figuras 50 e 51 representam as informações armazenadas no *Cloud Storage* e no *Cloud Firestore*.

Figura 50 – Informações armazenadas no *Cloud Storage*



Fonte: próprio autor

Figura 51 – Informações do perfil usuário cadastrado no *Cloud Firestore*



Fonte: próprio autor

5.3.3. Autenticar o perfil do usuário

Para autenticar o perfil do usuário foi utilizado o recurso *Authentication* do Firebase, que implementa as funções de gerenciamento dos perfis cadastrados. A Figura 52 apresenta o código para o autenticar o perfil do usuário.

Figura 52 – Código para autenticar o perfil do usuário

```
01  firebaseAuth.signInWithEmailAndPassword(email, senha)
02          .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
03              @Override
04              public void onComplete(@NonNull Task<AuthResult> task) {
05                  if (task.isSuccessful()) {
06                      usuarioDAO.getUsuarioCollectionReference() CollectionReference
07                              .whereEqualTo( field: "email", email) Query
08                              .limit(1)
09                              .get() Task<QuerySnapshot>
10                              .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
11                                  @Override
12                                  public void onComplete(@NonNull Task<QuerySnapshot> task) {
13                                      if (!task.isSuccessful()) {
14                                          Log.e( tag: "Erro", task.getException().getMessage());
15                                          return;
16                                      }
17
18                                      Usuario usuario = task.getResult().getDocuments().get(0)
19                                              .toObject(Usuario.class);
20
21                                      atualizarTokenUsuario(usuario);
22                                      sharedPreferencesDAO.atualizarUsuarioLogin(usuario, senha);
23                                      iniciarTelaPrincipal(usuario);
24                                  }
25                              });
```

Fonte: próprio autor

Na linha 1, o método *signInUserWithEmailAndPassword* do objeto instanciado de *FirebaseAuth* faz o login do usuário no Firebase com dois parâmetros passados sendo email e senha, respectivamente. Na linha 6, é realizada a pesquisa pelo usuário no *Cloud Firestore* usando o parâmetro único “email” (dentro do método *whereEqualsTo*). Na linha 18, dentro do *listener* de *callback onCompleteListener*, é recuperada a instância do usuário devidamente autenticado.

5.3.4. Solicitar permissões de localização do dispositivo do aluno em primeiro e em segundo plano

Devido a política de proteção e privacidade do usuário, os apps precisam solicitar explicitamente permissões de localização. As permissões solicitadas e a forma como são feitas dependem dos requisitos de localização específico para o uso do aplicativo. As permissões de localização processam as categorias de acesso à localização como: localização em primeiro plano e a localização em segundo plano (ANDROID DEVELOPERS, 2019).

A localização em primeiro plano é utilizada quando uma atividade pertencente ao app esteja visível ou estiver executando um serviço em primeiro plano que utiliza um recurso que compartilha ou receba informações de localização apenas uma vez ou por um período definido. Para tanto, é recomendável declarar um tipo de serviço em primeiro plano de “location” (ANDROID DEVELOPERS, 2019), conforme mostrado a partir da linha 6 da Figura 53.

Figura 53 – Código para solicitar as permissões de localização em primeiro e em segundo plano

```
01      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
02      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
03      <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
04      <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
05
06      <service
07          android:name=".RastreamentoAlunoIntentService"
08          android:enabled="true"
09          android:exported="true"
10          android:foregroundServiceType="location" />
11
```

Fonte: próprio autor

Os níveis de precisão da localização em primeiro plano são solicitados nas linhas 1 e 2, que definem precisão de localização de um quarteirão que utiliza a conexão de dados 3G/4G e/ou wi-fi (ACCESS_COARSE_LOCATION); e de poucos metros que utiliza GPS (ACCESS_FINE_LOCATION) (ANDROID DEVELOPERS, 2019).

Já a localização em segundo plano é utilizada em qualquer situação diferente das descritas para a localização em primeiro plano ou se um recurso do app compartilha constantemente a localização com outros usuários ou usa API *Fences* (ANDROID DEVELOPERS, 2019), que é a API utilizada neste aplicativo desenvolvido. Na linha 3 da figura anterior, encontra-se a permissão declarada para solicitar acesso à localização em segundo plano no momento da execução

5.3.5. Configurar o período e o intervalo de rastreamento

Para configurar o horário inicial e o de término e o intervalo de rastreamento, foi utilizado as instâncias de *TimerPickerDialog* e de *SeekbarPreference,* respectivamente. Essas instâncias obtém os valores para serem salvos no formato "chave/valor" em uma instância de *SharedPreferences* que é uma forma de armazenamento de configurações simples e que dispensa a necessidade de defini-las toda vez que executar o aplicativo. O código para implementar essas funções é apresentado na Figura 54.

Figura 54 – Código para configurar o período e o intervalo de rastreamento

```
01  private void exibirTimePickerDialog(final Preference preference, final String chave){
02      TimePickerDialog.OnTimeSetListener tratador =
03              new TimePickerDialog.OnTimeSetListener() {
04                  @Override
05                  public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
06                      Calendar horario = Calendar.getInstance();
07                      horario.set(Calendar.HOUR_OF_DAY, hourOfDay);
08                      horario.set(Calendar.MINUTE, minute);
09                      horario.set(Calendar.SECOND, 0);
10
11                      String horarioString = funcoes
12                              .converterHorarioMilisegundosEmString(horario.getTimeInMillis());
13                      preference.setSummary(horarioString
14                              .substring(0,horarioString.length() - 3));
15
16                      salvarPreferencias(chave, horario.getTimeInMillis());
17                  }
18              };
19      Calendar horario = Calendar.getInstance();
20      horario.setTimeInMillis(
21              funcoes.converterHorarioStringEmMilisegundos(preference
22                      .getSummary().toString()));
23      horario.setTimeZone(TimeZone.getTimeZone("GMT"));
24      TimePickerDialog dialog = new TimePickerDialog(
25              getContext(),
26              tratador,
27              horario.get(Calendar.HOUR_OF_DAY),
28              horario.get(Calendar.MINUTE), is24HourView: true);
29      dialog.show();
30  }
31
32  private void salvarPreferencias(String chave, long valor){
33      SharedPreferences.Editor editor = sharedPreferences.edit();
36      editor.putLong(chave, valor);
37      editor.commit();
38
39      TelaPrincipalUsuarioAlunoActivity telaPrincipalUsuarioAlunoActivity =
40              (TelaPrincipalUsuarioAlunoActivity) getActivity();
41      telaPrincipalUsuarioAlunoActivity.configurarAlarm(chave, valor);
42  }
```

Fonte: próprio autor

Na linha 3, é criado um listener *OnTimeSetListener* para guardar o horário definido em *SharedPreferences* através do método *salvarPreferencias* com os parâmetros de chave e o valor, exibido na linha 16. Da linha 24 a 29, é criada a instância de *TimerPickerDialog* que recebe os parâmetros do listener, horas e minutos que serão definidas ao exibir o Dialog e o formato de visualização (12h ou 24h). Na linha 33, é recuperado a instância de *SharedPreferences* para salvar os valores no arquivo de configuração.

5.3.6. Criar e monitorar área georreferenciada virtual

Para solicitar o monitoramento da área georrefrenciada virtual (conhecida com *geofence*) é necessário solicitar as permissões ACCES_FINE_LOCATION e ACCESS_BACKGROUND_LOCATION, e esse processo já foi demonstrado na seção anterior. Em seguida foi criado um *BroadcastReceiver* para detectará as transições nessa *geofence,* e o trecho do código que corresponde a essa é apresentado na Figura 55.

Figura 55 – Código para criar o *BroadcastReceiver* que detecta as transições na *geofence*

```
01      <receiver
02          android:name=".suporte.GeofenceBroadcastReceiver"
03          android:enabled="true"
04          android:exported="true" />
```

Fonte: próprio autor

Para criar e monitorar a *geofence*, foi utilizado a instância de *GeofencingClient* da API *Location*. A Figura 56 apresenta o código implementado para executar essas tarefas.

Figura 56 – Código para criar e monitorar a *geofence*

```
01  private PendingIntent getGeofencePendingIntent(){
02      if(geofencePendingIntent != null){
03          return geofencePendingIntent;
04      }
05      Intent intent = new Intent(getApplicationContext(), GeofenceBroadcastReceiver.class);
06      geofencePendingIntent = PendingIntent.getBroadcast(getBaseContext(), requestCode: 0,
07              intent, PendingIntent.FLAG_UPDATE_CURRENT);
08      return geofencePendingIntent;
09  }
10
```

```
11      public static GeoFenceInfo getGeofenceInfo() {
12          //LOCALIZAÇÃO DO IFAM CMC
13          return new GeoFenceInfo( id: "1",
14                  Constantes.KEY_LATITUDE_IFAM,
15                  Constantes.KEY_LONGITUDE_IFAM,
16                  Constantes.KEY_RAIO_FENCE_IFAM, Geofence.NEVER_EXPIRE,
17                   tipoTransicao: Geofence.GEOFENCE_TRANSITION_ENTER | Geofence.GEOFENCE_TRANSITION_EXIT);
18      }
19
20      private GeofencingRequest getGeofencingRequest(GeoFenceInfo geoFenceInfo) {
21          GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
22          builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
23          builder.addGeofence(geoFenceInfo.getGeofence());
24          return builder.build();
25      }
26
27      public void criarGeofence() {
28          final PendingIntent pendingIntent = getGeofencePendingIntent();
29          geoFenceInfo = getGeofenceInfo();
30          final GeofencingRequest geofencingRequest = getGeofencingRequest(geoFenceInfo);
31          geoFenceDB = new GeoFenceDB(getBaseContext());
32          geofencingClient = LocationServices.getGeofencingClient( context: this);
33          geofencingClient.addGeofences(geofencingRequest, pendingIntent)
36                  .addOnSuccessListener(new OnSuccessListener<Void>() {
37                      @Override
38                      public void onSuccess(Void aVoid) {
39                          geoFenceInfo = null;
40                      }
41                  });
42      }
```

Fonte: próprio autor

Da linha 1 a 9, o método *getGeofencingPendingIntent* define uma *PendingIntent* para iniciar o *BroadcastReceiver* que irá gerenciar a transição de fronteira virtual. Entre as linhas 11 e 18, é definido o método *getGeofenceInfo* que recupera uma instância de *GeofenceInfo* que contém as informações de latitude, longitude, raio, tempo de expiração e o tipo de transição para criar a *geofence*. A transição GEOFENCE_TRANSITION_ENTER é acionada quando o dispositivo entrada na área virtual e a transição GEOFENCE_TRANSITION_EXIT é acionada quando o dispositivo sai, sendo essas utilizadas na linha 17. Da linha 20 a 25, o método *getGeofencingRequest* cria uma instância de *GeofencigRequest* a partir de *GeofencingRequestBuilder* para solicitar a criação da *geofence*. A definição de INITIAL_TRIGGER_ENTER informa aos Serviços de localização que GEOFENCE_TRANSITION_ENTER precisa ser acionado antes se o dispositivo já estiver dentro da área georreferenciada, que é passado como parâmetro para o método *setInitialTrigger* na linha 22. É possível ainda utilizar INITIAL_TRIGGER_DWELL, que aciona eventos somente quando o dispositivo

permanece dentro da área por um tempo determinado. Por fim, entre as linhas 27 e 42, o método *criarGeofence* instancia um *GeofencingClient* e adiciona a área que deve ser monitorada passando como parâmetros a instância de *GeofencingRequest* e a *PendingIntent* que inicia o *BroadcastReceiver* para receber os eventos de transição.

5.3.7. Ver a última localização conhecida e solicitar novas atualizações

Para recuperar a última localização conhecida do dispositivo, deve-se usar a API *Fused Location Provider*, que é uma das APIs de localização do Google Play Services. Ela gerencia a tecnologia de localização fornecendo uma implementação simples com alta precisão ou baixo consumo de energia, otimizando o uso da bateria do dispositivo (ANDROID DEVELOPERS, 2019).

Foi utilizado uma instância do cliente do provedor de localização combinada *FusedLocationProviderClient.* Depois dessa etapa, o método *getLastLocation* foi utilizado para recuperar a última localização conhecida do dispositivo do aluno. A precisão de localização utilizada é determinada pela configuração de permissão que está no arquivo de manifesto do app, conforme demonstrado na seção 5.2.3. O código para implementar essa etapa é apresentado na Figura 57.

Figura 57 – Código para recuperar a última localização conhecida do dispositivo

```
01  fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient( context: this);
02  fusedLocationProviderClientTempoReal = LocationServices.getFusedLocationProviderClient( context: this);
03
04  getUltimaLocalizacaoConhecida();
05
06  private void getUltimaLocalizacaoConhecida() {
07      try {
08          fusedLocationProviderClient.getLastLocation()
09                  .addOnCompleteListener(new OnCompleteListener<Location>() {
10              @Override
11              public void onComplete(@NonNull Task<Location> task) {
12                  if (task.isSuccessful() && task.getResult() != null) {
13                      location = task.getResult();
14                  } else {
15                      Log.e( tag: "ERRO",  msg: "Erro ao recuperar localização!");
16                  }
17              }
18          });
19      } catch (SecurityException e) {
20          Log.e( tag: "ERRO",  msg: "Sem permissão para acesso à localização " + e);
21      }
22  }
```

Fonte: próprio autor

Na linha 1, foi criado uma instância de *FusedLocationProviderClient* através do método *getFusedLocationProviderClient*. A última localização conhecida é solicitada na linha 8 através do método *getLastLocation*. A localização é recuperada na linha 13 dentro do *call-back OnCompleteListener.*

Para solicitar atualização de localização do dispositivo, o dispositivo precisar ativar as configurações necessárias de sistema, como busca por Wi-fi ou GPS. O app especifica o nível adequado de precisão, consumo de energia e o intervalo de atualização desejado e o dispositivo faz as configurações automaticamente. Essas configurações são definidas pelo objeto de dados *LocationRequest* (ANDROID DEVELOPERS, 2019).

Para criar solicitações de atualização de localização ao provedor de localização combinada, foi utilizada uma instância de *LocationRequest* que utiliza parâmetros definindo o nível de precisão, intervalo de atualização, intervalo de atualização mais rápido e a prioridade. O método *setInterval* define a velocidade em milissegundos em que o app prefere receber as atualizações de localização que podem ter uma variação mais rápida ou mais lenta que o valor definido. O método *setFastestInterval* define a velocidade mais rápida em milissegundos em que o app consegue receber as atualizações com valores mais precisos. O método *setPriority* define a relação entre prioridade da solicitação, nível de precisão e consumo de bateria:

- PRIORITY_BALANCED_POWER_ACCURACY: utiliza Wi-fi e a rede móvel para definir a localização com precisão de aproximadamente de 100 metros. Essa configuração mantém o uso balanceado da bateria com um nível aproximado de precisão.
- PRIORITY_HIGH_ACCURACY: essa configuração utilizaGPS para determinar a localização de forma mais precisa possível, aumentando o consumo da bateria.
- PRIORITY_LOW_POWER: utiliza a rede móvel para definir a localização com precisão de aproximadamente 10 quilômetros consumindo menos energia do que os anteriores.
- PRIORITY_NO_POWER: essa configuração prioriza o baixíssimo consumo de energia e recebe as atualizações de localização somente quando utilizadas por outros apps (ANDROID DEVELOPERS, 2019).

Além disso, após recuperar a nova localização, o provedor de localização combinada *FusedLocationProviderClient* invoca o método de callback LocationCallback passando-a como parâmetro pelo método *onLocationResult*. A partir desse método é possível recuperar as informações como latitude, longitude, data, hora etc (ANDROID DEVELOPERS, 2019). O código implementado para criar e solicitar as atualizações de localização está representado na Figura 58.

Figura 58 – Código para criar o *LocationRquest*

```
01  @Override
02  public void onCreate() {
03      sharedPreferencesDAO = new SharedPreferencesDAO(getBaseContext());
04      usuario = sharedPreferencesDAO.getUsuarioLogado();
05      notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
06
07      locationCallback = new LocationCallback() {
08          @Override
09          public void onLocationResult(LocationResult locationResult) {
10              super.onLocationResult(locationResult);
11              salvarLocalizacao(locationResult.getLastLocation());
12          }
13      };
14  }
15
16  private void criarLocationRequestParaSalvarRotas() {
17      SharedPreferencesDAO sharedPreferencesDAO = new SharedPreferencesDAO( context: this);
18      INTERVALO_CURTO_ATUALIZACAO_LOCATION_REQUEST_EM_MILISSEGUNDOS = sharedPreferencesDAO
19              .getIntervaloLocationRequest();
20      INTERVALO_ATUALIZACAO_LOCATION_REQUEST_EM_MILISSEGUNDOS =
21              (long) (INTERVALO_CURTO_ATUALIZACAO_LOCATION_REQUEST_EM_MILISSEGUNDOS * 1.1);
22
23      locationRequest = new LocationRequest();
24      locationRequest.setInterval(INTERVALO_ATUALIZACAO_LOCATION_REQUEST_EM_MILISSEGUNDOS);
25      locationRequest.setFastestInterval(INTERVALO_CURTO_ATUALIZACAO_LOCATION_REQUEST_EM_MILISSEGUNDOS);
26      locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
27
28      fusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback,
29              Looper.myLooper());
30  }
```

Fonte: próprio autor

Da linha 7 a 13, a instância de *LocationCallback* é definida e na linha 11 é recuperada a nova localização e salva na CloudFirestore através do método *salvarLocalizacao*. Da linha 23 a linha 26, é instanciado um *LocationRequest* onde são passados todos os parâmetros necessários. E então, na linha 28, as instâncias de *LocationRequest* e de *LocationCalback* são passados como parâmetros para o método *requestLocationUpdates* que solicitará as atualizações de localização do dispositivo.

5.3.8. Tratar os eventos de transição na geofence e salvar informações no Cloud Firestore

Para tratar os eventos de transição na área georreferenciada, foi utilizado a instância de *BraodcastReceiver*. A Figura 59 apresenta o código para executar essas tarefas.

Figura 59 – Código para tratar os eventos de transição pelo *BroadcastReceiver*

```
@Override
public void onReceive(final Context context, Intent intent) {
    sharedPreferencesDAO = new SharedPreferencesDAO(context);
    final Usuario usuarioAluno = sharedPreferencesDAO.getUsuarioLogado();
    final GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);

    if(geofencingEvent.hasError()){
        int errorCode = geofencingEvent.getErrorCode();
        Toast.makeText(context,  text: "Erro mo serviço de localização:  " + errorCode,
                Toast.LENGTH_SHORT).show();
    } else {
        movimentacaoGeofenceDAO = new MovimentacaoGeofenceDAO();
        usuarioDAO = new UsuarioDAO();
        vinculoDAO = new VinculoDAO();

        vinculoDAO.getVinculosCollectionReference() CollectionReference
                .whereEqualTo( field: "uidUsuarioAluno", usuarioAluno.getUid()) Query
                .get() Task<QuerySnapshot>
                .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
            @Override
            public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                long dataHora = System.currentTimeMillis();
                int transicao = geofencingEvent.getGeofenceTransition();
                String movimentacao = "";

                if(transicao == Geofence.GEOFENCE_TRANSITION_ENTER){
                    movimentacao = "entrada";
                } else if(transicao == Geofence.GEOFENCE_TRANSITION_EXIT){
                    movimentacao = "saida";
                }

                for(DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments()){
                    Vinculo vinculo = documentSnapshot.toObject(Vinculo.class);
                    final MovimentacaoGeofence movimentacaoGeofence = new MovimentacaoGeofence(
                            dataHora, movimentacao, vinculo.getUidUsuarioAluno());

                    usuarioDAO.getUsuarioCollectionReference() CollectionReference
                            .document(vinculo.getUidUsuarioResponsavel()) DocumentReference
                            .get() Task<DocumentSnapshot>
                            .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                        @Override
                        public void onSuccess(DocumentSnapshot documentSnapshot) {
                            Usuario usuarioResponsavel = documentSnapshot.toObject(Usuario.class);
                            movimentacaoGeofenceDAO.adicionarMovimentacaoGeofence(movimentacaoGeofence,
                                    usuarioResponsavel);
                        }
                    });
```

Fonte: próprio autor

Na linha 5, é criado uma instância de *GeofencingEvent* que recebe o evento de transição pela *Intent* que ativou o *BroascastReceiver*. Na linha 23, é recuperado o tipo da transição detectado pelo *GeofencingEvent* e em seguida, entre as linhas 26 e 30, é feito a o teste para saber qual foi o evento de transição usado para acionar o *BroadcastReceiver*. Na linha 36, é criada uma instâncida de *MovimentacaoGeofence* que contêm as informações de data, hora, movimentação (entrada/saída), identificador único do perfil do usuário que transitou na área georreferenciada. Então, na linha 46, essas informações e o identificador único do perfil do responsável são salvos na Cloud Firestore através do método *adicionarMovimentacaoGeofence*.

5.3.9. Enviar notificação para o Responsável utilizando o Firebase

O primeiro passo necessário, é adicionar um serviço ao manifesto do aplicativo que estende *FirebaseMessagingService* para processar as notificações no app em primeiro e em segundo plano. O trecho do código implementado para adicionar o serviço é demonstrado na Figura 60.

Figura 60 – Código para adicionar o serviço *FirebaseMessagingService*

```
01  <service
02      android:name=".suporte.NotificacoesFirebaseService"
03      android:exported="false">
04      <intent-filter>
05          <action android:name="com.google.firebase.MESSAGING_EVENT" />
06      </intent-filter>
07  </service>
```

Fonte: próprio autor

Na primeira inicialização do app, o SDK do FCM gera um token de registro do dispositivo que identifica a instância do aplicativo de maneira exclusiva que se registra para receber mensagens. Esse token é utilizado como parâmetro para enviar as mensagens utilizando o recurso FCM do Firebase (ANDROID DEVELOPERS, 2019). A Figura 60 apresenta o código para recuperar o token de registro atual.

Figura 61 – Código para recuperar o token de registro do dispositivo

```
01  private void atualizarTokenUsuario(final Usuario usuario) {
02      FirebaseInstanceId.getInstance().getInstanceId()
03              .addOnCompleteListener(new OnCompleteListener<InstanceIdResult>() {
04          @Override
05          public void onComplete(@NonNull Task<InstanceIdResult> task) {
06              if (!task.isSuccessful()) {
07                  Log.e( tag: "ERRO NO TOKEN", String.valueOf(task.getException()));
08              } else {
09                  final String tokenAtualizado = task.getResult().getToken();
10                  final String tokenAntigo = usuario.getFirebaseToken();
11
12                  usuarioDAO.getUsuarioCollectionReference()
13                          .document(usuario.getUid())
14                          .update( field: "firebaseToken", tokenAtualizado);
15
16                  sharedPreferencesDAO.atualizarFirebaseTokenUsuarioLogado(tokenAtualizado);
17              }
18          }
19      });
20  }
```

Fonte: próprio autor

Na linha 2, foi utilizado o método *getInstanceId* da instância de *FirebaseInstanceId* para solicitar o token. Dentro do *callback OnCompleteListener*, na linha 9, é recuperado o token atual e é atualizado no banco de dados, conforme a linha 12.

Quando o aluno entra ou sai da zona georreferenciada, o método *adicionarMovimentacaoGeofence* que está dentro da instância do *BroadcastReceiver* responsável por identificar as transições na *geofence* é executado guardar as informações no Cloud *Firestore*, conforme descrito na seção 5.2.6. O código de implementação desse método é demonstrado na Figura 62.

Figura 62 – Código de implementação do método *adicionarMovimentacaoGeofence*

```
01  public void adicionarMovimentacaoGeofence(final MovimentacaoGeofence movimentacaoGeofence,
02                                             final Usuario usuarioResponsavel){
03      movimentacoesGeofenceCollectionReference.document(movimentacaoGeofence.getUidAluno()) DocumentReference
04              .collection( collectionPath: "movimentacoes_salvas").add(movimentacaoGeofence) Task<DocumentReference>
05              .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
06                  @Override
07                  public void onSuccess(DocumentReference documentReference) {
08                      long dataHorario = System.currentTimeMillis();
09                      NotificacaoMovimentacaoGeofence notificacaoMovimentacaoGeofence =
10                              new NotificacaoMovimentacaoGeofence(movimentacaoGeofence.getDataHora(),
11                                      movimentacaoGeofence.getAcao(), movimentacaoGeofence.getUidAluno(),
12                                      usuarioResponsavel.getUid());
13
```

```
14                  NotificacaoDAO notificacaoDAO = new NotificacaoDAO();
15                  notificacaoDAO.getNotificacoesMovimentacaoGeogenceCollectionReference()
16                          .document(usuarioResponsavel.getFirebaseToken())
17                          .set(notificacaoMovimentacaoGeofence);
18              }
19          })
20          .addOnFailureListener((e) -> {
21                  Log.e( tag: "ERRO", e.getMessage());
22          });
23  }
```

Fonte: próprio autor

Dentro do *callback OnSucessListener*, entre as linhas 9 e 17, é criada uma instância de *NotificacaoMovimentacaoGeofence*, contendo as informações de data/hora, a transição, o identificador único do aluno e do responsável que serão utilizadas para a construir a notificação no app do responsável. Da linha 15 a 17, através da instância de *NotificacaoDAO*, essas informações são salvas no Cloud Firestore, passando como parâmetro o token do usuário Responsável que receberá a notificação. Quando é criado um documento dessa Notificação no banco de dados, O FCM executa a função previamente instalada em *Cloud Functions* para enviar uma mensagem ao dispositivo do usuário Responsável.

Utilizando o Firebase Cloud Messaging, é possível enviar dois tipos de mensagens:

- *Mensagens de notificação*: são criadas pelo FCM e exibidas diretamente no dispositivo do usuário final em nome do app cliente. Essas mensagens possuem um conjunto predefinido de chaves visíveis pelo usuário e uma carga de dados opcional de pares de chave-valor personalizados.
- *Mensagens de dados*: o app do cliente fica responsável pelo processamento das mensagens de dados, podendo criar suas notificações personalizáveis. As mensagens de dados possuem apenas pares de dados de chave-valor personalizados sem nomes de chave reservados (ANDROID DEVELOPERS, 2019).

Na implementação do aplicativo, foi utilizado o tipo de mensagem de dados para ser enviado ao usuário Responsável. A Figura 63 apresenta o código para implementação da função de envio da mensagem.

Figura 63 – Código para implementar a função de envio da mensagem de dados pelo FCM

```
1  const functions = require('firebase-functions');
2  const admin = require('firebase-admin');
3
4  admin.initializeApp();
5
6  exports.notificacoes_geofence = functions.firestore
7      .document('notificacoes_movimentacao_geofence/{token}')
8      .onCreate((snap, context) => {
9          const document = snap.data();
10         console.log('document is', document);
11
12         var tokenID = context.params.token;
13         console.log('token is', tokenID);
14         var notificacao = {
15             data: {
16                 uidResponsavel: document.uidResponsavel,
17                 dataHorario: `${document.dataHorario}`,
18                 movimentacao: document.movimentacao,
19                 uidAluno: document.uidAluno,
20                 tipo: 'movimentacao_geofence'
21             },
22             //priority: "high",
23             token: tokenID
24         }
25
26         admin.messaging().send(notificacao)
27             .then((response) => {
28                 console.log('Mensagem enviada com sucesso! ', response);
29                 return;
30             })
31             .catch((error) => {
32                 console.log('Erro!', error);
33             })
34
35         admin.firestore()
36             .collection('notificacoes_movimentacao_geofence')
37             .doc(tokenID)
38             .delete();
39
40         return Promise.resolve(0);
41
42     });
```

Fonte: próprio autor

Na linha 12, é recuperado o token pelo caminho do documento do banco de dados que foi recentemente adicionado. Na linha 14, é criada a mensagem de dados personalizada contendo os pares de chave-valor *data* e *token.* Dentro de *data* contêm as informações de identificador único do usuário Responsável, data e horário, o tipo da movimentação (entrada/saída), o identificador único do usuário Aluno e o tipo na notificação que será processada pelo dispositivo do usuário final. Na linha 26, o

método *send* envia a mensagem e da linha 35 a 38, o documento da notificação é deletado do Cloud Firestore que foi criado anteriormente.

Para receber a mensagem no aplicativo cliente do dispositivo o usuário, é necessário utilizar o serviço *FirebaseMessagingService,* conforme explicado anteriormente. No método *onMessageReceived*, é onde deve ser tratado os dados para gerar a notificação a ser visualizada pelo o usuário. O código implementado para executar essa função está representado na Figura 64.

Figura 64 – Código para receber a mensagem do FCM

```
01  @Override
02  public void onMessageReceived(RemoteMessage remoteMessage) {
03      Map<String, String> dados = remoteMessage.getData();
04      UsuarioDAO usuarioDAO = new UsuarioDAO();
05
06      if(dados.get("tipo").equals("movimentacao_geofence")){
07          String uidResponsavel = dados.get("uidResponsavel");
08          final long dataHorario = Long.parseLong(dados.get("dataHorario"));
09          final String movimentacao = dados.get("movimentacao");
10          String uidAluno = dados.get("uidAluno");
11
12          usuarioDAO.getUsuarioCollectionReference() CollectionReference
13                  .document(uidAluno) DocumentReference
14                  .get() Task<DocumentSnapshot>
15                  .addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
16                      @Override
17                      public void onSuccess(DocumentSnapshot documentSnapshot) {
18                          Usuario usuarioAluno = documentSnapshot.toObject(Usuario.class);
19                          CarregarNotificacaoImagemPerfilUsuario carregarNotificacaoImagemPerfilUsuario
20                                  new CarregarNotificacaoImagemPerfilUsuario();
21                          carregarNotificacaoImagemPerfilUsuario.setMovimentacao(movimentacao);
22                          carregarNotificacaoImagemPerfilUsuario.setDataHorario(dataHorario);
23                          carregarNotificacaoImagemPerfilUsuario.execute(usuarioAluno);
24
25                      }
26                  });
27
28      }
29  }
30
31  public class CarregarNotificacaoImagemPerfilUsuario extends AsyncTask<Usuario,Void,Void> {
32
33      String movimentacao;
36      long dataHorario;
37      public void setMovimentacao(String movimentacao){
38          this.movimentacao = movimentacao;
39      }
40      public void setDataHorario(long dataHorario){
41          this.dataHorario = dataHorario;
42      }
43
```

```
44      @Override
45      protected Void doInBackground(Usuario... usuarios) {
46          Usuario usuarioAluno = usuarios[0];
47          Intent notificacoesIntent = new Intent(getApplicationContext(),
48                  TelaPrincipalUsuarioResponsavelActivity.class);
49          notificacoesIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
50          PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),
51                  requestCode: 0, notificacoesIntent, flags: 0);
52          NotificationManager notificationManager =
53                  (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
54          String notificationChaneelId = "channel_notificacao";
55          String notificationChannelNome = "notificacao_geofence";
56
57          if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
58              NotificationChannel notificationChannel =
59                      new NotificationChannel(notificationChaneelId, notificationChannelNome,
60                              NotificationManager.IMPORTANCE_HIGH);
61              notificationChannel.setDescription("channel_descricao");
62              notificationChannel.setLightColor(getApplicationContext().getResources()
63                      .getColor(R.color.cor_primaria_escura));
65              notificationManager.createNotificationChannel(notificationChannel);
66          }
67
68          String movimentacao_aluno;
69          if(movimentacao.equals("entrada")){
70              movimentacao_aluno = "Chegou ao";
71          } else {
72              movimentacao_aluno = "Saiu do";
73          }
74
75          Funcoes funcoes = new Funcoes();
76          NotificationCompat.Builder notificacaoBuilder =
77                  new NotificationCompat.Builder(getApplicationContext(),
78                          Constantes.NOTIFICACAO_CHANNEL_ID);
79
80          try{
81              Bitmap imagemPerfilUsuario = Picasso.with(getApplicationContext())
82                      .load(usuarioAluno.getFotoURL()).get();
83
84              notificacaoBuilder
85                      .setSmallIcon(R.drawable.ic_notificacao)
86                      .setLargeIcon(imagemPerfilUsuario)
87                      .setColor(getApplicationContext().getResources()
88                              .getColor(R.color.cor_primaria_escura))
89                      .setContentText(movimentacao_aluno + " IFAM às "
90                              + funcoes.converterHorarioMilisegundosEmString(dataHorario))
91                      .setContentTitle(usuarioAluno.getLogin() + " ("
92                              + funcoes.converterLetrasNome(usuarioAluno.getNome()) + ")")
93                      .setContentIntent(pendingIntent)
94                      .setPriority(NotificationCompat.PRIORITY_HIGH)
95                      .setAutoCancel(true)
96                      .setLights(Color.BLUE, onMs: 1000, offMs: 5000);
99
100             notificationManager.notify(Constantes.FLAG_USUARIO_ALUNO, notificacaoBuilder.build());
101         } catch (IOException e) {
102             e.printStackTrace();
103         }
104         return null;
105     }
106 }
```

Fonte: próprio autor

Na linha 3, é recuperado o conjunto de pares chave-valor através do método *getData* da instância de *RemoteMessage*. Na linha 12, é recuperado o documento com todas as informações do perfil do Aluno que transitou na *geofence*. Da linha 19 é 23, é criada uma instância de *CarregarNotificacaoImagemPerfilUsuario* que executa seu método *doInBackground* em segundo plano. Da linha 84 a 96, é configurada a notificação que deverá ser visualizada pelo o usuário. Por fim, na linha 100, é exibida a notificação no dispositivo do Responsável.

5.4. TELAS DO APLICATIVO

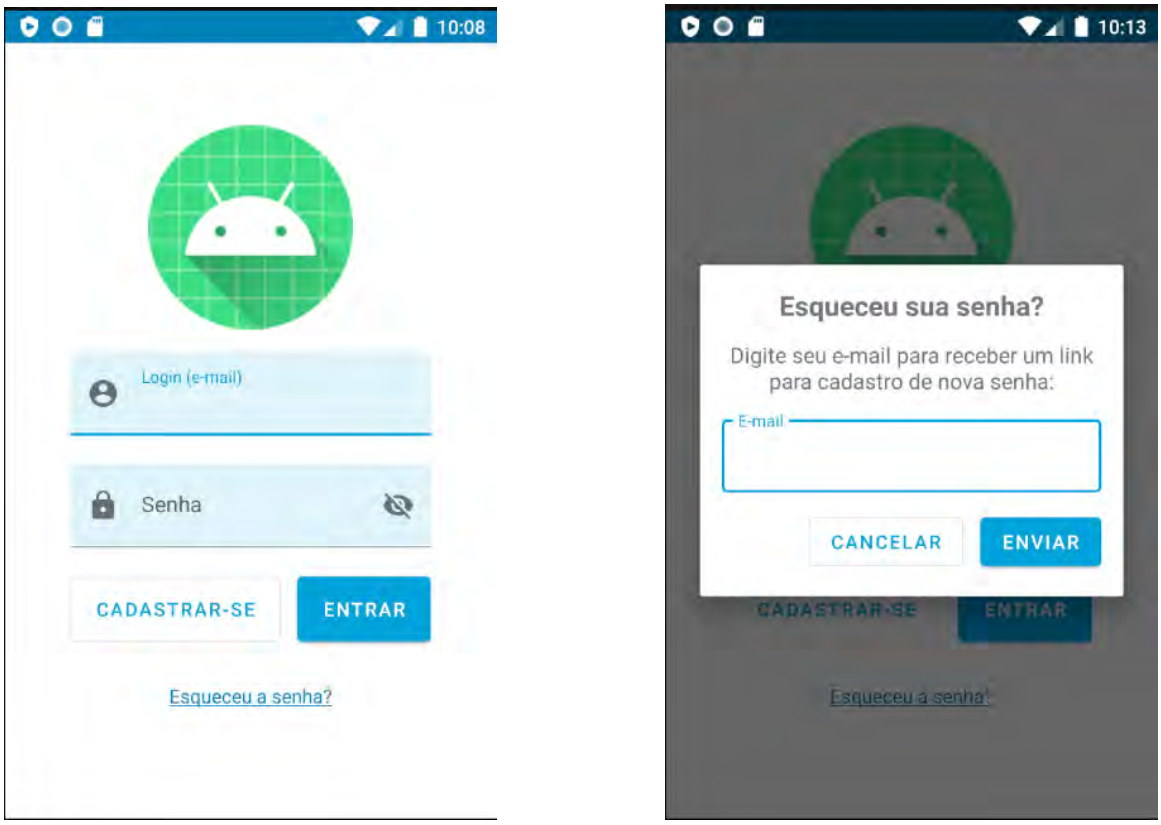
Para a experiência interativa do usuário, foram criadas 9 telas

- Tela de Login do usuário
- Tela de Cadastro de perfil do usuário
- Tela Principal (perfil Responsável)
- Tela Principal (perfil Aluno)
- Tela de Configurações (perfil Aluno)
- Tela de Vínculos (perfil Responsável)
- Tela de Vínculos (perfil Aluno)
- Tela de Notificações (perfil Responsável)
- Tela de Rotas (perfil Responsável)

5.4.1. Tela de Login do usuário

Após a primeira inicialização do aplicativo, o usuário visualiza essa tela onde lhe é permitido fazer o login preenchendo os campos de e-mail ou fazer o cadastro de um novo usuário. É possível recuperar a senha, caso o usuário não lembre, tocando no link Esqueceu a senha?. Em seguida será exibido uma caixa de diálogo para que o usuário insira o e-mail para a recuperação da senha. A Figura 65 exibe todas as telas alternativas da Tela de Login.

Figura 65 – Telas alternativas da Tela de Login



(a) Tela Login

(b) Caixa de diálogo Esqueceu sua Senha?

Fonte: próprio autor

5.4.2. Tela de Cadastro de perfil do usuário

Nessa tela (Figura 66), o usuário deve inserir suas informações como: e-mail, login, senha, confirmação da senha, nome completo, perfil e, opcionalmente, carregar uma imagem da galeria ou tirar uma foto com a câmera do celular. Pode-se optar por carregar algumas dessas informações utilizando a conta do Google ou do Facebook.

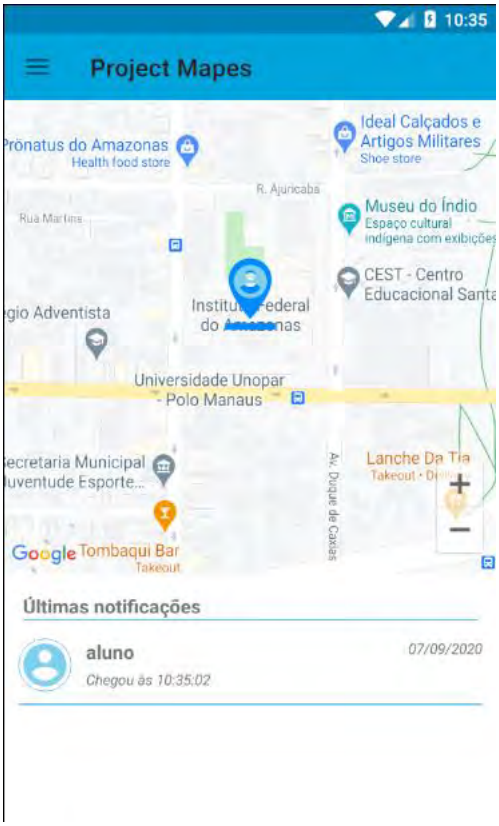
Figura 66 – Tela de Cadastro de usuário

Fonte: próprio autor

5.4.3. Tela principal (perfil Responsável)

Nessa tela (Figura 67), o usuário Responsável pode visualizar a localização em tempo real do usuário Aluno com quem tem vínculo, além de poder visualizar as últimas notificações recebidas.

Figura 67 – Tela principal – perfil Responsável

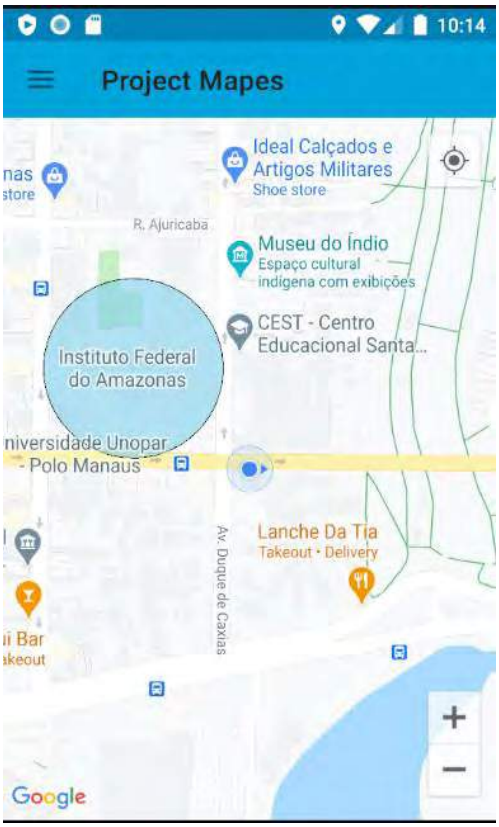


Fonte: próprio autor

5.4.4. Tela principal (perfil Aluno)

Nessa tela (Figura 68), o usuário consegue visualizar a *Geofence* e a sua posição atual no Google Maps.

Figura 68 – Tela principal – perfil Aluno

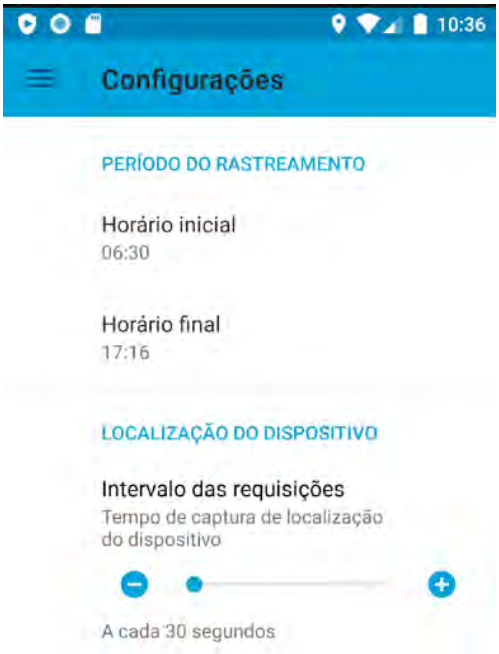


Fonte: próprio autor

5.4.5. Tela de configurações (perfil Aluno)

A Figura 69 exibe a Tela de Configurações para que o usuário Aluno possa configurar o período de rastreamento e o intervalo das requisições de localização para serem salvas e, posteriormente, criar as rotas.

Figura 69 – Telas de configurações – perfil Aluno

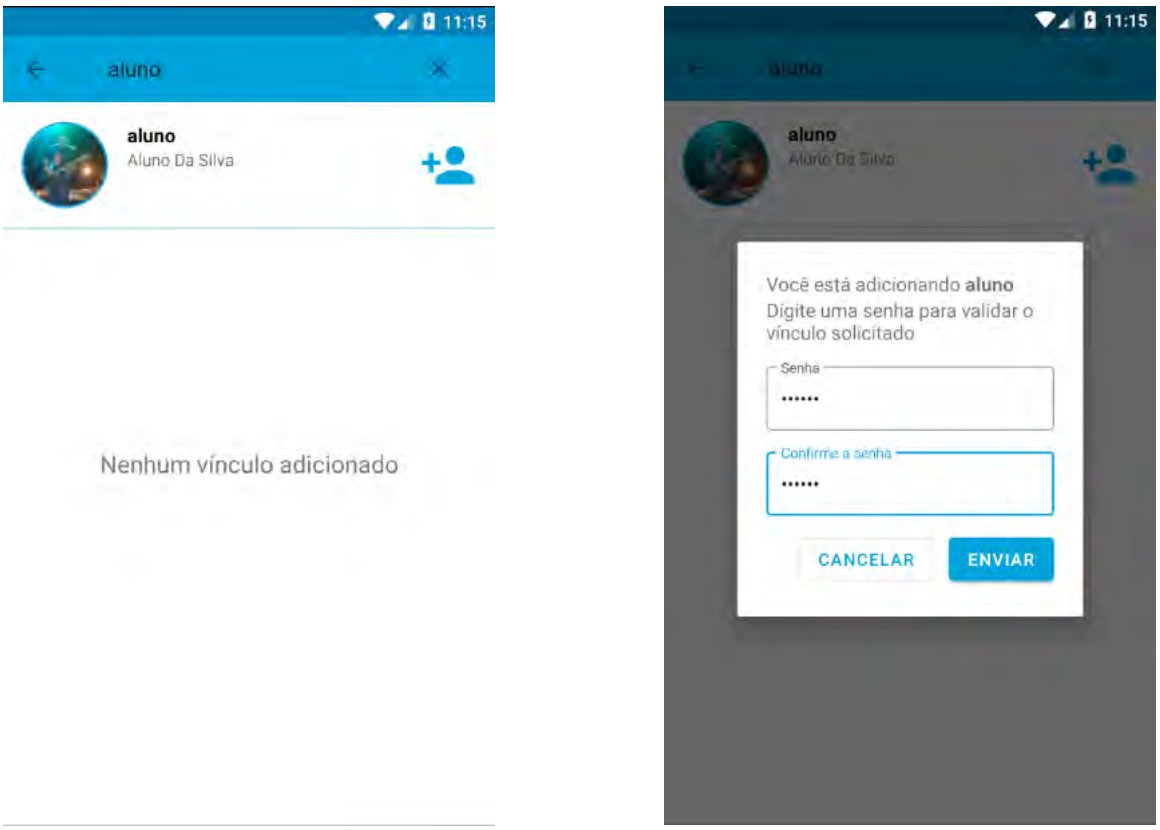


Fonte: próprio autor

5.4.6. Tela de Vínculos (perfil Responsável)

Nessa tela, o usuário Responsável pode pesquisar os usuários Aluno e solicitar vínculo. Após encontrar o usuário e tocar no botão de adicionar, é exibida uma caixa de diálogo para que o Responsável crie uma senha de validação do vínculo solicitado. A Figura 70 exibe as alternativas para a Tela de Vínculos para o perfil Responsável.

Figura 70 – Telas alternativas da Tela de Vínculos – perfil Responsável



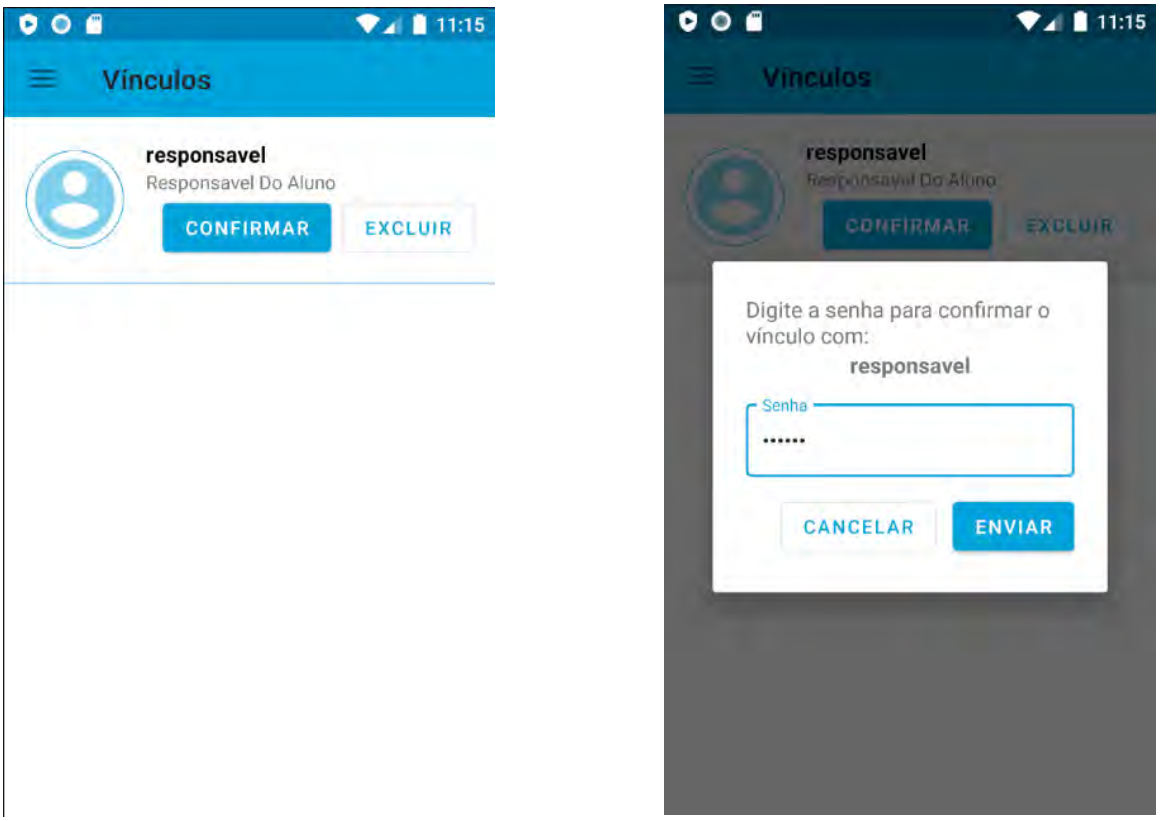
(a) Pesquisa de usuários na Tela Vínculos (b) Caixa de diálogo para criar senha de validação do vínculo

Fonte: próprio autor

5.4.7. Tela de Vínculos (perfil Aluno)

Nessa tela, o usuário Aluno pode visualizar as solicitações de vínculos recebidas. Após tocar no botão confirmar, é exibida uma caixa de diálogo para que o Aluno insira a senha de validação criada pelo Responsável. A Figura 71 exibe as alternativas para a Tela de Vínculos para o perfil Aluno.

Figura 71 – Telas alternativas pela Tela de Vínculos – perfil Aluno



(a) Solicitações de vínculos recebidas

(b) Caixa de diálogo para inserir a senha de validação

Fonte: próprio autor

5.4.8. Tela de Notificações (perfil Responsável)

A Figura 72 exibe a Tela de Notificações para que o usuário Responsável possa pesquisar as notificações de chegada e saída de um usuário Aluno com quem tem vínculo dentro de intervalo de datas informado

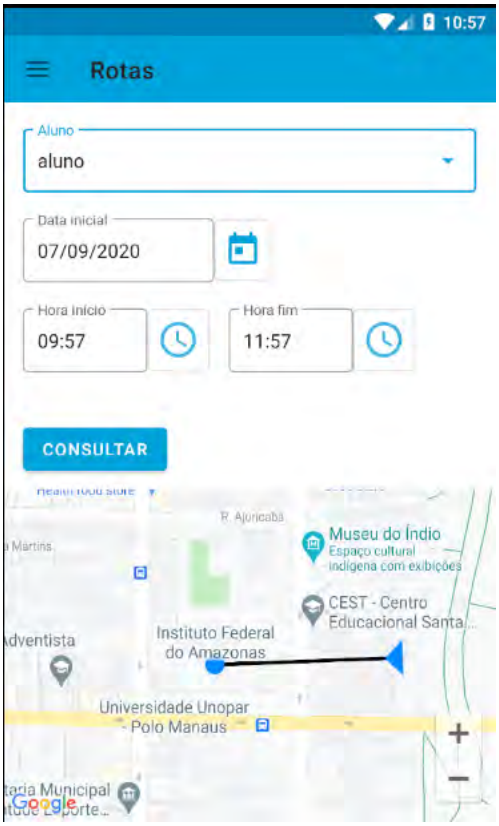
Figura 72 – Telas alternativas pela Tela de Login

Fonte: próprio autor

5.4.9. Tela de Rotas (perfil Responsável)

Nessa tela, o usuário Responsável pode visualizar a rota que o usuário Aluno fez dentro de um intervalo de horas e de data informados. Essa tela pode ser observada na Figura 73.

Figura 73 – Telas alternativas pela Tela de Login



Fonte: próprio autor

6. CONCLUSÕES FINAIS

A aflição e a preocupação de pais e responsáveis com seus filhos adolescentes e jovens quando estes saem de casa têm se tornado cada vez mais evidente. Considerando que o grupo composto por jovens de 15 a 19 anos é o mais afetado pela violência, é importante considerar e procurar formas para mitigar essa preocupação. Por isso, o aplicativo MAPES é tido como uma alternativa para tentar resolver esse problema, e pode ser instalado em um dispositivo móvel com sistema operacional Android.

Sendo assim, neste trabalho foi apresentado a implementação da proposta de desenvolvimento do aplicativo que é capaz de auxiliar os pais na visualização da localização em tempo real de seus filhos dentro da escola, além de enviar notificações quando os alunos chegam e/ou saiam da instituição. A principal influência para escolher o desenvolvimento de um aplicativo Android, foi o fato de que o número de usuários de smartphones já superou os usuários exclusivos de dispositivos computadores, notebooks e tablets e que esse dispositivo móvel dispões de vários recursos úteis e cada vez mais modernos.

6.1. DIFICULDADES ENCONTRADAS

Apesar de que todos os objetivos estabelecidos foram alcançados, algumas adversidades dificultaram o desenvolvimento do projeto: (1) implementação do serviço de rastreamento em versões diferentes do sistema operacional Android: observou-se que a partir da versão 8.0 (Oreo), o sistema restringiu o a frequência com que os apps em background possam recuperar a localização atual do usuário com o intuito de reduzir o consumo de energia no dispositivo. Por isso foi necessário buscar alternativas para implementar o serviço nas versões mais recentes, o que demandou um tempo maior de pesquisa. (2) A Google não carregou a planta baixa do IFAM-CMC para utilização do API do Google Indoors devido a prioridade dada a plantas de shoppings e parceiros já existentes, conforme pode ser observado na resposta dada na Figura 74. Isso impactou na precisão da visualização em tempo real da posição do aluno, já que pretendia-se implementar os recursos da API para o responsável pudesse identificar o andar e a sala que seu filho está, por exemplo. (3) em virtude da suspensão das atividades acadêmicas presenciais no Campi do IFAM iniciada em 18

de março como medida de controle ante a pandemia de COVID-19 no estado, os testes reais do aplicativo ficaram impossibilitados.

Figura 74 – Resposta Google Indoor

Gmail — Daniel Santos <danesansilva@gmail.com>

Status upload planta Instituto Federal de Educação, Ciência e Tecnologia do Amazonas
2 mensagens

Daniel Santos <danesansilva@gmail.com> 16 de junho de 2020 14:22
Para: indoorpartners-americas@google.com
Cc: Jucimar Souza <Jucibs@gmail.com>, Maison Marcel Madri Galvão dos Santos Souza <maisongalvao144@gmail.com>

Boa tarde, prezado(a).

No dia 12/05/2020, carreguei duas plantas para a seguinte construção:

Instituto Federal de Educação, Ciência e Tecnologia do Amazonas
Av. Sete de Setembro, 1975 - Centro, Manaus - AM, 69020-120, Brasil
Coordenadas: -3.1338407, -60.0127647

Gostaria de saber o status dessa solicitação e, se possível, o prazo para conclusão da análise do upload.

Desde já, agradeço.

indoorpartners-americas <indoorpartners-americas+noreply@google.com> 16 de junho de 2020 14:23
Para: danesansilva@gmail.com

Hello,

Thank you for contacting us. The Indoor Google Maps project is currently focused on malls and existing partners, only.

If you're one of our existing partners or your organization is a shopping mall, one of our representatives will be reaching out to you shortly.

To see in which countries the project is available, check here: https://support.google.com/maps/answer/1685827?hl=en.

For all other inquiries, we've recorded your interest and will reach back out shall the project expand to other venue types in the future.

Thank you,
The Indoor Google Maps Team

Fonte: Gmail – Google Maps Team

6.2. DIRECIONAMENTO PARA TRABALHOS FUTUROS

O protótipo do aplicativo se mostrou bastante funcional e atendeu a todos os requisitos propostos. Ainda assim, outras *features* podem ser adicionadas, tais como:

- Visualização mais precisa da posição do aluno (depende do download da planta baixa da escola)
- Permitir que o aluno pesquise a localização de salas ou laboratórios dentro da instituição (depende do download da planta baixa da escola)
- Utilizar chat para que o responsável e o aluno possam se comunicar
- Customização dos ícones dos usuários no mapa, além de inserir outras informações úteis de interação
- Aplicar o uso das áreas georreferenciadas em outras escolas da cidade
- Disponibilização do app para download na Play Store

REFERÊNCIAS BIBLIOGRÁFICAS

A MENTE É MARAVILHOSA. **A preocupação obsessiva de manter as crianças em segurança**, fevereiro, 2018. Disponível: https://amenteemaravilhosa.com.br/preocupacao-manter-as-criancas-em-seguranca/

ABLESON, Frank. KING, Chris, SEN, Robi. **Android em ação**, 3a ed. Elsevier Brasil, 2012. Acesso em 11 de agosto de 2019.

ALVES, Daniel. CALADO, Ivo. SILVA, Gilton. **Um Sistema de Recomendação de Pontos de Interesse Baseados na Localização e no Contexto Social do Usuário**. VII CONEPI. 2012. Disponível em < http://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/5137/1001> Acesso em 12 de julho de 2019.

AMAL W., Raj. **Learning Android Google Maps**. Ed. Packt Publishing. 2015. Acesso em 19 de agosto de 2019.

ANDROID DEVELOPERS. **Criar e monitorar fronteiras geográficas virtuais**. Disponível em: <https://developer.android.com/training/location/geofencing.html>. Acesso em 15 de agosto de 2019.

APPS4BUSINESS. **Marista Virtual**. Disponível em: <http://apps4business.com.br/portfolio/marista-virtual/>. Acesso em 10 de agosto de 2019.

ATLAS DA VIOLÊNCIA 2019. Instituto de Pesquisa Econômica Aplicada; Fórum Brasileiro de Segurança Pública Disponível em: <http://www.crianca.mppr.mp.br/arquivos/File/publi/ipea/atlas_da_violencia_2019.pdf >. Acesso em 3 de junho de 2019.

BAHL, Paramvir; PADMANABHAN, Venkata N. **Radar: An In-Building RF-based User Location and Tracking System.** Em Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, volume 2, pp. 775 – 784. Disponível em < http://www.cs.columbia.edu/~drexel/CandExam/RADAR_in_building_RF.pdf >. Acesso em 26 de julho de 2019.

BARBOSA, Cícero. **A segurança nas escolas públicas: Corremos riscos?** Jornal do Sudoeste, 2019. Disponível em <http://www.jornaldosudoeste.com.br/noticia.php?codigo=202889> Acesso em 20 de maio de 2019.

BEZERRA, Eduardo. **Princípios de análise e projetos de sistemas com UML.** Ed. Elsevier. Rio de Janeiro, 2007. Acesso em 19 de janeiro de 2020.

BOOCH Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML, Guia do Usuário**. Ed.Campus.Rio de Janeiro, 2000. Acesso em 3 de fevereiro de 2020.

CORREIA, Adolfo Guilherme Silva. **Aplicações e serviços baseados em localização**. 2004. 16 f. Programa de Mestrado, Introdução à Computação Móvel, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2004. Disponível

em < http://www-di.inf.puc-rio.br/~endler/courses/Mobile/Monografias/04/AdolfoCorreia-Mono.pdf>. Acesso em 11 de agosto de 2019.

COSTA, I. d. O. **Modelos para análise de disponibilidade em uma plataforma de mobile backend as a service.** Universidade Federal de Pernambuco, 2015. Disponivel em <https://attena.ufpe.br/bitstream/123456789/15952/1/Disserta__o_Igor_Costa__Copy_%281%29.pdf>. Acesso em 26 de agosto.

EGGEA, Rodrigo Fagundes. **Aplicação Android utilizando sistema de localização geográfica para determinação de pontos turísticos na cidade de Curitiba**. 2013. 57 f. Trabalho de Conclusão de Curso (Especialização) – Universidade Tecnológica Federal do Paraná, Curitiba, 2013. Disponível em < http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2484/1/CT_TECJAVMOV_I_2012_10.pdf> Acesso em 12 de julho de 2019.

EXAME, **Controle de acesso nas escolas garante a segurança dos alunos**. Disponível em: <http://exame.abril.com.br/negocios/dino/controle-de-acesso-nas-escolas-garante-a-seguranca-dos-alunos-shtml/>. Acesso em 10 de junho de 2019.

FERRARO, Richard; AKTIHANOGLU, Murat. **Location-Aware Applications**. Shelter Island: Manning Publications, 2011. Acesso em 3 de julho de 2019

FIREBASE. **Adicionar o Firebase ao projeto para Android.** Disponível https://firebase.google.com/docs/android/setup#create-firebase-project. Acesso em 25 de março de 2020.

GALON, Handrey E. **Sistema de rastreamento e controle de recursos de um veículo utilizando um smartphone Android.** 2014. 79 f. Monografia de Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2014. Disponível em <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/4068/1/PB_COENC_2014_1_04.pdf> Acesso em 30 de julho de 2019.

GIBSON, Rich. ERLE, Schuyler. **Google Maps Hacks.** Ed O'Reilly 2006. Acesso em 18 de agosto de 2019.

GOOGLE. **Fence API Overview.** Google Awareness API. 2019. Disponível em: <https://developers.google.com/awareness/android-api/fence-api-overview>. Acesso em 21 de agosto de 2019.

GOOGLE. **Google Indoor Maps.** Google Maps. Disponível em: <https://www.google.com/intl/pt-BR/maps/about/partners/indoormaps>. Acesso em 22 de agosto de 2019.

GOOGLE. **Geolocalização: Como exibir a posição do usuário ou do dispositivo no Maps.** Disponível em: <https://developers.google.com/maps/documentation/javascript/geolocation?hl=pt-br>. Acesso em 20 de agosto de 2019.

GOOGLE MAPS INDOORS. **Upload de plantas.** Disponível em < http://maps.google.com/floorplans/find>. Acesso em 21 de agosto de 2019.

GUEDES, Gilleanes T. A. **UML 2: Guia Prático**, 2ª Ed. Editora Novatec, 2014. Acesso em 19 de janeiro de 2020.

HJELM, Johan. **Creating Location Services for the Wireless Web: Professional Developer's Guide**. John Wiley & Sons, 2003. Acesso em 22 de julho de 2019.

IBGE, **Projeção da população do Brasil e das Unidades da Federação.** Disponível em: <https://www.ibge.gov.br/apps/populacao/projecao/>. Acesso em 15 de maio de 2019.

IHA, **Índice de Homicídios na Adolescência: IHA 2014**. p. 108. Organizadores: Doriam Luis Borges de Melo, Ignácio Cano - Rio de Janeiro. Observatório de Favelas, 2017. Acesso em 15 de maio de 2019.

JOHNSON, Thienne M**. Java para dispositivos moveis: desenvolvendo aplicações com J2ME.** Editora Novatec, 2007. Acesso em 18 de julho de 2019.

KUPPER, Axel. **Location-based services: Fundamentals and Operation**. John Wiley & Sons Ltda, 2005. Acesso em 10 de julho de 2019

LEAL, Nelson Glauber de Vasconcelos. **Dominando o Android**. Editora Novatec. 2ª Ed. 2016. Acesso em 17 de agosto de 2019.

LECHETA, Ricardo R. **Android essencial.** Editora Novatec, 2016. Acesso em 11 de agosto de 2019.

MACHADO, Everton Fábio da Silva. **Desenvolvimento de sistemas de geolocalização e rastreamento para a plataforma android – COMPASS.** Monografia de especialização, 2015. 55 f. Trabalho de Conclusão de Curso (Especialização em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis) – Universidade Tecnológica Federal do Paraná, Francisco Beltrão, 2015. Disponível em < http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6914/1/FB_DESIDM_I_2014_01.pdf>. Acesso em 1 de agosto de 2019.

MACK, Roger Schneider. **Sistema de recomendação baseado na localização e perfil utilizando a plataforma Android.** 2010. 56 f. Trabalho de Graduação (Ciências da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010. Disponível em < https://www.lume.ufrgs.br/bitstream/handle/10183/28328/000767836.pdf >. Acesso em 13 de agosto de 2019.

MALLICK, Martyn. **Mobile and Wireless Design Essentials.** Wiley Publishing, United States, 2003. Acesso em 19 de julho de 2019.

MELL, P.; GRANCE, T. et al. **The nist definition of cloud computing.** Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011. Disponível em < https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>. Acesso em 25 de agosto de 2019.

MENDES, Leonardo Manoel; COSTA, Rogério Homem da; LORENSO, Reinaldo. **O Gerenciamento de requisitos e a sua importância em projetos de**

desenvolvimento de software. Revista Gestão Universitária, Ed. 2015, Volume 4, publicado em 13/10/2015. Disponível em < http://www.gestaouniversitaria.com.br/system/scientific_articles/files/000/000/105/original/O_GERENCIAMENTO_DE_REQUISITOS_E_A_SUA_IMPORT%C3%82NCIA_EM_PROJETOS_DE_DESENVOLVIMENTO_DE_SOFTWARE.pdf?1444586543>. Acesso em 22 de setembro de 2019.

MOURA, André Iasi. **WBSL: um sistema de localização de dispositivos móveis em redes Wi-Fi.** 2007. 134 f. Dissertação de Mestrado em Engenharia Elétrica – Escola Politécnica da Universidade de São Paulo, São Paulo, 2007. Disponível em < https://teses.usp.br/teses/disponiveis/3/3141/tde-07012008-171855/publico/dissertacao_andre_cf.pdf>. Acesso em 20 de julho de 2019.

MONTEIRO, Erich Farias. **Um framework para o gerenciamento da informação de localização**. 2005. 112 f. Dissertação de Mestrado em Engenharia de Eletricidade - Universidade Federal do Maranhão, São Luís, 2005. Disponível em < https://tedebc.ufma.br/jspui/bitstream/tede/396/1/Erich%20Farias%20Monteiro.pdf>. Acesso em 17 de julho de 2019.

NODE.JS. **Sobre Node.js.** Disponível em <http://nodejs.org/pt-br/about/>. Acesso em 17 de abril de 2020.

OLIVEIRA, Cristina de Chan de. **Estudo sobre a utilização de mapas em Android para itinerários de ônibus.** 2012. 47 f. Monografia de Especialização em Tecnologia Java - Universidade Tecnológica Federal do Paraná, Curitiba, 2012. Disponível em <http://www2.dainf.ct.utfpr.edu.br/esp/monografias-de-especializacao-da-turma-vii-2011-2012/CT_JAVA_VII_2011_04.PDF>. Acesso em 17 de agosto de 2019.

OMG UML, **OMG Unified Modeling Language™, Superstructure 2.4.1**. Disponível em <http://www2.imm.dtu.dk/courses/02291/files/UML2.4.1_superstructure.pdf>. Acesso em 15 de setembro de 2019.

ORLANDI, Claudio. **Firebase: serviços, vantagens, quando utilizar e integrações.** Rocketseat, 2018. Disponível em < https://blog.rocketseat.com.br/firebase/>. Acesso em 25 de março de 2020.

PACTO SOLUÇÕES, **Gestão de escolas: mais segurança e controle do acesso e frequência dos alunos**. Disponível em: <http://pactosolucoes.com.br/gestao-de-escolas-mais-seguranca-e-controle-do-acesso-e-frequencia-dos-alunos/>. Acesso em 17 de junho de 2019.

PAIVA, J. C.; LEAL, J. P.; QUEIRÓS, R. A. P. de. **Design and implementation of an ide for learning programming languages using a gamification service.** Gamification-Based E-Learning Strategies for Computer Programming Education, IGI Global, p. 295–308, 2016. Acesso em 25 de março de 2020.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software:** fundamentos, métodos e padrões. São Paulo: LTC Editora, 2000.

PEREIRA, Caio Ribeiro. **Aplicações web real-time com Node.js**, 185. Ed. Casa do código. Acesso em 17 de abril de 2020.

PEREIRA, Luiz Antônio de Moraes. **Análise e Modelagem de Sistemas com a UML.** 1ª Ed. Rio de Janeiro, 2011. Acesso em 27 de janeiro de 2020.

PEZZINI, Clenilda Cazarin. SZYMANSKI, Maria Lidia Sica. **Falta de desejo de aprender: Causas e Consequências**. 2008. Disponível em: < http://www.diaadiaeducacao.pr.gov.br/portals/pde/arquivos/853-2.pdf>. Acesso em 19 de maio de 2019.

PRESSMAN, Roger S.. **Engenharia de Software: uma abordagem profissional.** Editora Bookman, 7ª ed., Porto Alegre, 2011. Acesso em 10 de fevereiro de 2020.

RABELLO, Ramon Ribeiro. **Android: um novo paradigma de desenvolvimento móvel**. Disponível em: <https://dpwinfo.files.wordpress.com/2013/04/wm18_android.pdf>. Acesso em 12 de agosto de 2019.

RODRIGUES, Moisés Lisboa. **Localização em ambientes internos utilizando múltiplas tecnologias sem fio.** 2011. 145 f. Dissertação em Pós-Graduação em Ciência da Computação - Universidade Federal de Minas Gerais, Belo Horizonte, 2011. Disponível em <https://www.dcc.ufmg.br/pos/cursos/defesas/1329M.PDF>. Acesso em 23 de julho de 2019.

ROSSONI, Emanuele Carla. **Protótipo de Gerenciamento de *pet shops***. 85 f. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná, Pato Branco, 2011. Disponível em < http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/201/1/PB_COADS_2011_1_10.pdf>. Acesso em 29 de janeiro de 2020.

SABOYA, Patrícia. **A juventude brasileira e a violência**. Disponível em: <http://www.crianca.mppr.mp.br/modules/conteudo/conteudo.php?conteudo=260>. Acesso em 10 de junho de 2019.

SCHIMITT, Peterson Ricardo Maier. **Aplicação web utilizando API Google Maps**. 2013. 39 f. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná, Medianeira, 2013. Disponível em < http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/1718/1/MD_COADS_2012_2_06.pdf>. Acesso em 19 de agosto de 2019.

SERRA, Tatiana. **Matar aula: Reconquistando e controlando os alunos para que não haja faltas**. Disponível em: <http://www.educacaopublica.rj.gov.br/biblioteca/educacao/0266.html>. Acesso em 19 de maio de 2019.

SILVA, Luciano Alves da. **Apostila de Android – Programação Passo a Passo. Programação Básica (Versão Android Studio).** 2015. Disponível em: < http://www.leonardoleandrodev.com.br/downloads/apostila-android.pdf>. Acesso em 11 de agosto de 2019.

SILVA, Grace Kelly de Castro. PEREIRA, Patrícia Maria. MAGALHÃES, Geovane Cayres. **Disponibilização de serviços baseados em localização via *web services***. P. 331-342, 2004. Disponível em < http://mtc-m16c.sid.inpe.br/col/dpi.inpe.br/geoinfo@80/2006/08.21.14.02/doc/23_ok6392.pdf>. Acesso em 3 de julho de 2019

SILVA, Bárbara Tarouco da; SILVA, Mara Regina Santos da. **Necessidades e preocupações dos pais em diferentes etapas do ciclo vital**. Rev. bras. enferm., Brasília, v. 67, n. 6, p. 957-964, Dez. 2014. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-71672014000600957&lng=en&nrm=iso>. Acesso em 10 de junho de 2019.

SILVA, Alberto Manuel Rodrigues da; VIDEIRA, Carlos Alberto Escaleira. **UML, Metodologias e Ferramentas CASE**. Ed. Centro Atlântico, Portugal, 2001. Acesso em 25 de janeiro de 2020.

SOMMERVILLE, Ian. **Engenharia de Software.** Editora Pearson Prentice Hall, 9. ed. São Paulo, 2011.

TOPDATA, **Controle de acesso em escolas**. 2016. Disponível em: <https://www.topdata.com.br/controle-acesso-escolas/>. Acesso em 20 de maio de 2019.

VARGAS, Thânia Clair de Souza. **Suporte à Edição de UML 2 no Ambiente SEA.** 2008. 239 f. Trabalho de conclusão de curso (Graduação) - Universidade Federal de Santa Catarina, Florianópolis, 2008. Disponível em < https://projetos.inf.ufsc.br/arquivos_projetos/projeto_721/monografia.tcc.thania.clair.2008-1.pdf>. Acesso em 19 de janeiro de 2020.

VINHAL, Pedro**. O que é *geofencing* e como é útil para as PME?** PME APPS. 2015. Disponível em: <http://www.pmeapps.com/o-que-e-geofencing-e-como-e-util-para-as-pme/.> Acesso em 21 de agosto de 2019.

WAISELFISZ, Julio Jacobo. **Violência letal contra as crianças e adolescentes do Brasil.** 148 f. Relatório de pesquisa – Faculdade Latino-Americana de Ciências Sociais (Flacso), Brasil, 2015. Disponível em < http://flacso.org.br/files/2016/06/Viol%C3%AAncia_Letal_web.pdf>. Acesso em 3 de junho de 2019.

WILLIAMS, M. I. **A quick start guide to cloud computing: moving your business into the cloud.** [S.l.]: Kogan Page Publishers, 2010. Acesso em 25 de agosto de 2019.

APÊNDICE 1 - DESCRIÇÃO DOS CASOS DE USO DO APLICATIVO MAPES

CDU 01 – Cadastrar Perfil Responsável/Aluno

Tipo: Abstrato

Objetivo: Cadastrar o perfil do responsável ou do aluno com todos os seus dados, validando os campos obrigatórios e valores inválidos.

Ator: Responsável ou aluno

Pré-Condições: Instalar o aplicativo no smartphone

Pós-condições (Garantia de Sucesso): O usuário poderá fazer o login no aplicativo.

Cenário Principal:

1. Usuário toca no botão "*Cadastre-se*" presente na **Tela 01**.
2. Aplicativo exibe **Tela 02** para inserção dos dados.
3. O usuário informa os dados [Login], [E-mail], [Senha], [Confirmar Senha], [Nome completo] e seleciona uma foto para o perfil
4. Usuário seleciona o [Perfil] a ser cadastrado (Responsável ou Aluno) e toca no botão Salvar.
5. É realizada a validação das informações nos campos.
6. Aplicativo envia os dados para o sistema web para cadastro no banco de dados e mostra notificação "*Perfil cadastrado com Sucesso!*"
7. Aplicativo exibe a **Tela 01**.

Cenários Alternativos:

Campos Obrigatórios não preenchidos: Caso algum dos campos [Login], [E-mail], [Senha], [Confirmar Senha], [Nome Completo] e [Perfil] não tenha sido preenchido, o usuário será notificado através da mensagem "*Campo obrigatório".*

Login já em uso: Caso seja digitado um login que já exista na base de dados da aplicação web, será exibida a mensagem *"Login indisponível"*.

E-mail inválido: Caso o e-mail digitado seja inválido, será exibida a mensagem *“E-mail inválido”*;

E-mail já cadastrado: Caso o e-mail digitado já exista na base de dados da aplicação web, será exibida a mensagem *“E-mail já cadastrado”*.

Campos [Senha] e [Confirmar Senha] não coincidirem: será exibido mensagem *“Senhas não coincidem”*.

Vincular dados de redes sociais (Google/Facebook): será exibido uma tela para escolher a conta previamente habilitada no dispositivo. Após a confirmação da escolha, os campos [Login], [E-mail], [Nome Completo] e [Foto] serão automaticamente preenchidos.

Esboços de Tela

Tela 01:

Figura 75 – Tela de login

Fonte: próprio autor

Tela 02:

Figura 76 – Tela de cadastro de perfil

Fonte: próprio autor

CDU 02 – Fazer Login (Perfil Responsável)

Tipo: Abstrato

Objetivo: permitir acesso ao aplicativo.

Ator: Usuário utilizando perfil de Responsável

Pré-Condições: Ter um perfil cadastrado no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá acessar as principais funcionalidades do aplicativo.

Cenário Principal:

1. Aplicativo exibe a **Tela 01**.
2. Usuário informa seu [Login] / [E-mail] e [enha] e toca no botão “Entrar”
3. Aplicativo envia as informações para o sistema web e valida os dados.
4. Aplicativo exibe a **Tela 03**.

Cenários Alternativos:

Campos Obrigatórios não preenchidos: Caso algum dos campos [Login] / [E-mail] e [Senha] não tenham sido preenchidos, o usuário será notificado através da mensagem "*Login e senha obrigatórios".*

Senha incorreta: caso seja inserida a senha incorreta, será exibido a mensagem *"Senha incorreta"*. A partir disso, o usuário poderá recuperar a senha através do link [Esqueceu a senha?] (**estender CDU 03 - Esqueceu a senha?**)

Usuário não cadastrado: se os dados informados não estiverem no banco de dados, o aplicativo notificará com a mensagem "Perfil não cadastrado" Nesse caso, o usuário poderá se o cadastrar tocando no botão "Cadastre-se" (**estender CDU 01 - Cadastrar Perfil**).

Esboços de Tela

Tela 03:

Figura 77 – Tela principal do perfil Responsável

Fonte: próprio autor

CDU 03 – Fazer Login (Perfil Aluno)

Tipo: Abstrato

Objetivo: permitir acesso ao aplicativo.

Ator: Usuário utilizando perfil de Aluno

Pré-Condições: Ter um perfil cadastrado no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá acessar as funcionalidades do aplicativo para o perfil do aluno.

Cenário Principal:

1. Aplicativo exibe a **Tela 01**.
2. Usuário informa seu [Login] e [enha] e toca no botão “Entrar”
3. Aplicativo envia as informações para o sistema web e valida os dados.
4. Aplicativo exibe a **Tela 04**.

Cenários Alternativos:

Campos Obrigatórios não preenchidos: Caso algum dos campos [Login] e [Senha] não tenha sido preenchido, o usuário será notificado através da mensagem “*Login e senha obrigatórios”.*

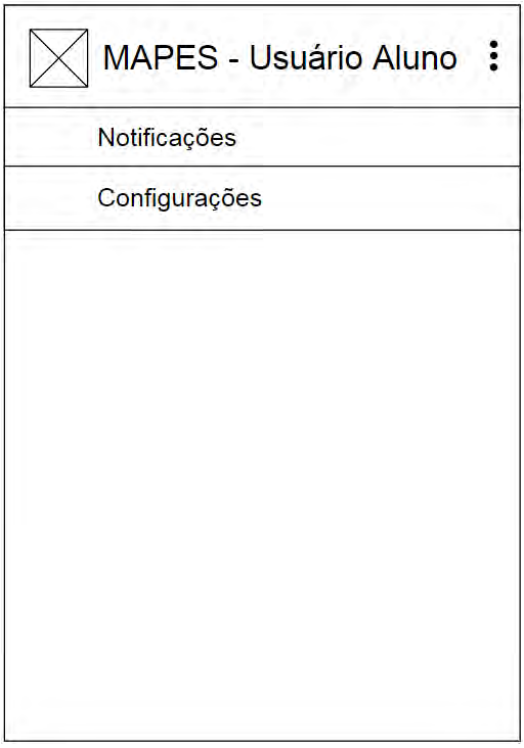
Senha incorreta: caso seja inserida a senha incorreta, será exibido a mensagem *“Senha incorreta”*. A partir disso, o usuário poderá recuperar a senha através do link [Esqueceu a senha?] (**estender CDU 04 - Esqueceu a senha?**)

Usuário não cadastrado: se os dados informados não estiverem no banco de dados, o aplicativo notificará com a mensagem ”Perfil não cadastrado” Nesse caso, o usuário poderá se o cadastrar tocando no botão “Cadastre-se” (**estender CDU 01 - Cadastrar Perfil**).

Esboços de Tela

Tela 04:

Figura 78 – Tela principal do perfil Aluno



Fonte: próprio autor

CDU 04 – Esqueceu a senha?

Tipo: Abstrato

Objetivo: permitir que o usuário recupere sua senha.

Ator: Usuário utilizando perfil Responsável ou Aluno

Pré-Condições: Ter um perfil cadastrado no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá recuperar sua senha para poder acessar o aplicativo.

Cenário Principal:

1. Aplicativo exibe a **Tela 01**.
2. Usuário clica no link [Esqueceu a senha?].

3. Aplicativo exibe uma notificação pop-up (**Tela 05**) para o usuário inserir seu e-mail de cadastro.
4. Aplicativo envia a informação para o sistema web, valida e o sistema envia e-mail com a senha cadastrada.
5. plicativo exibe notificação "*Sua senha foi enviada para o e-mail xxxx@xxxx.com.!*"
6. Aplicativo exibe a **Tela 01**.

Cenários Alternativos:

E-mail não preenchido: Caso o campo [e-mail] não tenha sido preenchido, o usuário será notificado através da mensagem "*Digite seu e-mail de cadastro".*

E-mail inválido: caso seja inserido um e-mail inválido, será exibido a mensagem *"O e-mail inserido é inválido"*

E-mail não cadastrado: se o e-mail informado não estiver no banco de dados, o aplicativo notificará com a mensagem "Nenhum perfil encontrado com o e-mail informado" Nesse caso, o usuário poderá se o cadastrar tocando no botão "Cadastre-se" (**estender CDU 01 - Cadastrar Perfil**).

Esboços de Tela

Tela 05:

Figura 79 – Tela para recuperar a senha

Fonte: próprio autor

CDU 05 – Visualizar notificação (perfil Responsável)

Tipo: Abstrato

Objetivo: permitir que o usuário responsável visualize a notificação de entrada/saída do aluno.

Ator: Usuário utilizando perfil de Responsável.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá visualizar as informações sobre o horário que o aluno chegou ou saiu da escola.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 03)**.

2. Usuário toca na notificação que quer visualizar, na parte inferior da Tela Principal
3. Aplicativo exibe uma notificação pop-up (**Tela 06**) contendo informações de horário de entrada/saída da escola.

Esboços de Tela

Tela 06:

Figura 80 – Tela de notificação

Fonte: próprio autor

CDU 06 – Visualizar posição atual do aluno

Tipo: Abstrato

Objetivo: permitir que o usuário Responsável visualize a posição atual do usuário Aluno

Ator: Usuário utilizando perfil Responsável.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá visualizar, em tempo real, a posição atual do aluno.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 03)**.
2. Através do mapa exibido, o usuário pode visualizar a posição atual do aluno, movimentando e aumentando o zoom do mapa.

CDU 07 – Visualizar as rotas de posição do aluno

Tipo: Abstrato

Objetivo: permitir que o usuário Responsável visualize a rota que o usuário Aluno fez em um determinado período.

Ator: Usuário utilizando perfil de Responsável.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá visualizar a rota do aluno.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 03)**.
2. Usuário toca no ícone de Mais Opções, no canto superior direito.
3. Aplicativo exibe a **Tela 07**
4. Usuário toca na opção [Rotas].
5. Aplicativo exibe a **Tela 08**.
6. O usuário informa os dados [Dia], [Horário Inicial], [Horário Final] e toca no botão [Consultar].
7. Aplicativo envia as informações de consulta para o Servidor Web.
8. Aplicativo exibe a **Tela 09** com a rota do aluno.

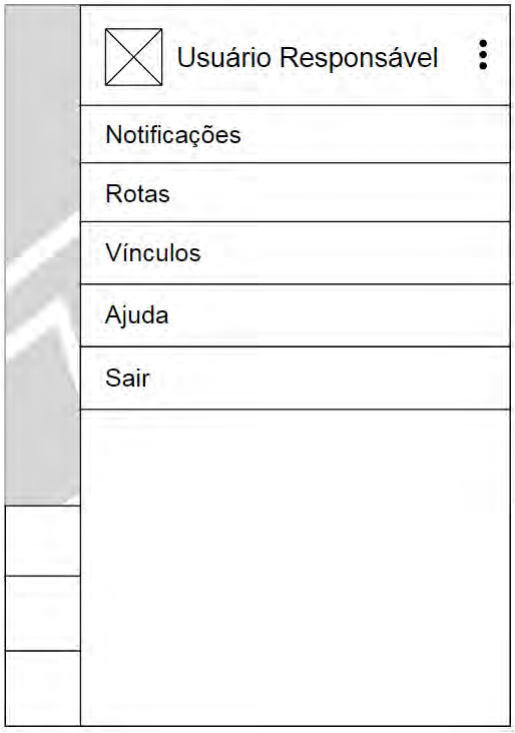
Cenários Alternativos:

Dados obrigatórios não preenchidos: Caso os campos [Dia], [Horário Inicial] e [Horário Final] não tenham sido preenchidos, o usuário será notificado através da mensagem "*Informe o dia e o horário para consulta".*

Esboços de Tela

Tela 07:

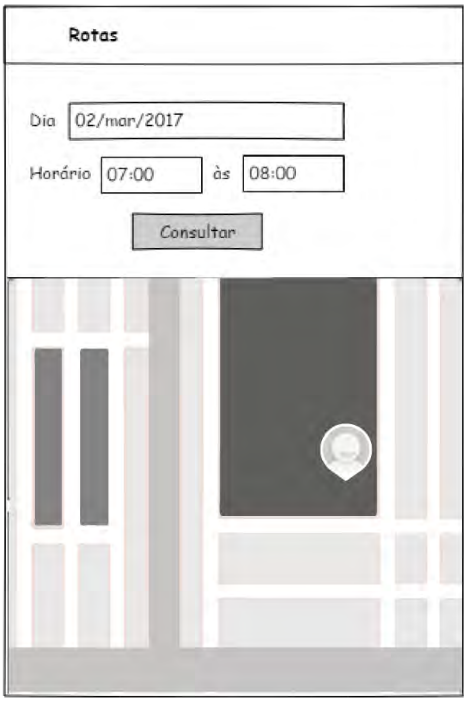
Figura 81 – Tela de opções do perfil Responsável



Fonte: próprio autor

Tela 08:

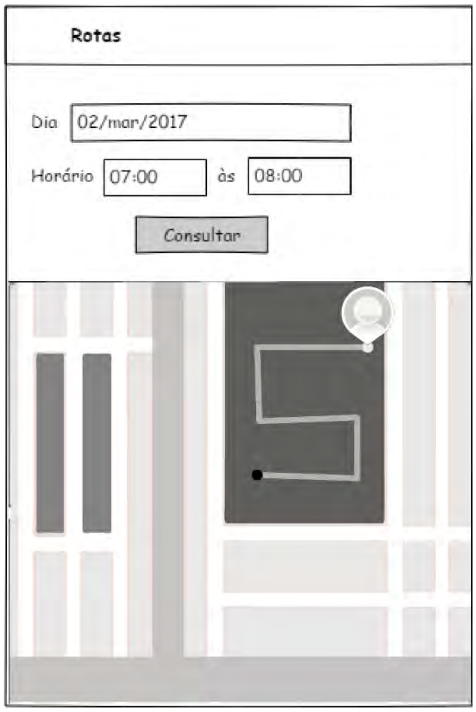
Figura 82 – Tela de consulta de rotas do perfil Aluno



Fonte: próprio autor

Tela 09:

Figura 83 – Tela de visualização de rota do perfil Aluno



Fonte: próprio autor

CDU 08 – Solicitar vínculo com o Perfil Aluno

Tipo: Abstrato

Objetivo: permitir que o usuário responsável adicione um vínculo com o Perfil Aluno.

Ator: Usuário utilizando perfil de Responsável.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): As funcionalidades de monitoramento do Perfil Aluno poderão ser habilitadas mediante a confirmação do vínculo pelo usuário Aluno.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 03)**.
2. Usuário toca no ícone de Mais Opções, no canto superior direito.
3. Aplicativo exibe a **Tela 07**
4. Usuário toca na opção [Vínculos].
5. Aplicativo exibe a **Tela 10**.
6. O usuário digita o [Login] do Perfil Aluno que deseja adicionar vínculo no campo [Pesquisa].
7. Toca no botão [Adicionar] referente ao Perfil Aluno que deseja adicionar vínculo dentre os resultados, conforme **Tela 10**.
8. Digita uma senha para confirmar o vínculo pelo Perfil Aluno que deseja adicionar e clica no botão [Enviar] na **Tela 11**.

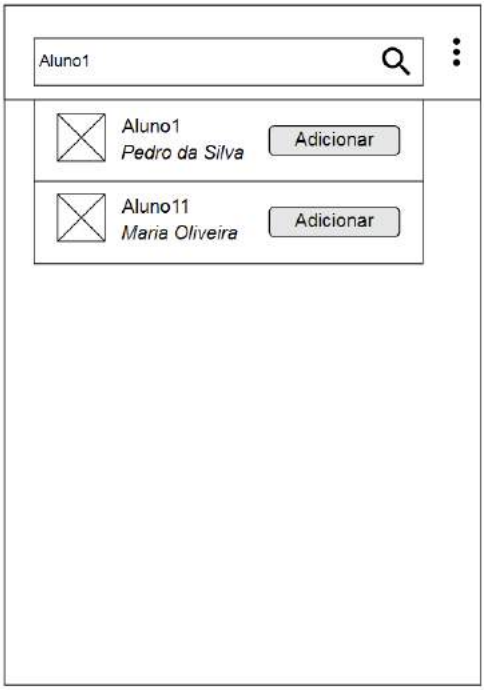
Cenários Alternativos:

Nenhum resultado encontrado para a busca: o aplicativo não encontra algum Perfil Aluno com o [Login] digitado no campo [Pesquisa], então exibirá a mensagem “*Nenhum resultado encontrado”.*

Esboços de Tela

Tela 10:

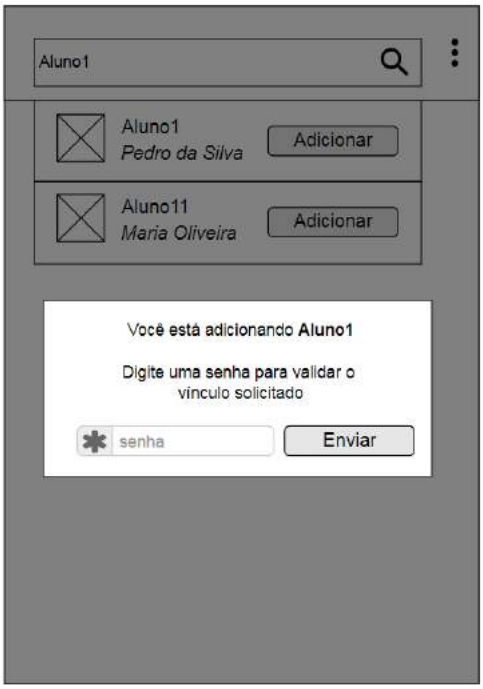
Figura 84 – Tela Gerenciar Vínculos



Fonte: próprio autor

Tela 11:

Figura 85 – Tela de cadastro de senha para adicionar vínculo



Fonte: próprio autor

CDU 09 – Visualizar notificações enviadas/recebidas

Tipo: Abstrato

Objetivo: permitir que o usuário Aluno/Responsável visualiza as notificações enviadas/recebidas pelo seu aplicativo.

Ator: Usuário utilizando perfil Aluno/Responsável.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): O usuário poderá visualizar as notificações enviadas/recebidas com as informações sobre o horário que Aluno chegou ou saiu da escola.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 03 / Tela 04)**.
2. Se estiver usando o Perfil Responsável, o usuário toca no ícone de Mais Opções, no canto superior direito e o aplicativo exibe a **Tela 07**
3. Usuário toca em [Notificações].
4. Aplicativo exibe **Tela 12** contendo as últimas notificações enviadas.
5. Usuário digita a [Data Inicial] e a [Data Final] e clica em [Consultar] para visualizar as notificações de um período específico.
6. Aplicativo exibe a **Tela 13** com as notificações enviadas dentro do período consultado.

Cenários Alternativos:

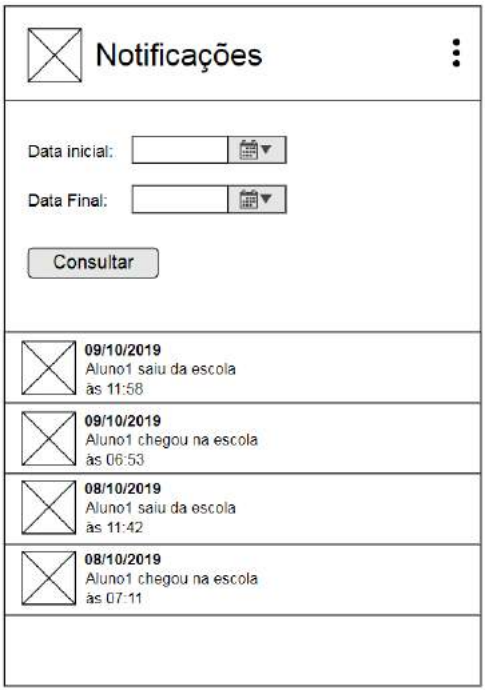
Dados obrigatórios não preenchidos: Caso o campo [Data Inicial] não tenha sido preenchido, o usuário será notificado através da mensagem "*Informe dia ou o período para consulta".*

Sem resultados para a consulta: Caso o aplicativo não encontre alguma notificação enviada dentro do período consultado, será exibido a mensagem "*Nenhuma notificação enviada dentro do período consultado".*

Esboços de Tela

Tela 12:

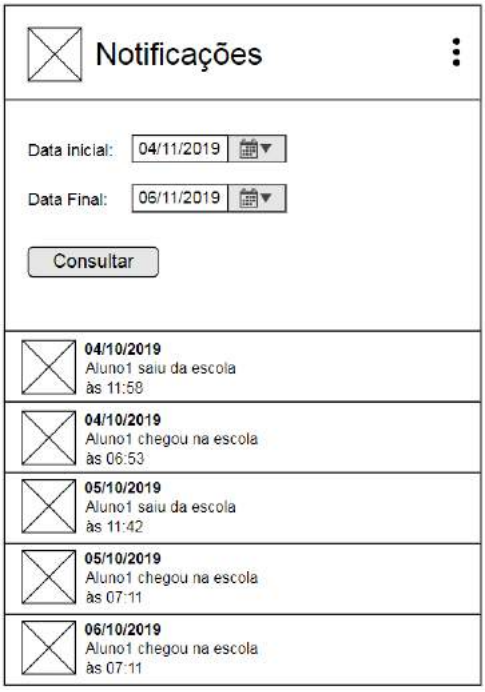
Figura 86 – Tela de notificações



Fonte: próprio autor

Tela 13:

Figura 87 – Tela de consulta de notificações



Fonte: próprio autor

CDU 10 – Aceitar Vínculo com Perfil Responsável

Tipo: Abstrato

Objetivo: validar o vínculo que usuário Responsável solicitou.

Ator: Usuário utilizando perfil Aluno.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): Validar o vínculo e habilitar as funcionalidades de monitoramento para o Perfil Responsável.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 04)**.
2. Usuário toca em [Vínculos].
3. Aplicativo exibe a **Tela 14.**
4. O usuário visualiza a solicitação enviada e clica no botão [Aceitar].
5. Digita senha cadastrada pelo Perfil Responsável para validar o vínculo entre os perfis e toca no botão [Confirmar] **(Tela 15)**.

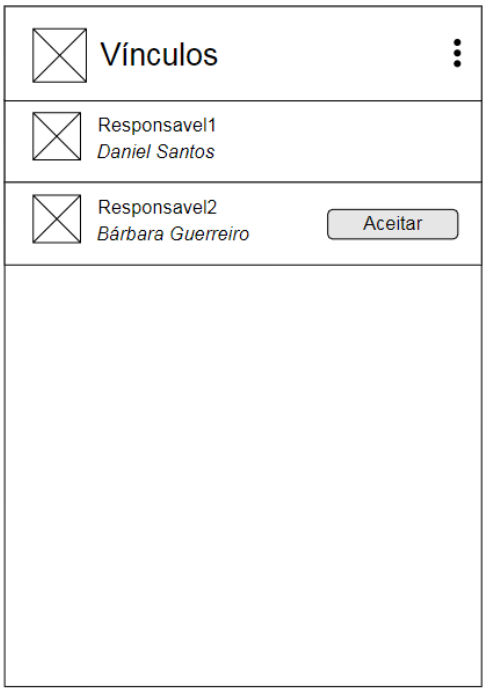
Cenários Alternativos:

Senha incorreta: o usuário Aluno digita uma senha diferente da cadastrado pelo usuário Responsável, então exibirá a mensagem "*Senha incorreta".*

Esboços de Tela

Tela 14:

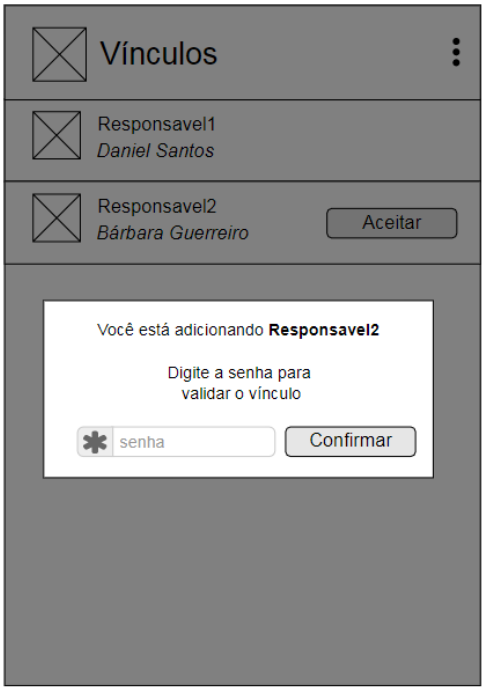
Figura 88 – Tela de vínculo do Perfil Aluno



Fonte: próprio autor

Tela 15:

Figura 89 – Tela de notificação



Fonte: próprio autor

CDU 11 – Configurar recursos de localização

Tipo: Abstrato

Objetivo: alterar as configurações de captura de informações geográficas.

Ator: Usuário utilizando perfil Aluno.

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): alterar o tempo em que o aplicativo captura as coordenadas geográficas do dispositivo através do Perfil Aluno.

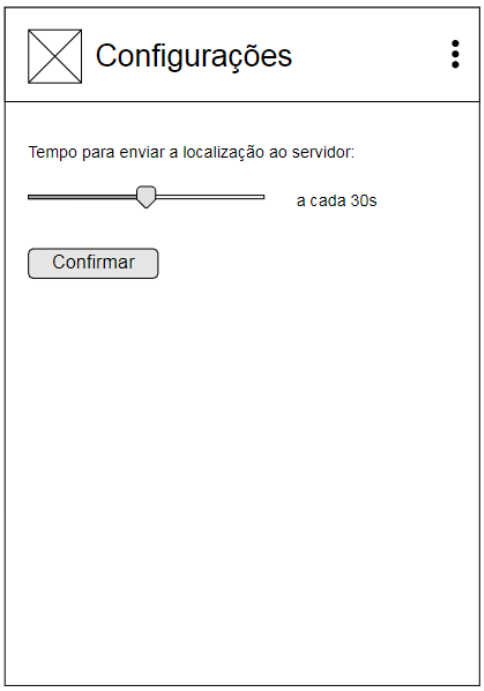
Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 04)**.
2. Usuário toca em [Configurações].
3. Aplicativo exibe a **Tela 16.**
4. O usuário altera o tempo de captura das informações geográficas.
5. Toca no botão [Confirmar].

Esboços de Tela

Tela 16:

Figura 90 – Tela de configurações do Perfil Aluno



Fonte: próprio autor

CDU 12 – Manter Perfil

Tipo: Abstrato

Objetivo: Alterar informações do perfil Responsável ou Aluno, validando os campos obrigatórios e valores inválidos.

Ator: Responsável ou aluno

Pré-Condições: Ter feito login no aplicativo.

Pós-condições (Garantia de Sucesso): Atualizar as informações no perfil do aplicativo.

Cenário Principal:

1. Aplicativo exibe a Tela Principal (**Tela 03 / Tela 04)**.
2. Se estiver usando o Perfil Responsável, o usuário toca no ícone de Mais Opções, no canto superior direito e o Aplicativo exibe a **Tela 07**
3. Usuário toca em cima do [login] ou da [foto].
4. Aplicativo exibe a **Tela 17**.
5. O usuário informa os novos dados [Login], [Senha], [Confirmar Senha], [Nome completo] e seleciona uma nova [Foto] para o perfil.
6. É realizada a validação das informações nos campos.
7. Aplicativo envia os dados para o sistema web para cadastro no banco de dados e mostra notificação "*Perfil atualizado com Sucesso!*"
8. Aplicativo exibe a **Tela 03 / Tela 04**.

Cenários Alternativos:

Campos Obrigatórios não preenchidos: Caso algum dos campos [Login] e [Nome Completo] não tenha sido preenchido, o usuário será notificado através da mensagem "*Campo obrigatório".*

Campos [Senha] e [Confirmar Senha] não coincidirem: será exibido mensagem *"Senhas não coincidem"*.

Senha não alterada: se o usuário deixar em branco os campos [Senha] e [Confirmar Senha], a senha atual será mantida.

Esboços de Tela

Tela 17:

Figura 91 – Tela para atualizar Perfil

Fonte: próprio autor

CDU 13 – Enviar coordenadas geográficas do dispositivo logado com o Perfil Aluno

Tipo: Abstrato

Objetivo: enviar as coordenadas geográficas do dispositivo de acordo com o período de tempo configurado no Perfil Aluno para o Servidor Web.

Ator: aplicativo utilizando perfil Aluno.

Pré-Condições: Ter feito login no aplicativo com o perfil Aluno pelo menos 1 vez.

Pós-condições (Garantia de Sucesso): enviar as coordenas geográficas para o Servidor Web para que este possa enviar notificação de entrada/saída ou sincronizar a posição atual dentro da área georreferenciada.

Cenário Principal:

1. Aplicativo captura as informações geográficas de acordo com o período pré-estabelecido no Perfil Aluno.
2. As informações são enviadas ao Servidor Web.

Cenários Alternativos:

Sem conexão com internet: o aplicativo enviará as coordenadas geográficas assim que recuperar a conexão com a internet.