PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

IMPLEMENTAÇÃO DE *BROADCAST E GEOFENCING* NO USO DE PROPAGANDA

HENRIQUE EMANOEL ALVES DE SOUSA

GOIÂNIA
2022

HENRIQUE EMANOEL ALVES DE SOUSA

IMPLEMENTAÇÃO DE *BROADCAST* E *GEOFENCING* NO USO DE PROPAGANDA

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de computação.

Orientador: Prof. Marcelo Antonio Adad de Araújo, M.E.E.

GOIÂNIA
2022

HENRIQUE EMANOEL ALVES DE SOUSA

IMPLEMENTAÇÃO DE *BROADCAST* E *GEOFENCING* NO USO DE PROPAGANDA

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em _____/_____/_____.

_____

Prof. Ma. Ludmilla Reis Pinheiro dos Santos

Coordenadora de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientador: Prof. Marcelo Antonio Adad de Araújo, M.E.E.

_____

Prof. Carlos Alexandre Ferreira de Lima, Me.

_____

Prof. Nilson Cardoso Amaral, Ph.D.

GOIÂNIA

2022

Dedico este trabalho a minha família e principalmente minha mãe que sempre me apoiou.

RESUMO

O trabalho a ser apresentado é a implementação de um aplicativo móvel para o sistema operacional *Android*, que utiliza da localização do usuário dentro de um perímetro virtual, chamado *geofence*, e registra todos que estão conectados à rede *Wi-Fi* para disseminação de propagandas utilizando *broadcast*, com as devidas permissões do usuário. Obtendo assim o endereço *Media Access Control* (MAC) do dispositivo. Para que isso seja possível, a utilização de *hardwares*, como roteadores, aliado ao *software*, foi necessária.

Palavras-Chave: Android; geofence; broadcast.

ABSTRACT

The work to be presented is the implementation of a mobile application for the Android operating system, which uses the user's location within a virtual perimeter, called geofence, and registers everyone who is connected to the Wi-Fi network for dissemination of advertisements using broadcast, with the proper permissions of the user. Thus, obtaining the Media Access Control (MAC) address of the device. For this to be possible, the use of hardware, such as routers, allied to software, will be necessary.

Keywords: Android; geofence; broadcast.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

Me(a)	Mestre(a)
M.E.E	Mestre em Engenharia Elétrica
Prof(a).	Professor(a)
Ph.D.	Doutorado
TCC	Trabalho de Conclusão de Curso
km	Quilômetros

LISTA DE SIGLAS

Wi-Fi	*Wireless Fidelity*
RFID	*Radio Frequency Identification*
GHz	*Giga-Hertz*
MHz	*Mega-Hertz*
CPU	Unidade Central de Processamento
GPS	Sistema de Posicionamento Global
API	*Interface de Programação de Aplicativos*
SSID	*Service Set Identifier*
MAC	*Media Access Control*
AI	Inteligência Artificial
RF	Rádio Frequência
RPB	Broadcast de rota inversa
IDE	*Integrated Development Environment*
IBGE	Instituto Brasileiro de Geografia e Estatística
IP	*Internet Protocol*
LGPD	Lei Geral de Proteção de Dados
SMS	*Short Message Service*
OSPF	*Open Shortest Path First*
SPAM	*Set of Guidelines for Mass Unsolicited Mailings and Postings*
IU	Interface do Usuário
TCC	Trabalho de Conclusão de Curso
TCC2	Trabalho de Conclusão de Curso para o segundo semester
SDK	*Software Development Kit*
APP	Aplicativos
API	*Application Programming Interface*

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho consta da utilização de *Broadcast*, ou seja, difusão de informações através do dispositivo mobile, com uso simultâneo de *Geofencing* para fins comerciais, de forma que o estabelecimento comercial possa usar esse canal como forma de divulgação para usuários que já estejam na base cadastral das lojas. (KEMMIS, 2020)

O *Broadcast* é um vocábulo em inglês que a tradução direta representa a junção de dois termos, *broad* (larga escala); *cast* (enviar, projetar, transmitir). Caracteriza-se pelo processo em que se transmite ou difunde determinada informação para muitos receptores e dispositivos ao mesmo tempo. Como exemplos de *Broadcast* são rádios, televisão, rádio navegação marítima, telefones sem fio e até *pagers.* (FOROUZAN, 2010).

O *Geofencing* tem como tradução aproximada *fence* (cerca); *geo* (geográfica). É uma tecnologia que tem como base a geolocalização, ou seja, a posição geográfica em que um dispositivo está inserido, estabelecendo um perímetro geográfico virtual, dentro do qual é possível configurar alertas ou comandos para os dispositivos que entrarem na área delimitada. Assim como no Broadcast, se faz uso de tecnologias como identificação de sistema de posicionamento global (GPS), identificação por radiofrequência (RFID), antenas de celulares e sinais de Wi-Fi. (WHITE, 2017; KEMMIS, 2020).

O objetivo de utilizar *Broadcast* e *Geofencing* é fornecer serviços e alertas sem interferir na privacidade do usuário, pois as mensagens são enviadas apenas para Smartphones dentro da área da cerca virtual, sendo possível oferecer uma solução que não afeta os consumidores com *Spam*, enviando apenas para usuários na localização geográfica específica, e com as devidas permissões aceitas pelo usuário (KEMMIS, 2020).

1.1 Objetivos

Os objetivos se dividem em objetivos gerais e objetivos específicos.

Objetivos gerais:

- Implementar um aplicativo em que toda área criada pelo *Geofencing* receba os alertas e mensagens por meio de *Broadcast*.
- Verificar tipos de conexão, exemplo Wi-Fi livre da loja, e aceitar o termo para utilização dos dados.
- Utilizar sinais *Broadcast* e a área virtual do *Geofencing,* sem interferir na privacidade do usuário, enviado alertas para pessoas apenas na área da cerca virtual.

Objetivos específicos:

- Identificar as áreas que englobam *Geofencing* e *Broadcast.*
- Conhecer ferramentas voltadas à *Geofencing* e *Broadcast*.
- Analisar e identificar dificuldades de aplicar *Geofencing* na área de marketing.
- Analisar as características, protocolos e leis que envolvem a privacidade do usuário em relação a *Spam*.
- Estudar métodos que auxiliem na Lei Geral de Proteção de Dados (LGPD) e as resoluções da Anatel.
- Identificar e analisar as fragilidades de segurança dos protocolos.
- Utilizar sistemas específicos para testes e implementação.
- Pesquisar ferramentas utilizadas com o mesmo objetivo do projeto.
- Estudar e apresentar os resultados obtidos.

1.2 Procedimentos metodológicos

Esta pesquisa segundo sua natureza é um resumo do assunto, pois busca sistematizar a área de conhecimento, indicando sua evolução, neste

caso, as áreas de conhecimentos de *Geofencing* e *Broadcast* (WAZLAWICK, 2014).

Observando o relacionamento do presente trabalho, pode-se induzir que esta pesquisa se traduz em exploratória e descritiva.

Segundo o pesquisador Wazlawick (2014), “Na pesquisa exploratória, o autor vai examinar um conjunto de fenômenos, buscando anomalias que não sejam ainda conhecidas e que possam ser, então, a base para uma pesquisa mais elaborada” (WAZLAWICK, 2014, p. 22).

Ainda de acordo com o pesquisador: “A pesquisa descritiva é caracterizada pelo levantamento de dados e pela aplicação de entrevistas e questionários. Assim como a pesquisa exploratória, ela pode ser considerada um passo prévio para encontrar fenômenos não explicados pelas teorias vigentes” (WAZLAWICK, 2014, p. 22).

Esta pesquisa é bibliográfica e experimental, e apresenta levantamento de dados, análise dos mesmos e uma busca pelas causas e explicações dos dados apresentados neste trabalho, de forma técnica.

Como um trabalho de pesquisa bibliográfica, este projeto é baseado em estudo de artigos, teses, livros, entrevistas e reportagens por editoras e emissores.

De acordo com Wazlawick (2014), esta pesquisa tem efeito de investigar os resultados obtidos e entender suas causas e explicações. Este projeto de pesquisa se propõe a abordar o tema de implementação de *broadcast* e *geofencing* no uso de propaganda, além de apresentar os levantamentos dos resultados encontrados sobre o assunto e fazer uma análise deles, e buscar explicações dos resultados apresentados neste trabalho.

Segundo estes objetivos, foi feito a pesquisa bibliográfica seguindo os seguintes passos:

a) Listados periódicos e eventos relevantes ao tema da pesquisa e periódicos gerais sobre computação, para verificar os artigos na área do tema que foi pesquisado.
b) Pesquisa dos artigos publicados a pelo menos cinco anos.
c) Selecionados da lista, títulos relacionados ao tema que foi pesquisado.

d) Feito leitura dos *abstracts* dos artigos selecionados, e feito a classificação em relevância “alta”, “média” ou “baixa”.
e) Leitura dos artigos com alta relevância, e feito resumos com os principais assuntos aprendidos sobre o tema. Anotados títulos que foram mencionados na bibliografia mesmo que tenham mais de cinco anos.
f) Foi necessário, ler artigos de relevância média ou baixa, porém sempre priorizando os artigos com alta relevância.
g) Necessidade de aumentar a pesquisa, com a leitura artigos mais antigos (Expandindo o passo b) ou periódicos com grau de relevância menor (Expandindo o passo a).

Para Wazlawick (2014), a pesquisa experimental se dá pela manipulação de uma parte da realidade do pesquisador, como por exemplo, esse trabalho, que visa mostrar o uso do *geofencing*.

Para Gil (2017), uma pesquisa experimental depende de um objeto de estudo, variáveis que podem manipulá-lo, formas de controle das variáveis e formas de observação dos efeitos que as variáveis produzem no objeto.

Gil (2017) define que para a realização de uma pesquisa experimental é necessário seguir os seguintes passos:

A) O problema dessa pesquisa é: como implementar *broadcast* e *geofencing* no uso de propaganda?
B) Quanto à definição do plano experimental, inicialmente os recursos envolvendo geofencing foram estudados.
C) Quanto ao ambiente experimental foi implementado no Android Studio da seguinte forma.
    - Foi utilizado as funções do Android Studio, que são relacionadas com *geofencing*, *broadcas* e entre outras.
D) A coleta de dados foi feita de forma bibliográfica buscando artigos, livros e documentos mais recentes para responder a seguinte questão: o que é *broadcast* e *geofencing*?
    - Quais as utilidades *geofencing* no uso de propaganda:

        Foi feito na prática o teste das principais utilidades do *geofencing* e *broadcast* para disseminar anúncios.

- Como minimizar os problemas com a Lei Geral de Proteção de Dados (LGPD):

  De acordo com a pesquisa bibliográfica e a parte prática que foi desenvolvido, foi apontado os pontos onde pode-se minimizar problemas em relação a LGPD.

E) A análise dos resultados que foi realizada através dos resultados obtidos pela coleta dele, do qual se procederá a análise desses fatos. De posse dessas análises os resultados são discutidos visando inferir conhecimentos relativos em relação as propagandas direcionadas, para todos que concordaram e estão na região do *geofencing*.

- Foi analisado todos os resultados obtidos a partir de testes com *broadcast*;

F) O trabalho foi registrado em forma de uma monografia de TCC (Trabalho de conclusão de curso).

1.3 Resultados esperados

Espera-se que os resultados dessa pesquisa possam contribuir para:

- Ampliar o debate sobre uso de *Geofencing* em outras áreas.
- Informar a sociedade e as empresas os benefícios de utilizar *Geofencing*.
- Apresentar os recursos do software e sua importância nas empresas.
- Compreender a importância do uso de dados dos clientes e de qualquer outro usuário.

1.4 O trabalho é desenvolvido da seguinte forma:

O capítulo 1 é a introdução do trabalho e descreve de forma sucinta como foi desenvolvida o mesmo.

Já o capítulo 2 trata das partes constituintes do trabalho e suas aplicações.

O capítulo 3 refere-se à explicação e informações relevante para solução do problema proposto.

No capítulo 4 é apresentada a implementação do aplicativo, cada tela e fragmentos feitos para a criação de *geofencing* pelo App.

O capítulo 5 refere-se à conclusão e perspectivas futuras sobre o presente TCC.

Nos apêndices encontra-se o programa desenvolvido, e no anexo 1 apresenta-se o termo de autorização de publicação de produção acadêmica.

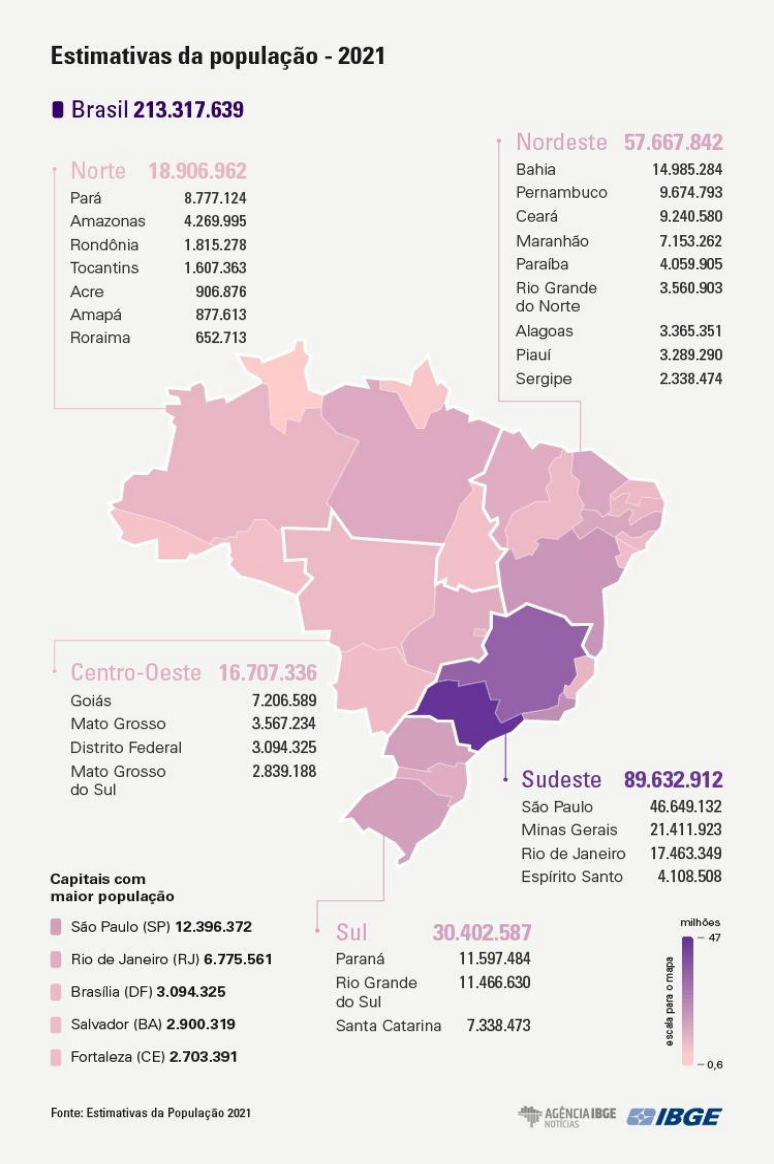
2 REFERENCIAL TEÓRICO

O celular é o dispositivo pessoal mais onipresente e fornece um meio eficiente para as empresas comercializarem seus serviços diretamente aos consumidores. Para se ter uma base, o Brasil possui mais de 2 dispositivos digitais para cada habitante. (Meirelles, F. S, 2021)

É estimado que a população brasileira chegou a 213,3 milhões de habitantes. Essa estimativa é do Instituto Brasileiro de Geografia e Estatística (IBGE) que foi realizada em 2021. E analisando a pesquisa feita pela Fundação Getúlio Vargas (FGV), que teve como objetivo pesquisar e estimar a quantidade de aparelhos e dispositivos digitais em 2021, se concluiu que há aproximadamente 440 milhões de dispositivos digitais (computador, notebook, tablet e smartphone) em uso no Brasil, sendo assim dois ou mais por habitante, com base na 32ª edição da Pesquisa Anual do FGVcia sobre o Mercado Brasileiro de TI e Uso nas Empresas, divulgada pelo Centro de Tecnologia de Informação Aplicada da Escola de Administração de Empresas de São Paulo da Fundação Getúlio Vargas (FGVcia). (Meirelles, F. S, 2021)

A Figura 1 apresenta a distribuição da população brasileira em 2021 segundo dados do IBGE. (IBGE, 2021)

Figura 1 – Estimativa da distribuição da População brasileira em 2021



Fonte: IBGE, 2021.

O marketing móvel tem se tornado cada vez mais popular e mais viável, e sua utilidade é vislumbrada para várias empresas e anunciantes pelo mundo, uma vez que os serviços podem ser associados com base em localização. No entanto, os serviços de marketing baseados em *Short Message Service* (SMS), que em português significa Serviço de Mensagens Curtas, ainda não tiveram seus problemas totalmente resolvidos sobre *Spam* (*Set of Guidelines for Mass Unsolicited Mailings and Postings)* sendo em português a tradução como: um conjunto de diretrizes para correspondências e postagens não solicitadas em massa, que em sua maioria exige a aprovação do usuário em receber tais

mensagens, levantando as preocupações sobre a privacidade. (KEMMIS, 2020)

As mensagens não devem ser invasivas, e quando devidamente configuradas podem aparecer na tela de descanso do celular do cliente ou qualquer outra pessoa transitando pelo local, que pode ou não interagir, recebendo notícias, informações, promoções, dentre outros (KEMMIS, 2020).

Levando em conta essa privacidade do usuário, é importante observar a legislação vigente, principalmente as resoluções da Anatel e a nova Lei Geral de Proteção de Dados (LGPD).

A resolução nº 632, de 7 de março de 2014 regulamenta o envio de mensagens de marketing determinando as normas ou condutas para o envio de mensagens publicitárias. A regra da Anatel determina que a prestadora só pode mandar suas mensagens de cunho publicitário para o consumidor se esse consumidor permitir. Além de que, os contratos devem possuir uma cláusula permitindo que o usuário assinale se deseja, ou não, receber mensagens publicitárias da empresa. Este tipo de medida não se aplica caso a seja de cunho informativo. As mensagens publicitárias de terceiros não são regulamentadas pela Anatel. (Art. 76 da Resolução nº 632/2014 da Anatel)

A resolução nº 671, de 3 de novembro de 2016 em seu artigo 15 diz que “o uso de radiofrequências, tendo ou não caráter de exclusividade, dependerá de prévia outorga da Anatel, mediante autorização.” (Art. 15 da Resolução nº 637/2016 da Anatel)

Já a nova Lei Geral de Proteção de Dados (LGPD), em vigor desde agosto de 2021, tem como objetivo padronizar normas e garantir a privacidade dos dados dos usuários envolvidos. O regulamento estabelece definições sobre os usos e políticas que as organizações podem fazer com os dados obtidos. O consentimento tem que ser explícito pelo usuário que fornece os dados a alguma companhia e a possibilidade de que essa pessoa solicite a exclusão dos dados, são parte dos fatores que passaram a valer a partir dessa legislação (LEI Nº 13.709, DE 14 DE AGOSTO DE 2018).

Sendo assim, as políticas de privacidade em relação a estes alertas e notificações aos consumidores, viriam com termos de aceitação, para clientes que já possuam acesso a redes *Wireless,* ou redes sem fio do local designado.

Com as devidas permissões aceitas, os alertas *Geofencing* sempre ativariam para os clientes nas posições da cerca geográfica (KEMMIS, 2020).

Para que os alertas sejam disseminados, é necessário identificar os usuários por seus dispositivos, com sua identificação dentro da rede Wi-Fi, como MAC, IP, e os tipos de alertas que serão transmitidos por essas redes. (KEMMIS, 2020).

2.1 Redes sem fio

Quando se fala de redes sem fio, pensa-se logo em mobilidade. Existem, porém, diversos ambientes de redes nos quais apenas os nós da rede são sem fio, mas não móveis, como por exemplo, redes residências sem fio ou redes de escritórios onde se evitou o custo da instalação do cabeamento (FILHO, 2016).

Nas conexões Wi-Fi, é possível ter um identificador que se chama SSID, *Service Set Identifier*, ou traduzido para o português, Identificador de Conjunto de Serviços. Os SSIDs são utilizados para nomear as diversas redes sem fio espalhada em um local, e caso o celular *Android* esteja com Wi-Fi ligado sem conectar em nem uma rede, ainda é possível identificar a localização dos aparelhos, de acordo com o sinal de diversos roteadores, estando com o GPS ativo em segundo plano (KOVACS, 2021).

2.2 GPS

Com a evolução da tecnologia, se tornou muito mais fácil obter quaisquer dispositivos eletrônicos, como tablets, smartphones e notebooks. Atualmente todos esses aparelhos possuem uma tecnologia que possibilita identificar a sua localização que é chamada de *GPS* (*Global Positioning System)*, ou em português Sistema de Posicionamento Global (MACHADO, 2015).

O GPS está em funcionamento desde 1995, com lançamento de 24 satélites para órbita da terra, sendo desenvolvido pelo Departamento de Defesa dos Estados Unidos com o projeto chamado de “NAVSTAR”. Atualmente o projeto conta com total de 32 satélites com a órbita de

aproximadamente duas voltas por dia na terra, sendo 8 deles desativados para servirem como reserva, caso algum dos 24 satélites falhe (FAGGIAN, 2019).

A Figura 2 apresenta os planos orbitais em que se encontram os satélites ao redor da terra.

Figura 2 - Disposição dos satélites de GPS

Fonte: MACHADO, 2015.

Para que o GPS tenha a localização aproximada, ou até mesmo exata de algum ponto ou receptor, é necessário que os satélites emitem sinais constantes e ter como base a hora do envio do sinal. O aparelho calcula a distância que o receptor está do satélite, verificando o cálculo realizado por no mínimo 4 satélites. Através do processo de triangulação o sistema identifica um ponto comum entre as distâncias e os satélites utilizados. (MACHADO, 2015).

A geolocalização de aparelhos celulares através do GPS, principalmente com sistema operacional Android, é mais eficiente na identificação da localização, independente das conexões de rede, como em lugares afastados, com pouco sinal, distantes das cidades, contanto que seja uma área aberta será localizado pelo GPS (MACHADO, 2015).

2.3 Geofencing

O *geofencing* é uma tecnologia que utiliza a localização geoposicionada, a partir da posição dos dispositivos, como GPS, Wi-Fi e outras tecnologias,

estabelecendo um perímetro geográfico virtual, alertando dispositivos que entrarem na área delimitada, chamada de *geofence* (KEMMIS, 2020).

A partir de diversas configurações do *geofence*, pode-se solicitar o envio de alertas ou ainda alterar configurações do dispositivo, verificar usuários, ou permitir o monitoramento das áreas entrando ou saindo delas (KEMMIS, 2020).

O perímetro geográfico criado, ou a cerca, pode ser configurado em diversos aparelhos móveis, tablets ou até computadores, em qualquer local, como shoppings, que possuem muitas lojas e restaurantes com um fluxo grande de pessoas. E não se restringe a apenas isso, já que é visto em indústrias, na pecuária, agronegócio e até mesmo os drones se utilizam desse recurso (WHITE, 2017).

Das diversas áreas que são aplicadas o *geofencing*, se destacam:

- Marketing: o *geofencing* é uma maneira fácil de entregar promoções. A partir do momento que o cliente entrar na área das lojas, irá receber essa propaganda. Ajudando também a manter o público mais propício a receber esse tipo de promoção.
- Participação da população: *Geofencing* é utilizado em grande concentração de pessoas em eventos. Eventos como festas, festivais de *shows*, feirões. Utilizando da geocerca em locais que sinalizem postagem de mídias, informações dos eventos ou até mapas das regiões.
- Dispositivos inteligentes: com a evolução dos aparelhos, com tecnologias de Bluetooth e Wi-Fi. É fácil, usando uma geocerca, difundir as notificações ou alertas de coisas que o usuário precisa, como passar no mercado para comprar comida, ou notificar as lojas de ferragens próximas, caso não lembre de comprar as ferramentas.
- Recursos humanos: O monitoramento dos funcionários dentro da empresa pode ser feito por *geofencing*, identificando com facilidades os horários de entrada e saída dos mesmos.
- Segurança: Geofencing na segurança, pode ser usado para alertar invasões, furtos de dispositivos configurados, entradas e saídas não identificadas, entre outras (WIEXP, 2018).

Geofences não necessariamente possuem algo físico. As áreas definidas por aplicativo em um mapa, e essas cercas virtuais podem ser ativas ou passivas. As ativas utilizam o aplicativo habilitado, e os serviços de GPS, obtendo com precisão a localização, e destacando se os dispositivos são localizados na área demarcada (STATLER, 2016).

Com os dados enviados para a central, e com as funções configuradas, esta implementação possui grandes vantagens, como a rápida entrega do processamento de dados para a central, economizando no gasto de bateria, tirando a necessidade de o aplicativo criar geofences despretensiosamente. Em termos de precisão, o tipo ativo, tem maior vantagem que o tipo passivo, permitindo criar pequenas áreas de cerca, de apenas 10 metros de diâmetro (STATLER, 2016).

As *geofences* passivas estão sempre ativas, funcionando em segundo plano, mas sua criação possui menos precisão, de até 100 metros de diâmetro, se utilizando menos do GPS que tem um custo alto de bateria, para utilizar dos dados móveis ou Wi-Fi (STATLER, 2016).

Os aparelhos estão sempre se comunicando com as torres de celulares, e cada torre possui sua localização fixa já conhecida. A passiva é menos vantajosa que a geofence ativa, mas com a identificação das torres, é possível localizar o usuário, além da combinação com Wi-Fi ou Bluetooth, permitido ter referências da atual localização (STATLER, 2016).

2.4 Endereços MAC, IPv4 e IPv6

Primeiramente IP é a sigla de *Internet Protocol* ou, em português, Protocolo de Internet, e MAC é *Media Access Control*, em português, Controle de Acesso de Mídia. Tanto o endereço MAC como o IP trabalham em conjunto para que seja identificado um dispositivo, e como a conexão de internet é estabelecida, ou as especificações e seus endereços na rede (FILHO, 2016).

Estas funções permitem que as informações sejam enviadas entre os dispositivos pela internet, pois os dados que referem ao IP, possuem o endereço que cada dispositivo que está conectado na rede, como smatphones,

computadores, tablets. Além de que o IPv4 e IPv6 são versões do sistema de IP (FERRAUDO, 2020).

2.4.1 Endereços MAC

O endereço MAC de um host não tem alteração, ele é atribuído fisicamente à placa de rede do host, conhecido como endereço físico. Este endereço físico permanece igual, independentemente de onde o host esteja alocado, e só existe um único endereço para cada dispositivo, porém alguns aparelhos mais atuais, possuem a função de ter um MAC aleatório (FILHO, 2016).

2.4.2 Endereço IPv4

O IPv4, ou Protocolo de Internet versão 4, possui endereços no padrão 32 bits, e possui cerca de 4 bilhões de combinações de endereços IP em todo o mundo. (FERRAUDO, 2020).

Um endereço IPv4 de gateway padrão é necessário para acessar redes remotas e os endereços IPv4 do servidor DNS (Sistema de nome de domínio) são necessários para converter nomes de domínio em endereços IPv4 (CCNA, 2019).

É utilizada normalmente a máscara de sub-rede IPv4 para distinguir a parte da rede da parte do host de um endereço IPv4. Caso um endereço IPv4 seja atribuído a um dispositivo, é usada a máscara de sub-rede para determinar o endereço de rede do aparelho. O endereço de rede se refere a todos os dispositivos na mesma rede (CCNA, 2019).

2.4.3 Endereço IPv6

O IPv6 é nada mais que a versão 6 do Protocolo de Internet, com endereços no padrão 128 bits, se tratando assim de uma versão melhorada do IPv4, já que o antigo protocolo não suporta completamente a demanda de todos os endereços. Com atuação em 128 bits, o IPv6 tem suporte para cerca

de 340 undecilhões de endereços, contra 4 bilhões que IPv4 suporta (FERRAUDO, 2020).

2.5 Unicast, Broadcast e Multicast

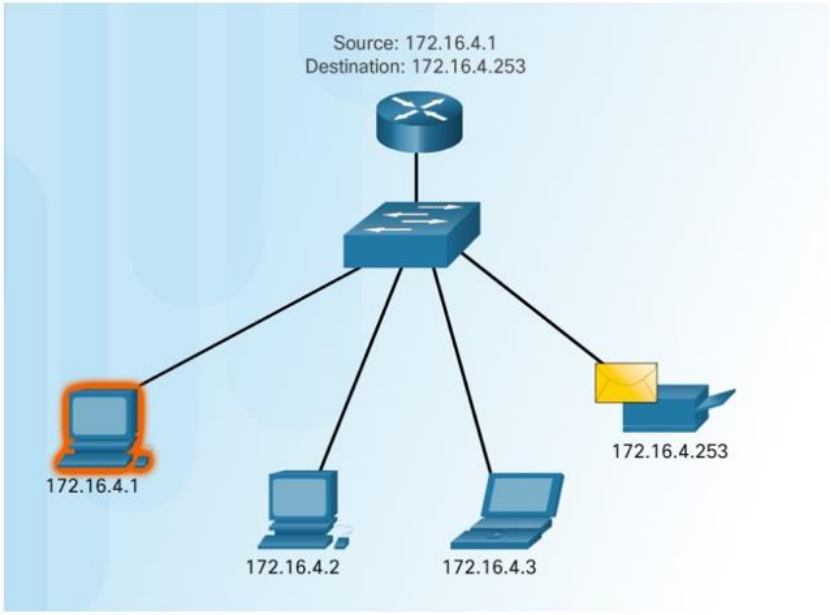
Nas tecnologias de rede, as mensagens, notificações e dados podem ser difundidas para máquinas ou dispositivos conectados a essa rede, por Unicast ou Broadcast ou Multicast. Os três tipos são usados em situações diversas, os quais serão apresentados nos tópicos seguintes (PINTO, 2018).

2.5.1. Unicast

A transmissão Unicast se trata do envio de mensagem de algum dispositivo para outro, em uma comunicação um a um, ou *point-to-point comunication*, em português, protocolo de ponto a ponto (CCNA, 2019).

O pacote unicast tem um endereço IP com destino único na origem, sendo enviado para um único destinatário. Este endereço IP que foi a origem só poderá ser unicast, pois o pacote só pode se originar de um único local. E não depende que o endereço IP de destino seja unicast, multicast ou broadcast. Como apresentado na Figura 3 (CCNA, 2019).

Figura 3 – Transmissão Unicast

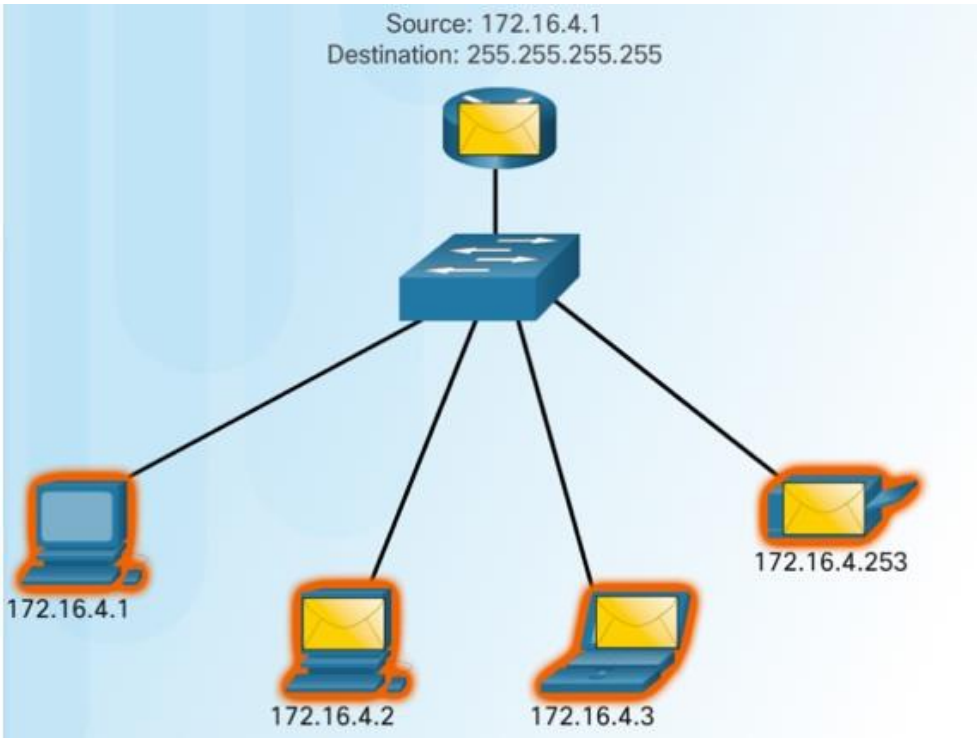


Fonte: PINTO, 2018.

2.5.2 Broadcast

As comunicações de um para todos é chamada de broadcast, que se refere a um aparelho que envia mensagens a todos em uma rede. Como apresentado na Figura 4, este pacote é enviado da máquina com o endereço 172.16.4.1 recebido por todas as máquinas que estão na rede (PINTO, 2019).

Figura 4 – Transmissão Broadcast



Fonte: PINTO, 2018.

O pacote de broadcast deve ser processado por todos os aparelhos no domínio. E o domínio identifica todos os hosts na mesma rede. Além de que essa transmissão pode ser limitada ou direcionada. A transmissão dirigida, é aquela enviada a todos os hosts em uma rede. Como é padronizado de fábrica, todos os roteadores não encaminham broadcasts. (CCNA,2019).

O broadcast de rota inversa (RPB) cria uma árvore de broadcast de rota mais curta desde a origem até cada destino. Ela garante que cada destino receba somente uma cópia do pacote (FOROUZAN, 2010).

Há sempre um endereço de difusão IPv4 para cada rede, também para o endereço de difusão 255.255.255.255.255, chamado endereço de broadcast

encaminhado. Este endereço utiliza o mais alto endereço da rede, que é onde todos os bits do anfitrião são 1s. A partir de que o endereço de broadcast dirigido para 192.168.1.0/24 é 192.168.1.255. E o endereço permite a comunicação com todos os hosts ou anfitriões dessa rede. Para enviar dados a todos os hosts de uma rede, pode-se enviar um único pacote que é endereçado para o endereço de broadcast da rede. IPv4 utiliza pacotes de difusão, mas não há pacotes de broadcast com IPv6 (ODOM, 2019).

Diferente do IPv4, o IPv6 não possui um endereço de broadcast. Porém, há um endereço multicast de todos os nós IPv6 que possui o mesmo resultado (CCNA, 2019).

Um dispositivo que não está ligado à rede de destino encaminha uma emissão dirigida a IP da mesma forma que encaminha pacotes IP unicast destinados a um host ou anfitrião nessa rede. Quando um pacote de difusão dirigida chega a um roteador ligado à rede de destino, esse pacote é difundido na rede de destino (ODOM, 2019).

2.5.3 Multicast

A transmissão multicast reduz o tráfego ao permitir que um anfitrião envie um único pacote para um conjunto selecionado de host que subscreva um grupo multicast (CCNA, 2019).

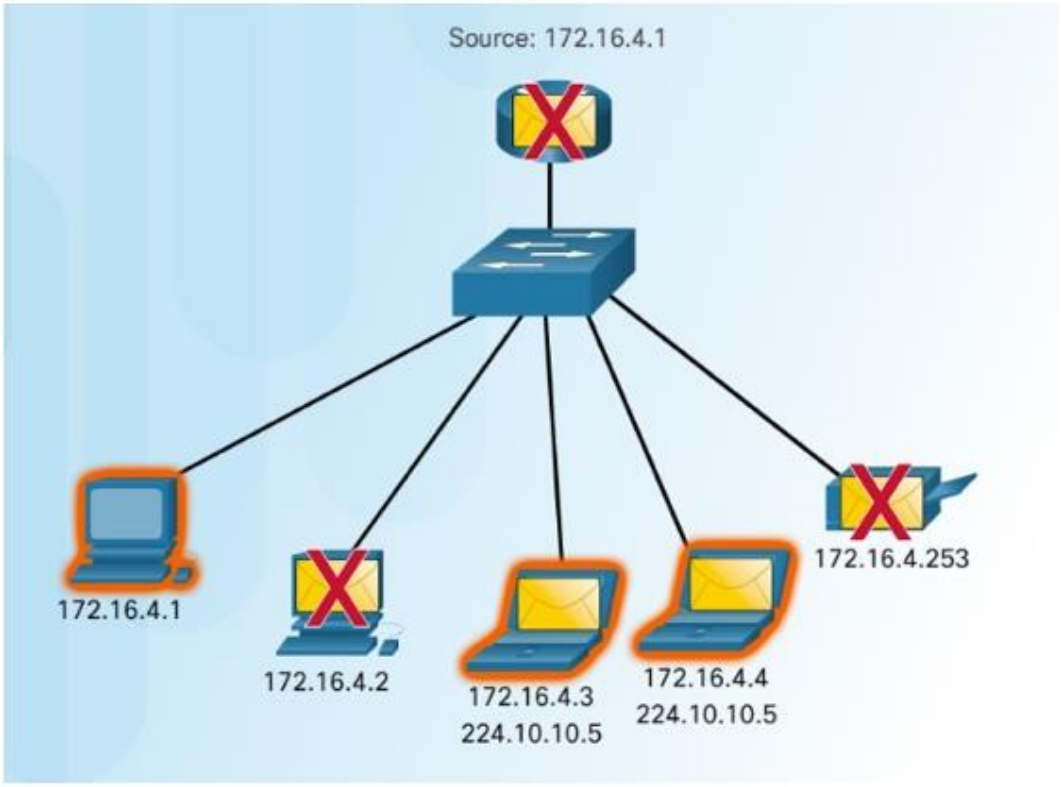
Cada pacote multicast é representado por um pacote com o endereço IP de destino, sendo um endereço multicast. O IPv4 reserva os endereços como um intervalo multicast. Os anfitriões que recebem pacotes multicast específicos são chamados clientes multicast e utilizam-nos como serviços solicitados por um programa cliente, para subscrever o grupo multicast (CCNA, 2019).

Para cada grupo multicast é caracterizado por um único endereço de destino IPv4 multicast. Quando um anfitrião IPv4 subscreve um grupo multicast, o anfitrião processa pacotes endereçados a esse endereço multicast e pacotes endereçados apenas ao seu endereço unicast atribuído (CCNA, 2019).

Os protocolos de encaminhamento, tais como OSPF (*Open Shortest Path First*), em português, abrir primeiro o caminho mais curto, utilizam

transmissões multicast. Os roteadores com OSPF ativado, se comunicam entre si utilizando o endereço OSPF já reservados. Apenas os dispositivos ativados ou habilitados para OSPF, podem processar estes pacotes como o endereço IPv4 de destino. Todos os outros dispositivos irão ignorar estes pacotes. Como ilustrado na Figura 5 (CCNA,2019).

Figura 5 – Transmissão Multicast



Fonte: PINTO, 2018.

3 SOLUÇÃO DO PROBLEMA

Este capítulo apresenta a solução do problema, com introdução aos componentes de software, para a prototipação das novas funcionalidades a serem implementadas ao aplicativo, utilizando Android Studio.

3.1 Android Studio (IDE)

O Android *Studio* é um IDE ou (*Integrated Development Environment*), com tradução português, ambiente de desenvolvimento integrado, com objetivo de desenvolvimento de aplicativos Android, com editor de código, ferramentas, e diversos recursos, que são baseados em outro IDE chamado *IntelliJ* (DEVELOPERS, 2021).

Dos recursos que possui, se destaca seu sistema baseado em *Gradle*, emulação de ótimo desempenho, um ambiente com a possibilidade de desenvolver para qualquer aparelho Android, alterar códigos do aplicativo em execução, comunicação com o GitHub para importação e exportação de códigos, compatibilidade com C++, entre outros recursos (DEVELOPERS, 2021).

O Android Studio possui uma estrutura dividida em módulos com visualização rápida e organizada, como bibliotecas, aplicativos Android e Google App *Engine*, que é uma plataforma de computação em nuvem. Os arquivos podem ser vistos em *Gradle Scripts*, e cada módulo contém as pastas de *Manifests* contendo o arquivo *AndroidManifest.xml*, a pasta Java, que contém os arquivos de código-fonte e, por fim, a pasta de Recursos, contendo partes do programa que não são código, como interfaces ou imagens (DEVELOPERS, 2021).

Como dito anteriormente, o Android *Studio* tem seu sistema de compilação baseado em *Gradle*, além de outros recursos do Android. Essa ferramenta é integrada no seu menu, bem maleável, fazendo alterações nos arquivos sem alterar o aplicativo original (DEVELOPERS, 2021). Para este trabalho foi utilizada a linguagem de programação Kotlin no Android Studio, apresentado a seguir.

3.2 Kotlin

O Android Studio tem compatibilidade perfeita com o Kotlin, assim pode se criar projetos com arquivos Kotlin, além de poder converter códigos Kotlin em Java e vice e versa (DEVELOPERS, 2020). A escolha feita de utilizar esta linguagem no desenvolvimento do App, foi baseada nas vantagens que ela oferece.

A partir do planejamento feito durante o trabalho, e das linguagens que poderiam ser usadas, o Kotlin seria uma melhor escolha, por ser uma linguagem nova e derivada do Java, e pelo desafio de aprimorar os conhecimentos em Android Studio e programação. Diante desses fatores, foi pensada essa linguagem como um apoio ou base, que se baseia na inovação e no legado do Java ao mesmo tempo.

O Kotlin fornece interoperabilidade com a linguagem Java, logo, a chamada de APIs do Android é praticamente igual ao código em Java correspondente. Uma das diferenças é que se pode combinar essas chamadas de método, aos recursos de sintaxe do Kotlin (DEVELOPERS, 2020).

A escrita do código em Kotlin, por ser parecida com Java, é de fácil compreensão, se tornando uma ótima escolha para a criação de aplicativos móveis. Pois é uma linguagem multiparadigma, que significa, que ela é compatível com a Orientação a Objetos e Paradigma Funcional de Programação (OLIVEIRA, 2019).

3.3 Atividades

Uma *Activity* ou Atividade no Android Studio, é uma ferramenta única e focada em atividades que o usuário pode realizar. Em sua maioria as atividades interagem com o usuário, portanto a classe *Activity* se encarrega de criar uma janela para qual se pode colocar a IU, Interface do Usuário, com *setContentView(View),* definirExibiçãodeConteúdo() (DEVELOPERS, 2022).

Normalmente são apresentadas ao usuário como janelas em tela cheia, além de poderem ser utilizadas de outras maneiras, como janelas flutuantes, no modo Multi-Janela ou incorporadas a outras janelas. Dos vários métodos

que podem ser programados, dois estão em quase todas as subclasses de *Activity*: (DEVELOPERS, 2022)

- *onCreate(Bundle)*: local onde se inicializa a atividade. A mais importante normalmente se chamará *setContentView(int)* com um recurso de layout definindo na IU e usando *findViewById(int)* para recuperar os *widgets* nessa IU com que se precisa interagir programaticamente.
- *onPause()*: local onde se lida com o usuário, pausando a interação ativa com a atividade. Qualquer alteração feita pelo usuário deve ser confirmada neste ponto.

Para que seja utilizado com *Context.startActivity()*, todas as classes de *activity* devem ter uma <*activity*> com declaração correspondente em seu pacote *AndroidManifest.xml*. (DEVELOPERS, 2022)

3.4 Fragmentos

Um *Fragment* ou Fragmento no Android Studio, representa uma parte de um comportamento ou completamente, apresentado na interface do usuário em um *FragmentActivity*. Existe a possibilidade de combinar vários fragmentos em uma única atividade, para que seja criado uma interface do utilizador de várias telas e reutilizar um fragmento em outras diversas atividades. É um tipo de “subatividade”, podendo utilizar os fragmentos como uma seção modular de uma ou mais atividades, com seu próprio ciclo de vida, recebendo os próprios eventos de entrada, e ser removida ou adicionada durante a execução da atividade. (DEVELOPERS, 2021)

O ciclo de vida do fragmento é diretamente impactado pelo ciclo de vida da atividade, pois ele deve sempre ser hospedado em uma atividade. Por exemplo, quando a atividade é destruída, todos os fragmentos também são, e quando a atividade é iniciada, os fragmentos também são, além de que é possível processar cada fragmento independentemente, enquanto uma atividade estiver em execução. (DEVELOPERS, 2021)

Ao ser adicionado um fragmento como parte do *layout* da atividade, ele se encontrará em um *ViewGroup,* o que significa estar dentro da hierarquia de visualizações, e esses fragmentos definirão os próprios *layouts* de exibição. Para inserir um fragmento no *layout*, deve-se declarar no arquivo de *layout* da *activity* como um elemento <*fragment*>, ou no código do App. (DEVELOPERS, 2021)

Este trabalho também descreve no capítulo 4 como foi feita a divisão dos fragmentos para compor uma única atividade, e compilar o aplicativo com base nas informações apresentadas.

3.5 SDK do *Places* para Android

O SDK (*Kit de desenvolvimento de software)* do *Places* (Lugares) para Android permite criar Apps com reconhecimento de local, que correspondem as empresas e lugares próximos ao aparelho do usuário. Significa que se pode criar Apps avançados com base em vários lugares, para complementar os serviços baseados em localização geográficas, que são oferecidos pelos serviços de localização do Android (DEVELOPERS,2022).

As duas partes a seguir oferecem os principais pontos de entrada:

Places seria o acesso ao banco de dados do Google, informações de locais e de empresas, além do local atual do aparelho.

Autocomplete concede *widgets* predefinidos, que retornam previsões de lugares em que o usuário começa a pesquisar.

A definição de lugar ou local é como um espaço físico com um nome. Também é visto como qualquer conhecimento que se encontra em um mapa. A interface *Place* (Lugar), na API, seria um lugar que é representado pelas informações, como nomes e endereços dos lugares, ID (identidade) desses lugares, telefone, localização geográfica, tipo de lugar, até mesmo links de sites e entre outros (DEVELOPERS,2022).

Todo momento em que o App traz ou exibe informações sobre lugares a partir do SDK do *Places* para Android, ele também apresentará todas as atribuições importantes repassadas pela API. (DEVELOPERS, 2022)

3.5.1 Local atual

O local atual pode ser retornado com o SDK do *Places* para Android, como verificar o local informado pelo aparelho celular no mesmo momento, além de lugares próximos e localizações geográficas, até mesmo empresas. (DEVELOPERS, 2022)

Se o aplicativo utilizar o *PlacesClient.findCurrentPlace*() (Locais Cliente encontra local atual), é necessário solicitar permissões de localização no momento da sua execução. É fundamental adicionar no manifesto a permissão "*android.permission.ACCESS_FINE_LOCATION*" para funcionar corretamente. (DEVELOPERS, 2022)

Para que seja verificado se o usuário concedeu a permissão, como a de acessar a localização do dispositivo, é necessário chamar a função *ContextCompat.checkSelfPermission* (Auto permissão de verificação de compatibilidade de contexto). Também é preciso que no App tenha a inclusão do código para solicitar a permissão do usuário, e processar o resultado devolvido (DEVELOPERS, 2022).

3.5.2 IDs de local

Os IDs de local, ou identidade do local se identificam de forma única dentro do banco de dados do Google *Places* e no Google *Maps*. Esses IDs de lugar são aceitos em solicitações para diversas APIs do *Maps*, como API *Geocoding* (Geocodificação), API *Directions* (Direções), *Place Details* (Detalhes do lugar) entre outras, além de como encontrar e desenvolver polígonos de limite no estilo aprimorado com dados. (DEVELOPERS, 2022)

Um ID de local é um identificador único textual, que nomeia e identifica um local de forma exclusiva. O tamanho do identificador do local varia, pois não há um tamanho máximo definido. (DEVELOPERS, 2022)

Os IDs de lugar estão acessíveis para a maioria dos locais, incluindo empresas, pontos turísticos e parques. É possível que o mesmo local tenha vários IDs de lugar, e que também podem mudar ao longo do tempo (DEVELOPERS, 2022).

No SDK do *Places* para Android, é possível recuperar o ID de um lugar chamando o comando *Place.getId*(). O serviço *Place* Autocomplete também retorna um ID de lugar, para cada lugar que corresponde à consulta de pesquisa e aos tipos fornecidos. Usando o ID do lugar para recuperar o objeto *Place* novamente, posteriormente. (DEVELOPERS, 2022)

Para receber o identificador de um lugar, chama-se *PlacesClient*.*fetchPlace*() (Local de busca do cliente do Google *Places*), enviando um *FetchPlaceRequest* (Buscar resposta do lugar), que correspondente ao ID de lugar fornecido. (DEVELOPERS, 2022)

O código de status *INVALID_REQUEST* (PEDIDO INVÁLIDO) indica que o ID do lugar especificado não é válido, e poderá ser retornado quando o ID de lugar tiver sido trocado ou modificado e não estiver mais correto (DEVELOPERS, 2022).

O código de status *NOT_FOUND* (NÃO ENCONTRADO) indica que o ID de lugar especificado está obsoleto, ele poderá se tornar obsoleto se uma empresa fechar ou mudar de endereço. Nesses tipos de casos, um lugar pode receber um novo ID de lugar, e o ID antigo retorna uma resposta *NOT_FOUND*. (DEVELOPERS, 2022)

3.6 *Place* Autocomplete

O serviço de preenchimento automático, ou *Place* Autocomplete, retorna previsões de lugares de maneira aproximada, em resposta a consultas de pesquisa do usuário. Conforme o usuário digita, o serviço de preenchimento automático retorna essas sugestões de lugares, como empresas, endereços e pontos de interesse. (DEVELOPERS, 2022)

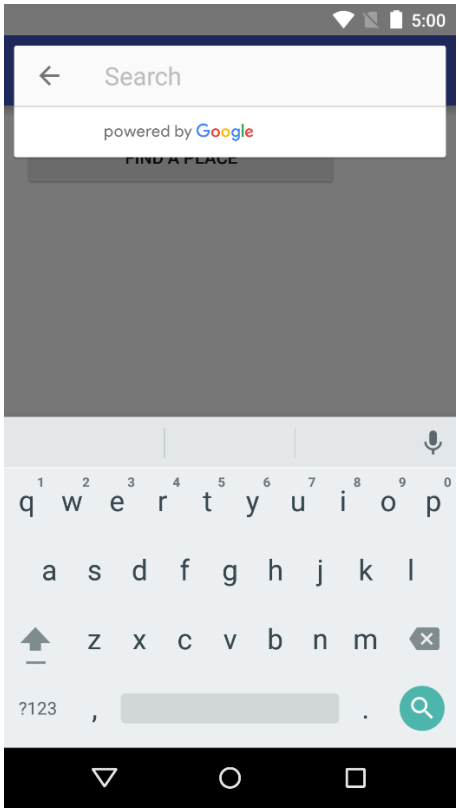
Para adicionar no aplicativo o preenchimento automático, foi apresentado as seguintes formas:

Adicionando um *widget*, ou ferramenta, de preenchimento automático no App, para economizar tempo de desenvolvimento e execução, garantindo um estudo consistente aos usuários.

Obter previsões de lugar de maneira precisa, para criar uma experiência personalizada aos usuários.

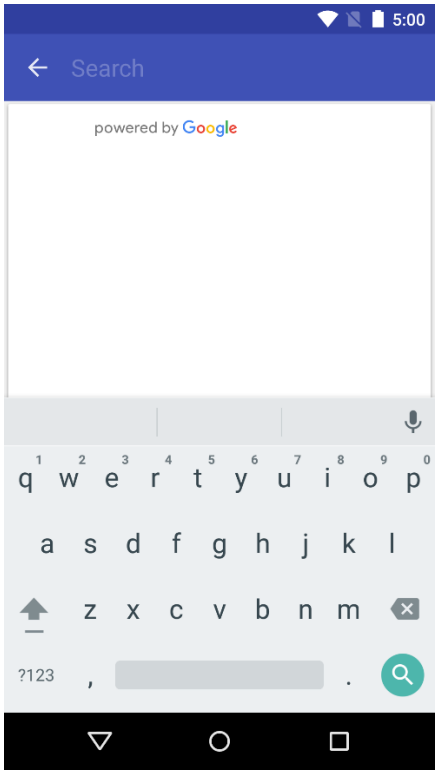
O *widget* (ferramenta) de preenchimento automático é uma caixa de diálogo de pesquisa com a funcionalidade de preenchimento automático integrada, com dois modos: Sobreposição e *FULLscreen* (tela cheia) como nas figuras 6 e 7, respectivamente, a seguir. À medida que um usuário digita os termos de pesquisa, o *widget* apresenta uma lista de locais previstos para que seja escolhido. Quando o usuário faz a seleção, uma instância *Place* é retornada, e o App vai poder usá-la para ver detalhes sobre o lugar selecionado (DEVELOPERS, 2022).

Figura 6 – *Widget* de preenchimento automático Sobreposição



Fonte: DEVELOPERS, 2022.

Figura 7 - *Widget* de preenchimento automático *FULLscreen*



Fonte: DEVELOPERS, 2022.

3.6.1 *Tokens* de sessão

O *Place* Autocomplete utiliza *tokens* (fichas) de sessão, para que seja agrupada as fases de uma consulta e seleção de uma pesquisa de preenchimento automático, como medida para economizar e otimizar o desempenho. A sessão inicia quando o usuário começa a digitar uma consulta, e finaliza quando ele seleciona um lugar, e uma chamada para o *Place Details* (Detalhes do lugar) é feita. Cada uma das sessões pode ter várias consultas de preenchimento automático, seguidas pelas seleções de lugar (DEVELOPERS, 2022).

As chaves de API usadas para cada solicitação em uma sessão necessitam permanecer vinculadas ao mesmo projeto do Console do Google *Cloud*. Após a conclusão de uma sessão, o *token* não é mais válido, então o aplicativo precisa gerar um novo *token* para cada sessão. Se o parâmetro *sessiontoken* for omitido ou reutilização de um *token*, a sessão será cobrada para cada solicitação, e será faturada separadamente (DEVELOPERS, 2022).

É necessário utilizar *tokens* de sessão em todas as sessões de preenchimento automático.

- É importante gerar um novo *token* para cada sessão.
- Observar se as chaves de API que foram usadas em todas as solicitações do *Place* Autocomplete e do *Place Details*, foi apenas em uma sessão de um mesmo projeto do Console do Google Cloud.
- Transmitir um *token* de sessão exclusivo para cada nova sessão.

É possível também rejeitar o *token* da sessão de preenchimento automático de uma solicitação. Se ele for rejeitado, cada solicitação será faturada separadamente, o que vai acionar a Autocomplete - *Per Request* (Por pedido) (DEVELOPERS, 2022).

A partir do que o usuário está digitando em uma consulta, a solicitação de preenchimento automático é chamada, a cada preenchimento com as teclas, não necessariamente caracteres, uma lista de possíveis resultados é retornada se referindo ao Autocomplete. Quando o utilizador faz uma seleção na lista de resultados apresentados, está seleção conta como uma solicitação, e todas as solicitações feitas são agrupadas e considerada como uma única solicitação (DEVELOPERS, 2022).

Por exemplo, se um usuário começar a digitar uma consulta para pesquisar "Brasília, Brasil", seguiria o seguinte fluxo:

Detectando a entrada do usuário, o App criará um novo *token* de sessão, "quot;*Token* A"".

E considerando o que o usuário digita, a API faz uma solicitação de preenchimento automático, em intervalos dos caracteres digitados, exibindo uma lista de possíveis resultados para cada um:

"B"

"Bras”

"Brasili"

"Brasília, Br"

Quando o usuário seleciona uma das opções:

E todas as solicitações restantes da consulta são somadas e adicionadas à sessão representada por "*Token* A" como uma única solicitação feita.

A seleção feita pelo usuário conta como uma solicitação de *Place Detail* e é adicionada à sessão representada pelo "*Token* A" (DEVELOPERS, 2022).

3.7 Notificações

No Android a notificação nada mais é que uma mensagem exibida fora da *UI* ou *User Interface*, que significa "Interface do Usuário" em português, que fornece ao usuário comunicações, lembretes, informações oportunas do aplicativo. Além disso, os usuários podem tocar nas notificações para que seja executado vários tipos de ações (DEVELOPERS, 2022).

A partir do Android 13, a visualização expandida de notificações, inclui um botão que permite aos usuários interromper um aplicativo que tenha serviços em primeiro plano em andamento. Os usuários podem arrastar para baixo uma notificação na gaveta para revelar a exibição expandida, podem ver mais detalhes e realizar ações (DEVELOPERS, 2022).

3.8 Recepção de *Broadcast* (Transmissão*)*

Os aplicativos para Android podem receber ou enviar diversas mensagens de *broadcast*, tanto do sistema quanto de outros Apps. As transmissões ocorrem com base em eventos de interesse programados, para gerar notificações personalizadas, mensagem entre aplicativos e fora do fluxo de normal do usuário (DEVELOPERS, 2022)

Os receptores declarados dentro do *Manifests*, como *BroadcastReceiver* (receptor de transmissão), iniciarão no App a partir do momento em que a transmissão for enviada, adicionando *BroadcastReceiver* em uma subclasse e implementando *onReceive(Context, Intent)* no Android Studio*.* O broadcast receiver registra e exibe o conteúdo da transmissão que foi programada (DEVELOPERS, 2022).

O gerenciador de pacotes do Android registra o receptor quando o App é instalado. Para que as mensagens e notificações sejam exibidas mesmo que o aplicativo não esteja em execução, o receptor se torna um ponto de entrada separado para o App (DEVELOPERS, 2022).

Cada novo objeto de componente *BroadcastReceiver* Foi criado pelo sistema, para gerenciar cada transmissão recebida, contado a partir da duração da chamada para *onReceive(Context, Intent),* que esse objeto Foi válido. Depois do retorno desse método no código, foi considerado que o componente não está mais ativo (DEVELOPERS, 2022).

4 IMPLEMENTAÇÃO

Inicialmente, para o desenvolvimento do aplicativo de *geofencing*, foram projetados 6 fragmentos de telas que compõem uma única atividade, que seria a de criar uma área de *geofencing* no local desejado.

A primeira instância feita dentro do projeto foi a implementação de um gráfico de navegação, para navegar entre as telas que foram criadas, e a ordem de prioridade entre elas.

Diante disso, o primeiro fragmento como na Figura 8, é um fragmento de permissão, e nele que é solicitado a permissão para localização final dos usuários.

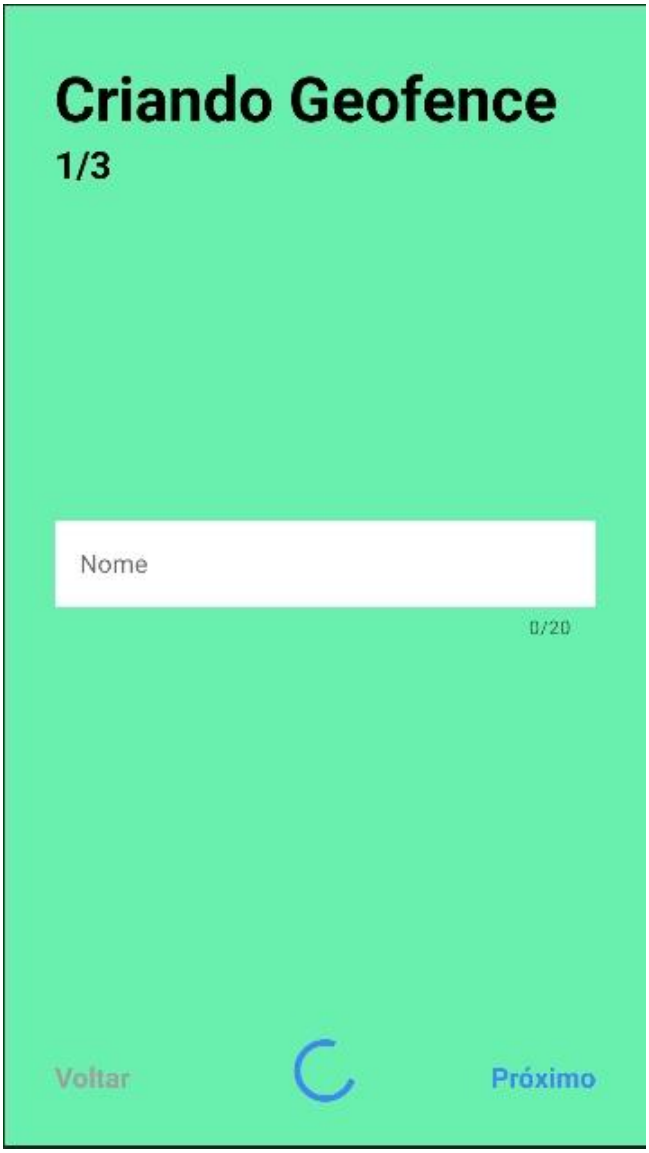
Figura 8 – Fragmento de Permissão

Fonte: Elaborado pelo autor, 2022.

Após a confirmação da permissão ao aplicativo, na primeira vez que executar o aplicativo, a próxima tela que será direcionada ao usuário será o fragmento da etapa um para criar a *geofence*. Além dessa etapa, seguirá por mais três telas ou três fragmentos para essa criação.

Esse primeiro passo ou etapa será para nomear as *geofences*, com qualquer nome desejado, como na Figura 9.

Figura 9 – Fragmento do Primeiro Passo



Fonte: Elaborado pelo autor, 2022.

Além disso, nesse fragmento da primeira etapa, será verificada a localização atual do usuário usando o SDK de lugar, para obter o código ID do país, que foi enviado para a segunda etapa.

Como é apresentado na Figura 10, o fragmento da segunda etapa, possui o *widget* de preenchimento automático, que foi personalizado para trazer o ID do local do usuário. Dessa forma, é melhorado o desempenho do aplicativo, permitindo o usuário escolher apenas as cidades dentro do país, já que seria uma forma mais lógica para adicionar *geofences*, no país em que se encontra.

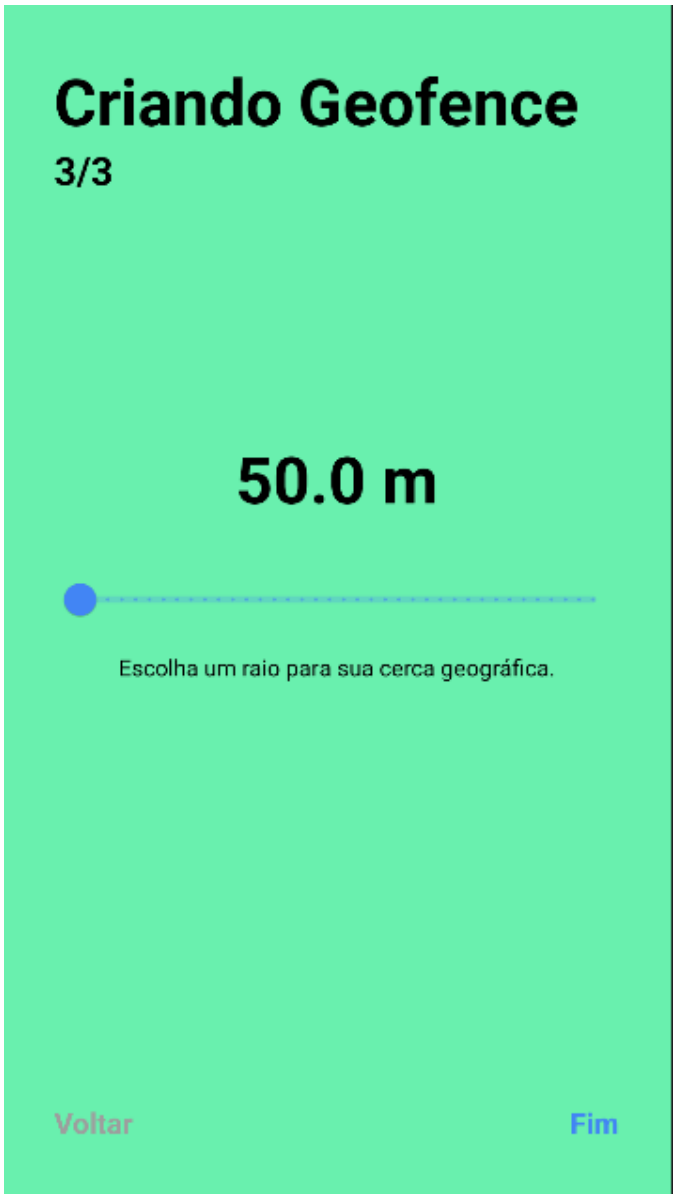
Figura 10 – Fragmento do Segundo Passo

Fonte: Elaborado pelo autor, 2022.

Depois de selecionar a cidade do país de origem e seguir, será apresentado o fragmento do terceiro passo, onde é selecionado o raio da cerca

geográfica, o valor mínimo Foi de 50 metros e o valor máximo Foi de 10 quilômetros, assim como ilustrado na Figura 11.

Figura 11 – Fragmento do Terceiro Passo



Fonte: Elaborado pelo autor, 2022.

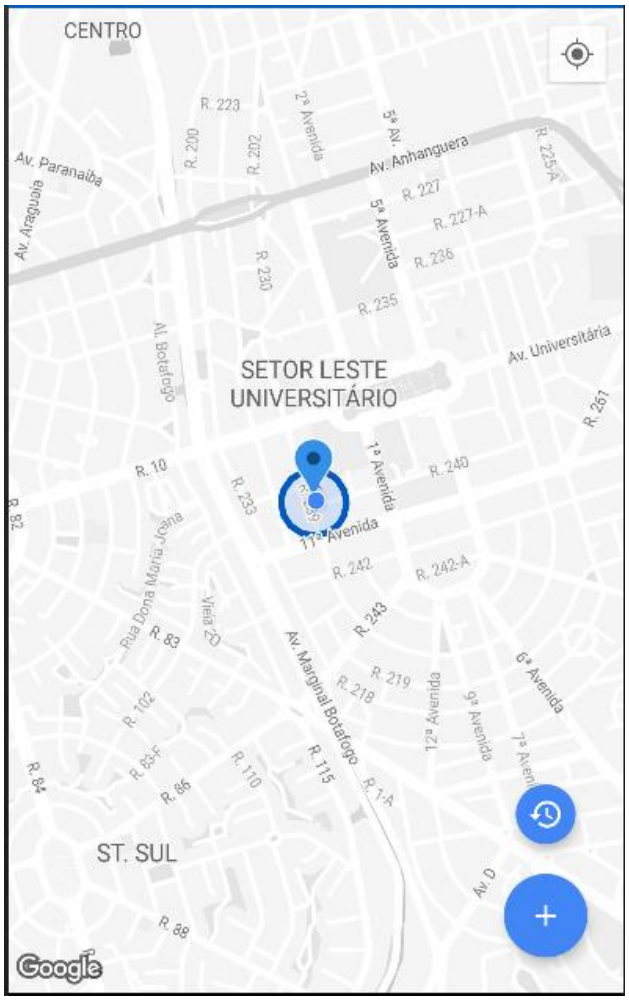
Logo depois que um usuário pressionar e garantir a escolha do raio, chegará ao fragmento de mapas, onde exibirá um texto simples ao usuário que dirá para tocar e pressionar no mapa, para adicionar a cerca geográfica.

Na primeira vez que o usuário pressionar no mapa, é necessário ativar outra permissão, a de local em segundo plano, e somente se um usuário

conceder essa permissão de localização, que sempre estará ativo o local do aparelho.

Então poderá ser adicionada à primeira cerca geográfica, além de poder adicionar várias *geofences* no aplicativo, e dentro do fragmento de mapas que possuem dois botões extras flutuantes principais, como na Figura 12.

Figura 12 – Fragmento do Mapa



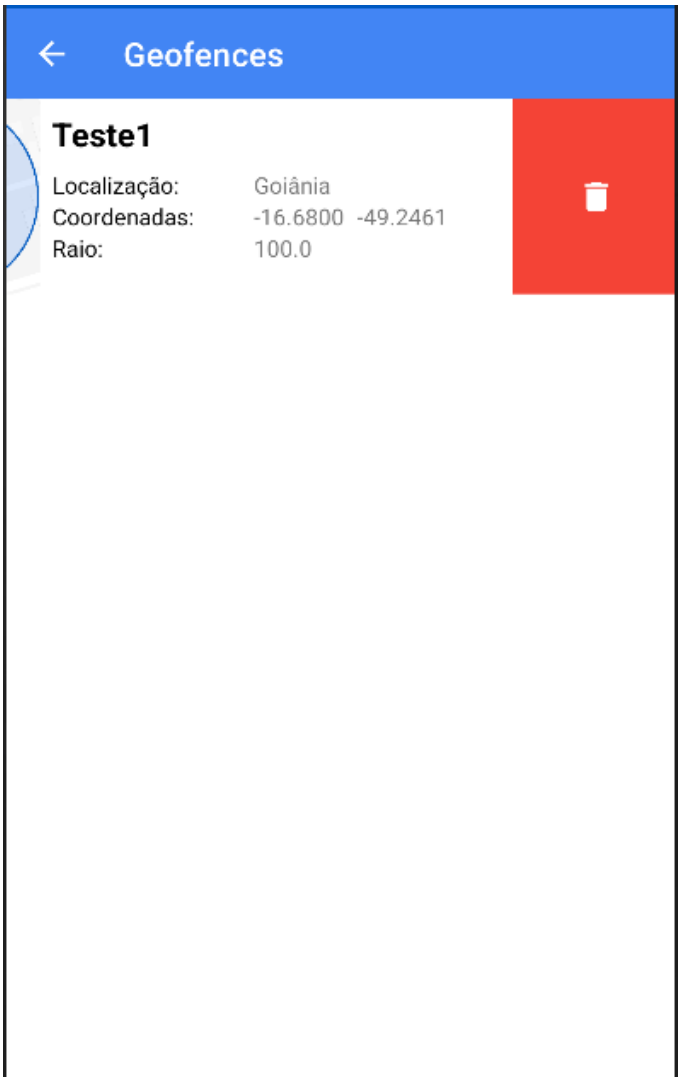
Fonte: Elaborado pelo autor, 2022.

Primeiro, o botão de ação flutuante maior, navegará o usuário para o fragmento da etapa um, onde podem ser adicionadas novas *geofences*.

No segundo botão flutuante de histórico, o menor, selecionando o usuário navegará para um fragmento de cercas geográficas, onde é capaz de ver uma lista de cercas criadas.

Nessa tela, é mostrado alguns detalhes sobre todas as cercas geográficas, que o usuário adicionou, como as coordenadas de localização no raio, e o nome da cerca geográfica como ilustrado na Figura 13 a seguir. Também é possível excluir essas *geofeces*, passando os itens para o lado esquerdo e clicando no botão de exclusão.

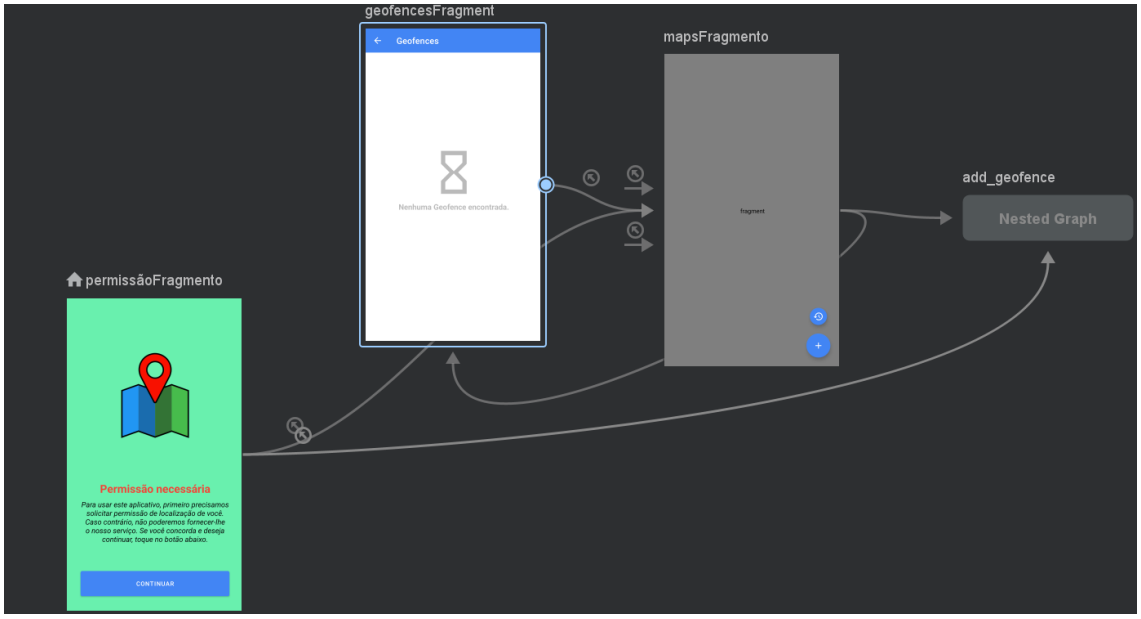
Figura 13 – Fragmento de *Geofences*



Fonte: Elaborado pelo autor, 2022.

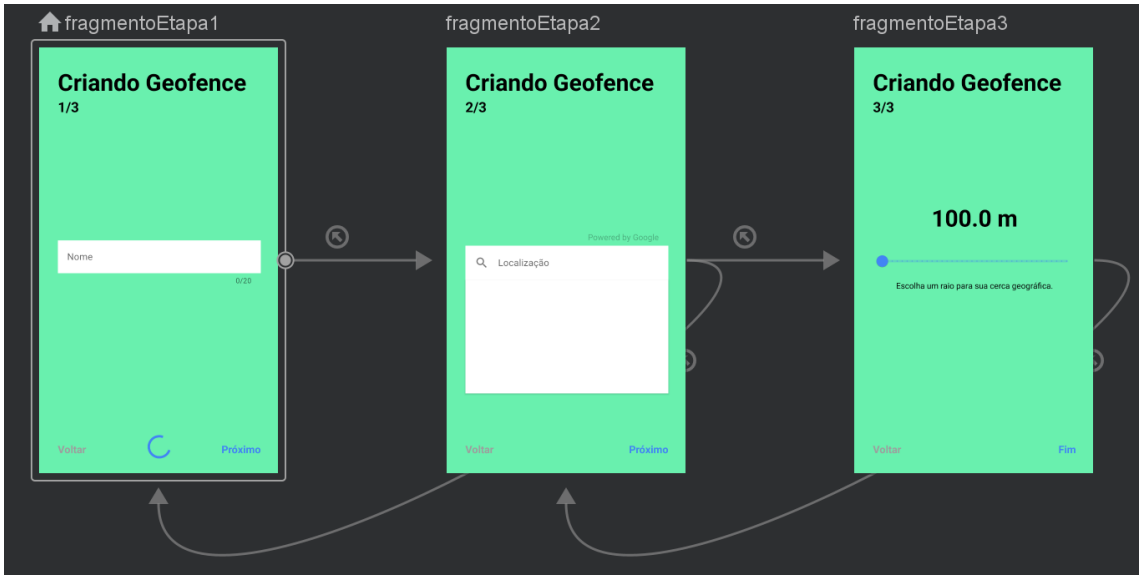
Por fim, é apresentado que os seis fragmentos diferentes, com funções que se complementam para uma única atividade final, como apresentado no gráfico de navegação da Figura 14 e Figura 15 a seguir.

Figura 14 – Gráfico de Navegação



Fonte: Elaborado pelo autor, 2022.

Figura 15 – Gráfico de Navegação *Add Geofence*



Fonte: Elaborado pelo autor, 2022.

A partir dos gráficos de navegação nas figuras, é possível perceber o fluxo dos fragmentos e telas do aplicativo, onde a primeiro fragmento de permissão tem ligação com o fragmento do mapa, que é ligado no fragmento que armazena as *geofences* criadas e a que adiciona novas *gefences*. Os fragmentos que compõe essa adição são divididos em 3 etapas apresentadas na figura 15, e que são de fácil compreensão das conexões entre uma e outra.

5 CONCLUSÃO

Partindo do referencial teórico, tudo está implementado no presente trabalho e funcionando como o esperado, e dessa forma permitir a implementação almejada nos objetivos, de acordo com as ações a seguir.

As funções e as logicas de programação apresentadas no presente trabalho, podem ser utilizadas em todas as áreas que englobam *Geofencing* e *Broadcast*.

Foram utilizadas as ferramentas voltadas as áreas *Geofencing* e *Broadcast*. E identificado as dificuldades de aplicar *Geofencing* na área de *marketing*.

Feita a análise das características, protocolos e leis que envolvem a privacidade do usuário em relação a *Spam*.

Estudado os métodos que auxilie a favor das leis, Lei Geral de Proteção de Dados (LGPD) e Anatel.

Reforçar a segurança dos protocolos, apresentando os resultados obtidos.

O aplicativo *Android* está implementado, com as funções desejadas, telas, fragmentos de mapas e fragmentos navegação.

Além disso, as perspectivas futuras em relação ao trabalho se estendem como, as melhorias que podem ser realizadas em suas funções, cores mais atrativas e almejar novas áreas de utilização do aplicativo.

Apresentar TCC na forma de resultados que foram obtidos.

Nos apêndices encontra-se o programa desenvolvido, e no anexo 1 apresenta-se o termo de autorização de publicação de produção acadêmica.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BARROS, Alexandre**. População estimada do país chega a 213,3 milhões de habitantes em 2021,** Editoria: Estatísticas Sociais, Disponível em https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/31458-populacao-estimada-do-pais-chega-a-213-3-milhoes-de-habitantes-em-2021 Acesso em: 19 fev. 2022.

CCNA-200-301. **Introdução às Redes (ITN) v7.0.** Criado por CCNA Network- Módulo 11 Endereçamento IPv4 - Disponível em: https://ccna.network/ccna-1/#Modulo_3_Protocolos_e_Modelos_CCNA_1_v7. Acesso em: 02 abril 2022.

DEVELOPERS, Google ***Activity*****.** 25 Agosto 2022 Disponível em: https://developer.android.com/reference/android/app/Activity?hl=pt-br**.** Acesso em 11 set. 2022.

DEVELOPERS, Google ANDROID STUDIO. **Conheça o Android Studio**. 2021. Disponível em: https://developer.android.com/studio/intro?hl=pt-br. Acesso em 14 maio 2022.

DEVELOPERS, Google **Fragmentos.** 2021 Disponível em: https://developer.android.com/guide/components/fragments?hl=pt-br**.** Acesso em 11 set. 2022.

DEVELOPERS, Google **Google Maps Platform.** 2022 Disponível em: https://developers.google.com/maps**.** Acesso em 11 set. 2022.

DEVELOPERS, Google **Places SDK for Android.** 2022 Disponível em: https://developers.google.com/maps/documentation/places/android-sdk/overview**.** Acesso em 11 set. 2022.

DEVELOPERS, Google **Primeiros passos com o Kotlin no Android** 2020. Disponível em: https://developer.android.com/kotlin/get-started?hl=pt-br Acesso em: 15 nov. 2022

DEVELOPERS, Google **Visão geral das notificações.** 2022 Disponível em: https://developer.android.com/develop/ui/views/notifications**.** Acesso em 14 nov. 2022.

DEVELOPERS, Google **Visão geral de transmissões.** 2022 Disponível em:https://developer.android.com/guide/components/broadcasts?hl=pt-br**.** Acesso em 15 set. 2022.

DOMPIERI, Márcia Helena Galina; SILVA, Marcos Aurélio Santos da; JÚNIOR, Lauro Rodrigues Nogueira. **Sistemas de referência terrestre e posicionamento por satélite**. 2015. 35 f. Embrapa, Embrapa Tabuleiros Costeiros, Aracaju, 2015. Disponível em: https://www.infoteca.cnptia.embrapa.br/infoteca/bitstream/doc/1042182/1/Doc197.pdf. Acesso em:10 mar. 2022.

FAGGIAN, Hugo César. **Geometria e GPS**. 2019. 60 f. Dissertação (Mestrado) - Curso de Matemática em Rede Nacional, Instituto de Ciências Matemáticas e de Computação ICMC - USP, Universidade de São Paulo (USP), São Carlos, 2019. Disponível em: https://www.teses.usp.br/teses/disponiveis/55/55136/tde-17092019150542/publico/HugoCesa rFraggian_revisada.pdf. Acesso em: 10 mar. 2022.

FERRAUDO, Gabriel. **Diferença entre IPv4 e IPv6**. Infraestrutura e tecnologia. https://www.cianet.com.br/blog/infraestrutura-e-tecnologia/diferenca-entre-ipv4-e-ipv6/. (23 abril 2020). Acesso em 09 abril 2022.

FILHO, Sidney Nicolau Venturi. **Fundamentos de redes de computadores.** Estácio; 1ª edição (1 janeiro 2016).176 p.

FOROUZAN, Behrouz. **Comunicação de Dados e Redes de Computadores**. 4. ed. Porto Alegre: AMGH, 2010. 1134 p.

KEMMIS, AMBER. ***What Is Geofencing? Everything You Need to Know About Location-Based Marketing***. *VP of Client Services at SmartBug Media*. 08 jan. 2020. Disponível em: https://www.smartbugmedia.com/blog/what-is-geofencing. Acesso em 11 mar. 2022.

KOTLIN, ***Get started with Kotlin*****.** 2022. Disponível em: https://kotlinlang.org/docs/getting-started.html . Acesso em 16 ago. 2022.

KOVACS, Leandro, **O que é SSID da rede? [Identificador de Conjunto de Serviços].** 2021**.** Disponível em: https://tecnoblog.net/responde/o-que-e-ssid-da-rede-identificador-de-conjunto-de-servicos/. Acesso em 14 maio 2022.

LEI Nº 13.709, DE 14 DE AGOSTO DE 2018**. Lei Geral de Proteção de Dados Pessoais (LGPD)-**Disponível em**:** http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm Acesso em: 19 fev. 2022.

MACHADO, Everto Fábio da Silva. **Desenvolvimento de sistemas de geolocalização e rastreamento para a plataforma Android – COMPASS**. 2015. 55 f. Monografia (Especialização) - Curso de Desenvolvimento de Sistemas para a Internet e Dispositivos Móveis, Coordenação de Licenciatura em Informática, Universidade Tecnológica Federal do Paraná, Francisco Beltrão, 2015. Disponível em: http://repositorio.utfpr.edu.br:8080/jspui/bitstream/1/20057/3/FB_DESIDM_I_2014_01.pdf. Acesso em: 10 mar. 2022.

Meirelles, F. S. Pesquisa Anual do Uso de TI nas Empresas, FGVcia: Centro de TI Aplicada, 32ª edição, 2021 (edições especiais: livrariagv@fgv.br) Questionário, Relatório e Apresentações. Disponível em: www.fgv.br/cia/pesquisa Acesso em: 25 fev. 2022.

ODOM, Wendell. **CCNA 200-301 Official Cert Guide, Volume 1**. Cisco Press; 1ª edição (10 outubro 2019). 848 p.

OLIVEIRA, Victor Laerte de. Um estudo empírico sobre o uso da linguagem de programação Kotlin para Desenvolvimento Android. Dissertação (Pós-graduação). 2019. Universidade Federal de Pernambuco

PINTO, Pedro. **IPv4: Qual a diferença entre Unicast, Broadcast e Multicast?** https://pplware.sapo.pt/tutoriais/ipv4-qual-a-diferenca-entre-unicast-broadcast-e-multicast/. 01 mar. 2018. Acesso em: 02 abril 2022.

SANTOS, Henrique Coelho. **Aplicação De Bluetooth Associada ao Geofencing**. 2021. TCC (Graduação) - Curso de Engenharia de Computação, Escola de Ciências Exatas e da Computação, Pontifícia Universidade Católica de Goiás, Goiânia, 2021

STATLER, Stephen. **Geofencing: Everything You Need to Know**. In: STATLER, Stephen. Beacon Technologies. San Diego, Califórnia, Apress, 2016. p. 307-316.

TEIXEIRA, André Manuel Rodrigues. **Análise e utilização de protocolos de redes de sensores sem fios**. 2016. 76 f. Dissertação (Mestrado) - Curso de Sistemas de Informação, Escola Superior de Tecnologia e de Gestão, Instituto Politécnico de Bragança, Bragança, 2016. Disponível em: https://bibliotecadigital.ipb.pt/bitstream/10198/14027/1/Andr%C3%A9_Teixeira_MSI_2016.pdf. Acesso em: 11 mar. 2022.

WAZLAWICK, Raul. **Metodologia de pesquisa para ciência da computação.** 2. ed. [*S. l.*: *s. n.*], 2014. 146 p.

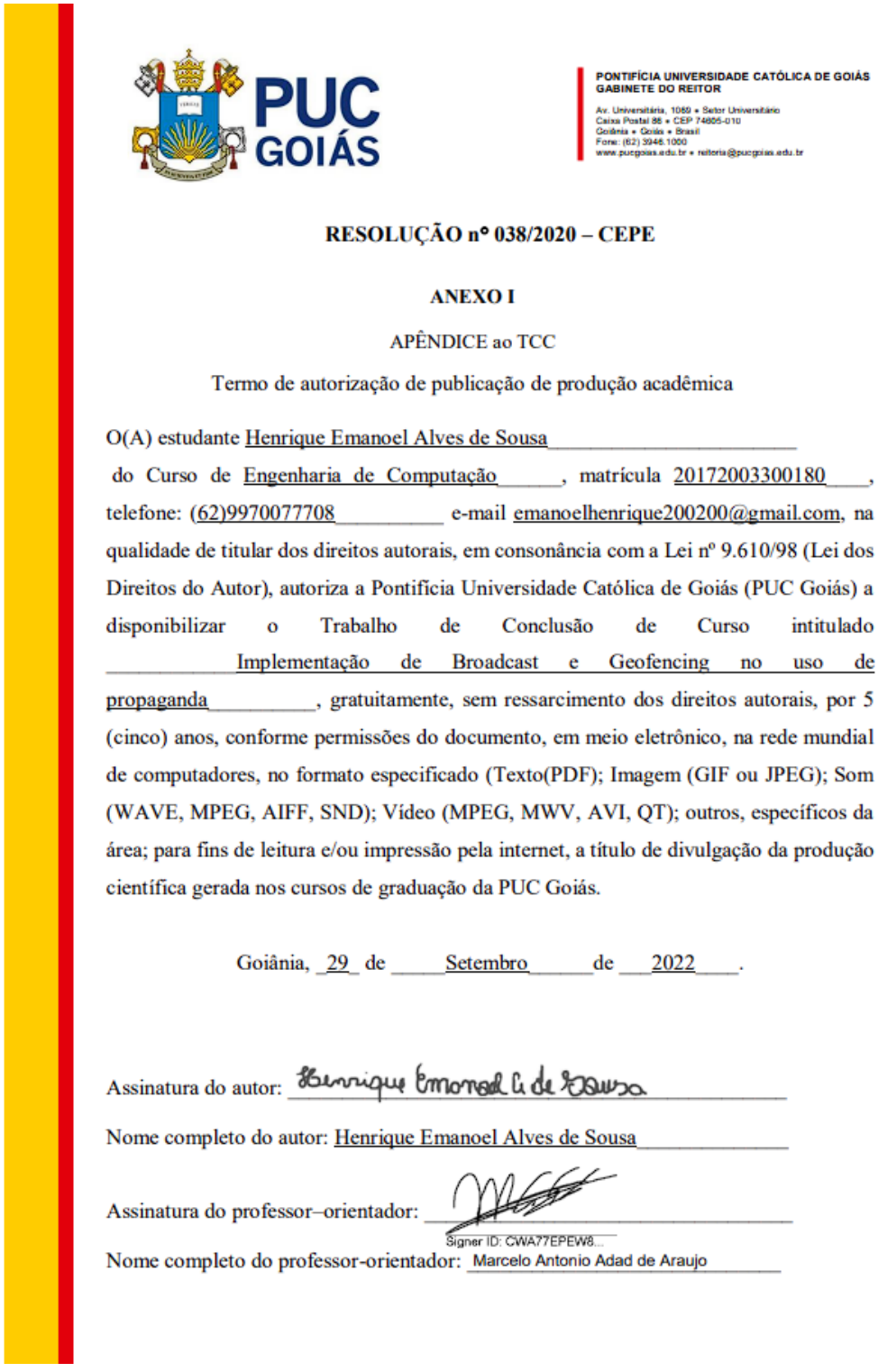
WIEXP**, GEOFENCING, o que é a geolocalização e a geocerca?** GEOFENCING \ MARKETING ANALÍTICO, 27 de julho de 2018, Disponível

em: https://wiexp.com/geofencing-o-que-e-a-geolocalizacao-no-marketing/ Acesso em: 14 maio 2022.

WHITE, Sarah K. ***What is geofencing? Putting location to work***. CIO Magazine, IDG Communications, Inc. 01 nov. 2017. Disponível em: https://www.cio.com/article/2383123/geofencing-explained.html. Acesso em 11 mar. 2022.

ZIN, Mohd Shahril Izuan Mohd; NURJI, M. F. M.; ISA, Azmi Awang Md; ISA, Mohd Sa'Ari Mohamad. ***Geofencing -based Auto-Silent Mode Application***. 2016. p. 199-204, 8 v. Tese (Doutorado) - Faculty Of Electronic And Computer Engineering (FKEKK), Centre Of Telecommunication Research And Innovation (CETRI), Universiti Teknikal Malaysia Melaka (UTEM), Malásia, 2016. Disponível em: https://journal.utem.edu.my/index.php/jtec/article/view/1398/880. Acesso em: 11 mar. 2022.

Anexo I



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR
Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Henrique Emanoel Alves de Sousa
do Curso de Engenharia de Computação, matrícula 20172003300180, telefone: (62)9970077708 e-mail emanoelhenrique200200@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Implementação de Broadcast e Geofencing no uso de propaganda, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 29 de Setembro de 2022.

Assinatura do autor: Henrique Emanoel A. de Sousa

Nome completo do autor: Henrique Emanoel Alves de Sousa

Assinatura do professor–orientador:

Signer ID: CWA77EPEW8...

Nome completo do professor-orientador: Marcelo Antonio Adad de Araujo

Anexo I - Termo de autorização de publicação de produção acadêmica

APÊNDICES

APÊNDICE 1 – Código AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.geofenceapp">

<!--Manifesto com todas as permições-->
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="WRITE_SETTINGS" />

    <application
        android:name=".MyApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.GeofenceApp">

        <receiver android:name=".broadcastreceiver.GeofenceBroadcastReceiver"
          />

<!--Entrada de chave do google API para o mapa-->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".ui.MainActivity"
            android:exported="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

APÊNDICE 2 – Código GeofencesAdapter.kt

```
package com.example.geofenceapp.adapters

import android.util.Log
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.lifecycle.viewModelScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.RecyclerView
import com.example.geofenceapp.R
import com.example.geofenceapp.data.GeofenceEntity
import com.example.geofenceapp.databinding.GeofencesRowLayoutBinding
import com.example.geofenceapp.ui.geofences.GeofencesFragmentDirections
import com.example.geofenceapp.util.MyDiffUtil
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.google.android.material.snackbar.Snackbar
import kotlinx.coroutines.launch

//Adaptador para mudanças Geofence
class GeofencesAdapter(private val sharedViewModel: SharedViewModel) :
    RecyclerView.Adapter<GeofencesAdapter.MyViewHolder>() {

    private var geofenceEntity = mutableListOf<GeofenceEntity>()

    class MyViewHolder(val binding: GeofencesRowLayoutBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun bind(geofenceEntity: GeofenceEntity) {
            binding.geofencesEntity = geofenceEntity
            binding.executePendingBindings()
        }

        companion object {
            fun from(parent: ViewGroup): MyViewHolder {
                val layoutInflater = LayoutInflater.from(parent.context)
                val binding = GeofencesRowLayoutBinding.inflate(layoutInflater,
parent, false)
                return MyViewHolder(binding)
            }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
        return MyViewHolder.from(parent)
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val currentGeofence = geofenceEntity[position]
```

```
        holder.bind(currentGeofence)

        holder.binding.deleteImageView.setOnClickListener {
            removeItem(holder, position)
        }

        holder.binding.snapshotImageView.setOnClickListener {
            val action =

GeofencesFragmentDirections.actionGeofencesFragmentToMapsFragment(cur
rentGeofence)
            holder.itemView.findNavController().navigate(action)
        }
    }

    //Remover Geofence da lista
    private fun removeItem(holder: MyViewHolder, position: Int) {
        sharedViewModel.viewModelScope.launch {
            val geofenceStopped =

sharedViewModel.stopGeofence(listOf(geofenceEntity[position].geoId))
            if (geofenceStopped) {
                sharedViewModel.removeGeofence(geofenceEntity[position])
                showSnackBar(holder, geofenceEntity[position])
            } else {
                Log.d("GeofencesAdapter", "Geofence NÃO REMOVIDO!")
            }
        }
    }

    //Barra de remoção e desfazer a remoção
    private fun showSnackBar(
        holder: MyViewHolder,
        removedItem: GeofenceEntity
    ) {
        Snackbar.make(
            holder.itemView,
            "Removido " + removedItem.name,
            Snackbar.LENGTH_LONG
        ).setAction("DESFAZER") {
            undoRemoval(holder, removedItem)
        }.show()
    }

    //desfazer a Remoção da geofence
    private fun undoRemoval(holder: MyViewHolder, removedItem:
GeofenceEntity) {
        holder.binding.motionLayout.transitionToState(R.id.start)
        sharedViewModel.addGeofence(removedItem)
        sharedViewModel.startGeofence(
```

```
                removedItem.latitude,
                removedItem.longitude
            )
        }


        //retorna contagem de cercas
        override fun getItemCount(): Int {
            return geofenceEntity.size
        }

        // setar nova entidade de geofence
        fun setData(newGeofenceEntity: MutableList<GeofenceEntity>) {
            val geofenceDiffUtil = MyDiffUtil(geofenceEntity, newGeofenceEntity)
            val diffUtilResult = DiffUtil.calculateDiff(geofenceDiffUtil)
            geofenceEntity = newGeofenceEntity
            diffUtilResult.dispatchUpdatesTo(this)
        }
    }
```

APÊNDICE 3 – Código PredictionsAdapter.kt

```
package com.example.geofenceapp.adapters

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.RecyclerView
import com.example.geofenceapp.databinding.PredictionsRowLayoutBinding
import com.example.geofenceapp.util.MyDiffUtil
import com.google.android.libraries.places.api.model.AutocompletePrediction
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

/*
   O serviço de preenchimento automático no SDK do Places para Android retorna previsões de lugar em resposta a consultas de pesquisa do usuário. Conforme o usuário digita, o serviço de preenchimento automático retorna sugestões de lugares, como empresas, endereços, códigos Plus e pontos de interesse.
*/

class PredictionsAdapter : RecyclerView.Adapter<PredictionsAdapter.MyViewHolder>() {

   private var predictions = emptyList<AutocompletePrediction>()

   private val _placeId = MutableStateFlow("")
   val placeId: StateFlow<String> get() = _placeId

   //Vinculação de layout de linha de previsões para o preenchimento automatico
   class MyViewHolder(val binding: PredictionsRowLayoutBinding) :
      RecyclerView.ViewHolder(binding.root) {
      fun bind(prediction: AutocompletePrediction) {
         binding.prediction = prediction
      }

      companion object {
         fun from(parent: ViewGroup): MyViewHolder {
            val layoutInflater = LayoutInflater.from(parent.context)
            val binding = PredictionsRowLayoutBinding.inflate(layoutInflater, parent, false)
            return MyViewHolder(binding)
         }
      }
   }

   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
```

```
        return MyViewHolder.from(parent)
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val currentPrediction = predictions[position]
        holder.bind(currentPrediction)

        holder.binding.rootLayout.setOnClickListener {
            setPlaceId(predictions[position].placeId)
        }
    }

    //contagem de itens
    override fun getItemCount(): Int {
        return predictions.size
    }

    //setar ID dos locais
    private fun setPlaceId(placeId: String){
        _placeId.value = placeId
    }

    //definir Previsão de preenchimento automático de dados
    fun setData(newPredictions: List<AutocompletePrediction>){
        val myDiffUtil = MyDiffUtil(predictions, newPredictions)
        val myDiffUtilResult = DiffUtil.calculateDiff(myDiffUtil)
        predictions = newPredictions
        myDiffUtilResult.dispatchUpdatesTo(this)
    }
}
```

APÊNDICE 4 – Código GeofencesBindings.kt

```
package com.example.geofenceapp.bindingadapters

import android.graphics.Bitmap
import android.view.View
import android.widget.ImageView
import android.widget.TextView
import androidx.constraintlayout.motion.widget.MotionLayout
import androidx.databinding.BindingAdapter
import coil.load
import com.example.geofenceapp.R
import com.example.geofenceapp.data.GeofenceEntity
import com.example.geofenceapp.util.ExtensionFunctions.disable
import com.example.geofenceapp.util.ExtensionFunctions.enable

//setar Visibilidade
@BindingAdapter("setVisibility")
fun View.setVisibility(data: List<GeofenceEntity>) {
    if (data.isNullOrEmpty()) {
        this.visibility = View.VISIBLE
    } else {
        this.visibility = View.INVISIBLE
    }
}

// lidar com a transição de movimento
@BindingAdapter("handleMotionTransition")
fun MotionLayout.handleMotionTransition(deleteImageView: ImageView) {
    deleteImageView.disable()
    this.setTransitionListener(object : MotionLayout.TransitionListener {
        override fun onTransitionStarted(p0: MotionLayout?, p1: Int, p2: Int) {}
        override fun onTransitionChange(p0: MotionLayout?, p1: Int, p2: Int, p3:
Float) {}
        override fun onTransitionTrigger(p0: MotionLayout?, p1: Int, p2: Boolean,
p3: Float) {}
        override fun onTransitionCompleted(motionLayout: MotionLayout?,
transition: Int) {
            if (motionLayout != null && transition == R.id.start) {
                deleteImageView.disable()
            } else if (motionLayout != null && transition == R.id.end) {
                deleteImageView.enable()
            }
        }
    })
}

//carregar imagem
@BindingAdapter("loadImage")
fun ImageView.loadImage(bitmap: Bitmap) {
```

```
    this.load(bitmap)
}

//analisar Coordenadas
@BindingAdapter("parseCoordinates")
fun TextView.parseCoordinates(value: Double) {
    val coordinate = String.format("%.4f", value)
    this.text = coordinate
}
```

APÊNDICE 5 – Código PredictionsBindings.kt

```
package com.example.geofenceapp.bindingadapters

import android.widget.TextView
import androidx.databinding.BindingAdapter
import com.google.android.libraries.places.api.model.AutocompletePrediction

// seta cidade do serviço de preenchimento automático
@BindingAdapter("setCidade")
fun TextView.setCity(prediction: AutocompletePrediction){
    this.text = prediction.getPrimaryText(null).toString()
}
// seta país do serviço de preenchimento automático
@BindingAdapter("setPais")
fun TextView.setCountry(prediction: AutocompletePrediction){
    this.text = prediction.getSecondaryText(null).toString()
}
```

APÊNDICE 6 – Código Step1Bindings.kt

```
package com.example.geofenceapp.bindingadapters

import android.util.Log
import android.view.View
import android.widget.ProgressBar
import android.widget.TextView
import androidx.core.widget.doOnTextChanged
import androidx.databinding.BindingAdapter
import androidx.navigation.findNavController
import com.example.geofenceapp.R
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.example.geofenceapp.viewmodels.Step1ViewModel
import com.google.android.material.textfield.TextInputEditText

//Fragmento Etapa 1 de Vinculação
//atualizar o nome da Geofence, e habilitar o botão Avançar
@BindingAdapter("updateGeofenceName", "enableNextButton", requireAll = true)
fun TextInputEditText.onTextChanged(
    sharedViewModel: SharedViewModel,
    step1ViewModel: Step1ViewModel
) {
    this.setText(sharedViewModel.geoName)
    Log.d("Bindings", sharedViewModel.geoName)
    this.doOnTextChanged { text, _, _, _ ->
        if (text.isNullOrEmpty()) {
            step1ViewModel.enableNextButton(false)
        } else {
            step1ViewModel.enableNextButton(true)
        }
        sharedViewModel.geoName = text.toString()
        Log.d("Bindings", sharedViewModel.geoName)
    }
}

//habilitar o botão Avançar e salvar ID da Geofence
@BindingAdapter("nextButtonEnabled", "saveGeofenceId", requireAll = true)
fun TextView.step1NextClicked(nextButtonEnabled: Boolean,
sharedViewModel: SharedViewModel) {
    this.setOnClickListener {
        if (nextButtonEnabled) {
            sharedViewModel.geoId = System.currentTimeMillis()

this.findNavController().navigate(R.id.action_step1Fragment_to_step2Fragment
)
        }
    }
}
```

```
//definir a visibilidade do progresso
@BindingAdapter("setProgressVisibility")
fun ProgressBar.setProgressVisibility(nextButtonEnabled: Boolean) {
    if (nextButtonEnabled) {
        this.visibility = View.GONE
    } else {
        this.visibility = View.VISIBLE
    }
}
```

APÊNDICE 7 – Código Step2Bindings.kt

```
package com.example.geofenceapp.bindingadapters

import androidx.databinding.BindingAdapter
import androidx.recyclerview.widget.RecyclerView
import com.example.geofenceapp.util.ExtensionFunctions.hide
import com.example.geofenceapp.util.ExtensionFunctions.show
import com.google.android.material.textfield.TextInputLayout

//Fragmento Etapa 2 de Vinculação
//manipular Conexão de Rede, lidar com RecyclerView
@BindingAdapter("handleNetworkConnection", "handleRecyclerView",
requireAll = true)
fun TextInputLayout.handleNetworkConnection(networkAvailable: Boolean,
recyclerView: RecyclerView) {
    if (!networkAvailable) {
        this.isErrorEnabled = true
        this.error = "Sem conexão com a Internet."
        recyclerView.hide()
    } else {
        this.isErrorEnabled = false
        this.error = null
        recyclerView.show()
    }
}
```

APÊNDICE 8 – Código Step3Bindings.kt

```
package com.example.geofenceapp.bindingadapters

import android.widget.TextView
import androidx.databinding.BindingAdapter
import com.example.geofenceapp.R
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.google.android.material.slider.Slider

//Fragmento Etapa 3 de Vinculação
//atualize o Slider Value (Valor do controle deslizante) TextView, obter GeoRadius
@BindingAdapter("updateSliderValueTextView", "getGeoRadius", requireAll = true)
fun Slider.updateSliderValue(textView: TextView, sharedViewModel: SharedViewModel) {
    updateSliderValueTextView(sharedViewModel.geoRadius, textView)
    this.addOnChangeListener { _, value, _ ->
        sharedViewModel.geoRadius = value
        updateSliderValueTextView(sharedViewModel.geoRadius, textView)
    }
}
// Atualizar SliderValue (Valor do controle deslizante)
fun Slider.updateSliderValueTextView(geoRadius: Float, textView: TextView) {
    val kilometers = geoRadius / 1000
    if (geoRadius >= 1000f) {
        textView.text = context.getString(R.string.display_kilometers, kilometers.toString())
    } else {
        textView.text = context.getString(R.string.display_meters, geoRadius.toString())
    }
    this.value = geoRadius
}
```

APÊNDICE 9 – Código GeofenceBroadcastReceiver.kt

```
package com.example.geofenceapp.broadcastreceiver

import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.os.Build
import android.util.Log
import androidx.core.app.NotificationCompat
import com.example.geofenceapp.R
import
com.example.geofenceapp.util.Constants.NOTIFICATION_CHANNEL_ID
import
com.example.geofenceapp.util.Constants.NOTIFICATION_CHANNEL_NAME
import com.example.geofenceapp.util.Constants.NOTIFICATION_ID
import com.google.android.gms.location.Geofence
import com.google.android.gms.location.GeofenceStatusCodes
import com.google.android.gms.location.GeofencingEvent

//Mensagens e notificações enviadas pela Gaeofence
class GeofenceBroadcastReceiver: BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        val geofencingEvent = GeofencingEvent.fromIntent(intent)
        if(geofencingEvent.hasError()){
            val errorMessage = GeofenceStatusCodes
                .getStatusCodeString(geofencingEvent.errorCode)
            Log.e("BroadcastReceiver", errorMessage)
            return
        }

        //EXEMPLOS para Retornos de Broadcasts para transições entrando,
dentro e fora da Geofence
        when(geofencingEvent.geofenceTransition){
            Geofence.GEOFENCE_TRANSITION_ENTER -> {
                Log.d("BroadcastReceiver", "Entrou na Geofence")
                displayNotification(context, "Só hoje! ... “Promoção imperdível: o
produto mais vendido da nossa loja com X% de desconto!")
            }
            Geofence.GEOFENCE_TRANSITION_EXIT -> {
                Log.d("BroadcastReceiver", "Fora da Geofence")
                displayNotification(context, "Não perca a promoção da nossa loja no
dia de hoje!!")
            }
            Geofence.GEOFENCE_TRANSITION_DWELL -> {
                Log.d("BroadcastReceiver", "Dentro da Geofence")
```

```
            displayNotification(context, "Parabéns! Você foi sorteado em nossa
promoção.")
        }
        else -> {
            Log.d("BroadcastReceiver", "Tipo inválido")
            displayNotification(context, "Tipo inválido Geofence")
        }
    }
}

//exibir notificação
private fun displayNotification(context: Context, geofenceTransition: String){
    val notificationManager =
        context.getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
    createNotificationChannel(notificationManager)

    val notification = NotificationCompat.Builder(context,
NOTIFICATION_CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_notification)
        .setContentTitle("Geofence")
        .setContentText(geofenceTransition)
    notificationManager.notify(NOTIFICATION_ID, notification.build())
}

//criar Canal de Notificação
private fun createNotificationChannel(notificationManager:
NotificationManager){
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
        val channel = NotificationChannel(
            NOTIFICATION_CHANNEL_ID,
            NOTIFICATION_CHANNEL_NAME,
            NotificationManager.IMPORTANCE_LOW
        )
        notificationManager.createNotificationChannel(channel)
    }
}
}
```

APÊNDICE 10 – Código Converters.kt

```
package com.example.geofenceapp.data

import android.graphics.Bitmap
import android.graphics.BitmapFactory
import androidx.room.TypeConverter
import java.io.ByteArrayOutputStream

class Converters {

    // Converte a imagem que é gerada da area do geofence
    @TypeConverter
    fun fromBitmap(bitmap: Bitmap): ByteArray {
        val outputStream = ByteArrayOutputStream()
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream)
        return outputStream.toByteArray()
    }

    @TypeConverter
    fun toBitmap(byteArray: ByteArray): Bitmap {
        return BitmapFactory.decodeByteArray(byteArray, 0, byteArray.size)
    }

}
```

APÊNDICE 11 – Código DataStoreRepository.kt

```
package com.example.geofenceapp.data

import android.content.Context
import androidx.datastore.core.DataStore
import androidx.datastore.preferences.core.Preferences
import androidx.datastore.preferences.core.booleanPreferencesKey
import androidx.datastore.preferences.core.edit
import androidx.datastore.preferences.core.emptyPreferences
import androidx.datastore.preferences.preferencesDataStore
import
com.example.geofenceapp.util.Constants.PREFERENCE_FIRST_LAUNCH
import com.example.geofenceapp.util.Constants.PREFERENCE_NAME
import dagger.hilt.android.qualifiers.ApplicationContext
import dagger.hilt.android.scopes.ViewModelScoped
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.catch
import kotlinx.coroutines.flow.map
import java.io.IOException
import javax.inject.Inject

// Armazenamento de Dados do Banco de dados
private val Context.dataStore by preferencesDataStore(PREFERENCE_NAME)

@ViewModelScoped
class DataStoreRepository @Inject constructor(@ApplicationContext private val
context: Context) {

    private object PreferenceKey {
        val                                    firstLaunch                                    =
booleanPreferencesKey(PREFERENCE_FIRST_LAUNCH)
    }

    private val dataStore: DataStore<Preferences> = context.dataStore

    //salvar primeiro lançamento
    suspend fun saveFirstLaunch(firstLaunch: Boolean) {
        dataStore.edit { preference ->
            preference[PreferenceKey.firstLaunch] = firstLaunch
        }
    }

    val readFirstLaunch: Flow<Boolean> = dataStore.data
        .catch { exception ->
            if (exception is IOException) {
                emit(emptyPreferences())
            } else {
                throw exception
            }
```

```
        }
        .map { preference ->
            val firstLaunch = preference[PreferenceKey.firstLaunch] ?: true
            firstLaunch
        }

}
```

APÊNDICE 12 – Código GeofenceDao.kt

```
package com.example.geofenceapp.data

import androidx.room.*
import kotlinx.coroutines.flow.Flow

//tabela Geofence Access Object, Objeto de acesso a dados
@Dao
interface GeofenceDao {

    //Ler geofence da tabela
    @Query("SELECT * FROM geofence_table ORDER BY id ASC")
    fun readGeofences(): Flow<MutableList<GeofenceEntity>>

    //adicionar geofence na tabela
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun addGeofence(geofenceEntity: GeofenceEntity)

    //excluir geofence na tabela
    @Delete
    suspend fun removeGeofence(geofenceEntity: GeofenceEntity)

}
```

APÊNDICE 13 – Código GeofenceDatabase.kt

```
package com.example.geofenceapp.data

import androidx.room.Database
import androidx.room.RoomDatabase
import androidx.room.TypeConverters

//Base de dados da Geofence
@Database(
    entities = [GeofenceEntity::class],
    version = 1,
    exportSchema = false
)
@TypeConverters(Converters::class)
abstract class GeofenceDatabase: RoomDatabase() {

    abstract fun geofenceDao(): GeofenceDao

}
```

APÊNDICE 14 – Código GeofenceEntity.kt

```
package com.example.geofenceapp.data

import android.graphics.Bitmap
import android.os.Parcelable
import androidx.room.Entity
import androidx.room.PrimaryKey
import com.example.geofenceapp.util.Constants.DATABASE_TABLE_NAME
import kotlinx.parcelize.IgnoredOnParcel
import kotlinx.parcelize.Parcelize

//Entidade de Geofence
@Entity(tableName = DATABASE_TABLE_NAME)
@Parcelize
class GeofenceEntity(
    val geoId: Long,
    val name: String,
    val location: String,
    val latitude: Double,
    val longitude: Double,
    val radius: Float,
    val snapshot: Bitmap
): Parcelable {
    @IgnoredOnParcel
    @PrimaryKey(autoGenerate = true)
    var id: Int = 0
}
```

APÊNDICE 15 – Código GeofenceRepository.kt

```
package com.example.geofenceapp.data

import dagger.hilt.android.scopes.ViewModelScoped
import kotlinx.coroutines.flow.Flow
import javax.inject.Inject

//Repositório da Geofence
@ViewModelScoped
class GeofenceRepository @Inject constructor(private val geofenceDao: GeofenceDao) {

    val readGeofences: Flow<MutableList<GeofenceEntity>> = geofenceDao.readGeofences()

    suspend fun addGeofence(geofenceEntity: GeofenceEntity) {
        geofenceDao.addGeofence(geofenceEntity)
    }

    suspend fun removeGeofence(geofenceEntity: GeofenceEntity) {
        geofenceDao.removeGeofence(geofenceEntity)
    }

}
```

APÊNDICE 16 – Código DatabaseModule.kt

```
package com.example.geofenceapp.di

import android.content.Context
import androidx.room.Room
import com.example.geofenceapp.data.GeofenceDatabase
import com.example.geofenceapp.util.Constants.DATABASE_NAME
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.android.qualifiers.ApplicationContext
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

//Módulo de banco de dados
@Module
@InstallIn(SingletonComponent::class)
object DatabaseModule {
    //fornecer banco de dados
    @Singleton
    @Provides
    fun provideDatabase(
        @ApplicationContext context: Context
    ) = Room.databaseBuilder(
        context,
        GeofenceDatabase::class.java,
        DATABASE_NAME
    ).build()
    @Singleton
    @Provides
    fun provideDao(database: GeofenceDatabase) = database.geofenceDao()}
```

APÊNDICE 17 – Pasta IU - Código Step1Fragment.kt

```
package com.example.geofenceapp.ui.addgeofence

import android.annotation.SuppressLint
import android.location.Geocoder
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import com.example.geofenceapp.R
import com.example.geofenceapp.databinding.FragmentStep1Binding
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.example.geofenceapp.viewmodels.Step1ViewModel
import com.google.android.gms.common.api.ApiException
import com.google.android.libraries.places.api.Places
import com.google.android.libraries.places.api.model.Place
import com.google.android.libraries.places.api.net.FindCurrentPlaceRequest
import com.google.android.libraries.places.api.net.PlacesClient
import kotlinx.coroutines.launch
import java.io.IOException

//Primeira etapa para adicionar a geofence
class Step1Fragment : Fragment() {

    private var _binding: FragmentStep1Binding? = null
    private val binding get() = _binding!!

    private val sharedViewModel: SharedViewModel by activityViewModels()
    private val step1ViewModel: Step1ViewModel by viewModels()

    private lateinit var geoCoder: Geocoder
    private lateinit var placesClient: PlacesClient

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Places.initialize(requireContext(), getString(R.string.google_maps_key))
        placesClient = Places.createClient(requireContext())
        geoCoder = Geocoder(requireContext())
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
```

```
    ): View {
        // Encher o layout para este fragmento Fragmento Etapa 1 de Vinculação
        _binding = FragmentStep1Binding.inflate(layoutInflater, container, false)
        binding.sharedViewModel = sharedViewModel
        binding.step1ViewModel = step1ViewModel
        binding.lifecycleOwner = this

        binding.step1Back.setOnClickListener {
            onStep1BackClicked()
        }

        getCountryCodeFromCurrentLocation()

        return binding.root
    }

    private fun onStep1BackClicked() {

findNavController().navigate(R.id.action_step1Fragment_to_mapsFragment)
    }

    //Verifica Local atual e locais proximos com FindCurrentPlaceRequest
    @SuppressLint("PermissãoAusente")
    private fun getCountryCodeFromCurrentLocation() {
        lifecycleScope.launch {
            //campos para definir os tipos de dados a serem retornados.
            val placeFields = listOf(Place.Field.LAT_LNG)
            val request: FindCurrentPlaceRequest =
FindCurrentPlaceRequest.newInstance(placeFields)

            //Chame findCurrentPlace e manipule a resposta, primeiro verificar se o
usuário concedeu permissão.
            val placeResponse = placesClient.findCurrentPlace(request)
            placeResponse.addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    try {
                        val response = task.result
                        val latLng = response.placeLikelihoods.first().place.latLng!!
                        val address = geoCoder.getFromLocation(
                            latLng.latitude,
                            latLng.longitude,
                            1
                        )
                        if (address != null) {
                            sharedViewModel.geoCountryCode =
address.first().countryCode
                        }
                    } catch (exception: IOException) {
                        Log.e("Step1Fragment", "getFromLocation FALHOU")
                    } finally {
```

```
                    enableNextButton()
                }
            } else {
                val exception = task.exception
                if (exception is ApiException) {
                    Log.e("Step1Fragment", exception.statusCode.toString())
                    Log.e("Step1Fragment", exception.message.toString())
                    Log.e("Step1Fragment", exception.cause?.stackTrace.toString())
                }
                enableNextButton()
            }
        }
    }
}

private fun enableNextButton() {
    if (sharedViewModel.geoName.isNotEmpty()) {
        step1ViewModel.enableNextButton(true)
    }
}

override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
}
}
```

APÊNDICE 18 – Pasta IU - Código Step2Fragment.kt

```
package com.example.geofenceapp.ui.addgeofence

import android.os.Build
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.core.widget.doOnTextChanged
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.geofenceapp.R
import com.example.geofenceapp.adapters.PredictionsAdapter
import com.example.geofenceapp.databinding.FragmentStep2Binding
import com.example.geofenceapp.util.ExtensionFunctions.hide
import com.example.geofenceapp.util.NetworkListener
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.example.geofenceapp.viewmodels.Step2ViewModel
import com.google.android.gms.common.api.ApiException
import com.google.android.libraries.places.api.Places
import
com.google.android.libraries.places.api.model.AutocompleteSessionToken
import com.google.android.libraries.places.api.model.Place
import com.google.android.libraries.places.api.model.TypeFilter
import com.google.android.libraries.places.api.net.FetchPlaceRequest
import
com.google.android.libraries.places.api.net.FindAutocompletePredictionsReque
st
import com.google.android.libraries.places.api.net.PlacesClient
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch
import java.lang.Exception

//Segunda etapa para adicionar a geofence
class Step2Fragment : Fragment() {

    private var _binding: FragmentStep2Binding? = null
    private val binding get() = _binding!!

    private val predictionsAdapter by lazy { PredictionsAdapter() }

    private val sharedViewModel: SharedViewModel by activityViewModels()
```

```
    private val step2ViewModel: Step2ViewModel by viewModels()

    //criar nova instancia de PlacesCliente
    private lateinit var placesClient: PlacesClient

    private lateinit var networkListener: NetworkListener

    //Inicializar o SDK
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Places.initialize(requireContext(), getString(R.string.google_maps_key))
        placesClient = Places.createClient(requireContext())
    }


    @RequiresApi(Build.VERSION_CODES.N)
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        // Encher o layout para este fragmento Fragmento Etapa 2 de Vinculação
        _binding = FragmentStep2Binding.inflate(inflater, container, false)
        binding.sharedViewModel = sharedViewModel
        binding.step2ViewModel = step2ViewModel
        binding.lifecycleOwner = this

        checkInternetConnection()

        binding.predictionsRecyclerView.layoutManager                    =
LinearLayoutManager(requireContext())
        binding.predictionsRecyclerView.adapter = predictionsAdapter

        binding.geofenceLocationEt.doOnTextChanged { text, _, _, _ ->
            handleNextButton(text)
            getPlaces(text)
        }

        binding.step2Back.setOnClickListener {

findNavController().navigate(R.id.action_step2Fragment_to_step1Fragment)
        }

        binding.step2Next.setOnClickListener {

findNavController().navigate(R.id.action_step2Fragment_to_step3Fragment)
        }

        subscribeToObservers()

        return binding.root
```

```
    }

    private fun handleNextButton(text: CharSequence?) {
        if (text.isNullOrEmpty()) {
            step2ViewModel.enableNextButton(false)
        }
    }

    private fun subscribeToObservers() {
        lifecycleScope.launch {
            predictionsAdapter.placeId.collectLatest { placeId ->
                if (placeId.isNotEmpty()) {
                    onCitySelected(placeId)
                }
            }
        }
    }

    //Selecionar a cidade
    private fun onCitySelected(placeId: String) {
        val placeFields = listOf(
            Place.Field.ID,
            Place.Field.LAT_LNG,
            Place.Field.NAME
        )
        val request = FetchPlaceRequest.builder(placeId, placeFields).build()
        placesClient.fetchPlace(request)
            .addOnSuccessListener { response ->
                sharedViewModel.geoLatLng = response.place.latLng!!
                sharedViewModel.geoLocationName = response.place.name!!
                sharedViewModel.geoCitySelected = true

binding.geofenceLocationEt.setText(sharedViewModel.geoLocationName)

binding.geofenceLocationEt.setSelection(sharedViewModel.geoLocationName.l
ength)
                binding.predictionsRecyclerView.hide()
                step2ViewModel.enableNextButton(true)
            }
            .addOnFailureListener { exception ->
                Log.e("Step2Fragment", exception.message.toString())
            }
    }

    //Inicializar o AutoComplete
    private fun getPlaces(text: CharSequence?) {
        if (sharedViewModel.checkDeviceLocationSettings(requireContext())) {
            lifecycleScope.launch {
                if (text.isNullOrEmpty()) {
                    predictionsAdapter.setData(emptyList())
```

```
            } else {
                val token = AutocompleteSessionToken.newInstance()

                val request =
                    FindAutocompletePredictionsRequest.builder()
                        .setCountries(sharedViewModel.geoCountryCode)
                        .setTypeFilter(TypeFilter.CITIES)
                        .setSessionToken(token)
                        .setQuery(text.toString())
                        .build()
                placesClient.findAutocompletePredictions(request)
                    .addOnSuccessListener { response ->

predictionsAdapter.setData(response.autocompletePredictions)
                        binding.predictionsRecyclerView.scheduleLayoutAnimation()
                    }
                    .addOnFailureListener { exception: Exception? ->
                        if (exception is ApiException) {
                            Log.e("Step2Fragment", exception.statusCode.toString())
                        }
                    }
            }
        }
    } else {
        Toast.makeText(
            requireContext(),
            "Ative as configurações de localização.",
            Toast.LENGTH_SHORT
        ).show()
    }
}

@RequiresApi(Build.VERSION_CODES.N)
private fun checkInternetConnection() {
    lifecycleScope.launch {
        networkListener = NetworkListener()
        networkListener.checkNetworkAvailability(requireContext())
            .collect { online ->
                Log.d("Internet", online.toString())
                step2ViewModel.setInternetAvailability(online)
                if (online && sharedViewModel.geoCitySelected) {
                    step2ViewModel.enableNextButton(true)
                } else {
                    step2ViewModel.enableNextButton(false)
                }
            }
    }
}

override fun onDestroyView() {
```

```
        super.onDestroyView()
        _binding = null
    }
}
```

APÊNDICE 19 – Pasta IU - Código Step3Fragment.kt

```
package com.example.geofenceapp.ui.addgeofence

import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.activityViewModels
import androidx.navigation.fragment.findNavController
import com.example.geofenceapp.R
import com.example.geofenceapp.databinding.FragmentStep3Binding
import com.example.geofenceapp.viewmodels.SharedViewModel

//Terceira etapa para adicionar a geofence
// Criando Geofence
class Step3Fragment : Fragment() {

    private var _binding: FragmentStep3Binding? = null
    private val binding get() = _binding!!

    private val sharedViewModel: SharedViewModel by activityViewModels()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        // Encher o layout para este fragmento Fragmento Etapa 3 de Vinculação
        _binding = FragmentStep3Binding.inflate(inflater, container, false)
        binding.sharedViewModel = sharedViewModel

        binding.step3Back.setOnClickListener {

findNavController().navigate(R.id.action_step3Fragment_to_step2Fragment)
        }

        //Escolher Tamanho da Geofence, seu raio
        binding.step3Done.setOnClickListener {
            sharedViewModel.geoRadius = binding.slider.value
            sharedViewModel.geofenceReady = true

findNavController().navigate(R.id.action_step3Fragment_to_mapsFragment)
            Log.d("Step3Fragment", sharedViewModel.geoRadius.toString())
        }

        return binding.root
    }
```

```
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

APÊNDICE 20 – Pasta IU - Código GeofencesFragment.kt

```
package com.example.geofenceapp.ui.geofences

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.geofenceapp.adapters.GeofencesAdapter
import com.example.geofenceapp.databinding.FragmentGeofencesBinding
import com.example.geofenceapp.viewmodels.SharedViewModel

// Fragmento de geofences Salvas ou não
class GeofencesFragment : Fragment() {

    private var _binding: FragmentGeofencesBinding? = null
    private val binding get() = _binding!!

    private val sharedViewModel: SharedViewModel by activityViewModels()
    private val geofencesAdapter by lazy { GeofencesAdapter(sharedViewModel)
}

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        // Encher o layout para este fragmento Fragmento Geofence de
Vinculação
        _binding = FragmentGeofencesBinding.inflate(inflater, container, false)
        binding.sharedViewModel = sharedViewModel

        setupToolbar()
        setupRecyclerView()
        observeDatabase()

        return binding.root
    }

    // barra de ferramentas de configuração
    private fun setupToolbar() {
        binding.toolbar.setNavigationOnClickListener {
            requireActivity().onBackPressed()
        }
    }

    //configurar Visualização do Reciclador
```

```
    private fun setupRecyclerView() {
        binding.geofencesRecyclerView.layoutManager                    =
LinearLayoutManager(requireContext())
        binding.geofencesRecyclerView.adapter = geofencesAdapter
    }

    //mostrar banco de dados das geofences
    private fun observeDatabase() {
        sharedViewModel.readGeofences.observe(viewLifecycleOwner) {
            geofencesAdapter.setData(it)
            binding.geofencesRecyclerView.scheduleLayoutAnimation()
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

APÊNDICE 21 – Pasta IU - Código MapsFragment.kt

```
package com.example.geofenceapp.ui.maps

import android.annotation.SuppressLint
import android.graphics.Bitmap
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.example.geofenceapp.R
import com.example.geofenceapp.databinding.FragmentMapsBinding
import com.example.geofenceapp.util.ExtensionFunctions.disable
import com.example.geofenceapp.util.ExtensionFunctions.enable
import com.example.geofenceapp.util.ExtensionFunctions.hide
import com.example.geofenceapp.util.ExtensionFunctions.show
import
com.example.geofenceapp.util.Permissions.hasBackgroundLocationPermission
import
com.example.geofenceapp.util.Permissions.requestsBackgroundLocationPermi
ssion
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.*
import com.vmadalin.easypermissions.EasyPermissions
import com.vmadalin.easypermissions.dialogs.SettingsDialog
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

// Fragmento de Mapa para vizualização dos locais
class MapsFragment : Fragment(), OnMapReadyCallback,
GoogleMap.OnMapLongClickListener,
    EasyPermissions.PermissionCallbacks, GoogleMap.SnapshotReadyCallback
{

    private var _binding: FragmentMapsBinding? = null
    private val binding get() = _binding!!

    private val args by navArgs<MapsFragmentArgs>()
```

```
    private val sharedViewModel: SharedViewModel by activityViewModels()

    private lateinit var map: GoogleMap
    private lateinit var circle: Circle

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentMapsBinding.inflate(layoutInflater, container, false)

        //navegação pelos fragmento de adicionar geofence
        binding.addGeofenceFab.setOnClickListener {

findNavController().navigate(R.id.action_mapsFragment_to_add_geofence_gra
ph)
        }

        //fragmento de geofences
        binding.geofencesFab.setOnClickListener {

findNavController().navigate(R.id.action_mapsFragment_to_geofencesFragmen
t)
        }

        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val mapFragment = childFragmentManager.findFragmentById(R.id.map)
as SupportMapFragment?
        mapFragment?.getMapAsync(this)
    }

    @SuppressLint("PermissãoAusente")
    override fun onMapReady(googleMap: GoogleMap) {
        map = googleMap

map.setMapStyle(MapStyleOptions.loadRawResourceStyle(requireContext(),
R.raw.mapstyle))
        map.isMyLocationEnabled = true
        map.setOnMapLongClickListener(this)
        map.uiSettings.apply {
            isMyLocationButtonEnabled = true
            isMapToolbarEnabled = false
        }
        onGeofenceReady()
```

```
        observeDatabase()
        backFromGeofencesFragment()
    }

    //Geocerca pronta
    private fun onGeofenceReady() {
        if (sharedViewModel.geofenceReady) {
            sharedViewModel.geofenceReady = false
            sharedViewModel.geofencePrepared = true
            displayInfoMessage()
            zoomToSelectedLocation()
        }
    }

    // mensagem de informação
    private fun displayInfoMessage() {
        lifecycleScope.launch {
            binding.infoMessageTextView.show()
            delay(2000)
            binding.infoMessageTextView.animate().alpha(0f).duration = 800
            delay(1000)
            binding.infoMessageTextView.hide()
        }
    }

    //zoom para o local selecionado
    private fun zoomToSelectedLocation() {
        map.animateCamera(
            CameraUpdateFactory.newLatLngZoom(sharedViewModel.geoLatLng,
10f), 2000, null
        )
    }

    // base de dados
    private fun observeDatabase() {
        sharedViewModel.readGeofences.observe(viewLifecycleOwner)              {
geofenceEntity ->
            map.clear()
            geofenceEntity.forEach { geofence ->
                drawCircle(LatLng(geofence.latitude,              geofence.longitude),
geofence.radius)
                drawMarker(LatLng(geofence.latitude,              geofence.longitude),
geofence.name)
            }
        }
    }

    //Voltar ao fragmento de cercas geográficas
    private fun backFromGeofencesFragment() {
        if (args.geofenceEntity != null) {
```

```
        val selectedGeofence = LatLng(
            args.geofenceEntity!!.latitude,
            args.geofenceEntity!!.longitude
        )
        zoomToGeofence(selectedGeofence, args.geofenceEntity!!.radius)
    }
}

//apertar na tela para configurar geofence
override fun onMapLongClick(location: LatLng) {
    if (hasBackgroundLocationPermission(requireContext())) {
        if (sharedViewModel.geofencePrepared) {
            setupGeofence(location)
        } else {
            Toast.makeText(
                requireContext(),
                "Você precisa criar uma nova cerca geográfica primeiro.",
                Toast.LENGTH_SHORT
            ).show()
        }
    } else {
        requestsBackgroundLocationPermission(this)
    }
}

// configurar cerca geográfica
private fun setupGeofence(location: LatLng) {
    lifecycleScope.launch {
        if (sharedViewModel.checkDeviceLocationSettings(requireContext())) {
            binding.geofencesFab.disable()
            binding.addGeofenceFab.disable()
            binding.geofenceProgressBar.show()

            drawCircle(location, sharedViewModel.geoRadius)
            drawMarker(location, sharedViewModel.geoName)
            zoomToGeofence(circle.center, circle.radius.toFloat())

            delay(1500)
            map.snapshot(this@MapsFragment)
            delay(2000)
            sharedViewModel.addGeofenceToDatabase(location)
            delay(2000)
            sharedViewModel.startGeofence(location.latitude, location.longitude)
            sharedViewModel.resetSharedValues()

            binding.geofencesFab.enable()
            binding.addGeofenceFab.enable()
            binding.geofenceProgressBar.hide()
        } else {
            Toast.makeText(
```

```
                requireContext(),
                "Ative as configurações de localização.",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}

//desenha o circulo da area
private fun drawCircle(location: LatLng, radius: Float) {
    circle = map.addCircle(
        CircleOptions().center(location).radius(radius.toDouble())
            .strokeColor(ContextCompat.getColor(requireContext(),
R.color.blue_700))
            .fillColor(ContextCompat.getColor(requireContext(),
R.color.blue_transparent))
    )
}

// desenha o marcador no mapa
private fun drawMarker(location: LatLng, name: String) {
    map.addMarker(
        MarkerOptions().position(location).title(name)

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZ
URE))
    )
}

// zoom para area da Geofence
private fun zoomToGeofence(center: LatLng, radius: Float) {
    map.animateCamera(
        CameraUpdateFactory.newLatLngBounds(
            sharedViewModel.getBounds(center, radius), 10
        ), 1000, null
    )
}

//foto da area
override fun onSnapshotReady(snapshot: Bitmap?) {
    sharedViewModel.geoSnapshot = snapshot
}

//permissões
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
```

```
//                  super.onRequestPermissionsResult(requestCode,   permissions,
grantResults)
        EasyPermissions.onRequestPermissionsResult(requestCode,
permissions, grantResults, this)
    }

    //permissão negada
    override fun onPermissionsDenied(requestCode: Int, perms: List<String>) {
        if (EasyPermissions.somePermissionPermanentlyDenied(this, perms)) {
            SettingsDialog.Builder(requireActivity()).build().show()
        } else {
            requestsBackgroundLocationPermission(this)
        }
    }

    override fun onPermissionsGranted(requestCode: Int, perms: List<String>) {
        onGeofenceReady()
        Toast.makeText(
            requireContext(),
            "Permissão concedida! Pressione e segure no mapa para adicionar uma
Geofence.",
            Toast.LENGTH_SHORT
        ).show()
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }

}
```

APÊNDICE 22 – Pasta IU - Código PermissionFragment.kt

```
package com.example.geofenceapp.ui.permission

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.activityViewModels
import androidx.navigation.fragment.findNavController
import com.example.geofenceapp.R
import com.example.geofenceapp.databinding.FragmentPermissionBinding
import com.example.geofenceapp.util.ExtensionFunctions.observeOnce
import com.example.geofenceapp.util.Permissions
import com.example.geofenceapp.viewmodels.SharedViewModel
import com.vmadalin.easypermissions.EasyPermissions
import com.vmadalin.easypermissions.dialogs.SettingsDialog
import dagger.hilt.android.AndroidEntryPoint

//Fragmento para Verificar as permissões do usuario
@AndroidEntryPoint
class PermissionFragment : Fragment(), EasyPermissions.PermissionCallbacks
{

    private var _binding: FragmentPermissionBinding? = null
    private val binding get() = _binding!!

    private val sharedViewModel: SharedViewModel by activityViewModels()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        // Encher o layout para este fragmento Fragmento permissão de
Vinculação
        _binding = FragmentPermissionBinding.inflate(inflater, container, false)

        binding.continueButton.setOnClickListener {
            if (Permissions.hasLocationPermission(requireContext())) {
                checkFirstLaunch()
            } else {
                Permissions.requestsLocationPermission(this)
            }
        }

        return binding.root
    }
```

```
    //verificar a primeira inicialização
    private fun checkFirstLaunch() {
        sharedViewModel.readFirstLaunch.observeOnce(viewLifecycleOwner)      {
firstLaunch ->
            if (firstLaunch) {

findNavController().navigate(R.id.action_permissionFragment_to_add_geofenc
e_graph)
                sharedViewModel.saveFirstLaunch(false)
            } else {

findNavController().navigate(R.id.action_permissionFragment_to_mapsFragme
nt)
            }
        }
    }

    //Resultado da solicitação de permissões
    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
//                    super.onRequestPermissionsResult(requestCode,   permissions,
grantResults)
        EasyPermissions.onRequestPermissionsResult(requestCode,
permissions, grantResults, this)
    }

    //Permissões negada
    override fun onPermissionsDenied(requestCode: Int, perms: List<String>) {
        if (EasyPermissions.somePermissionPermanentlyDenied(this, perms)) {
            SettingsDialog.Builder(requireActivity()).build().show()
        } else {
            Permissions.requestsLocationPermission(this)
        }
    }

    //Permissões concedidas
    override fun onPermissionsGranted(requestCode: Int, perms: List<String>) {
        Toast.makeText(
            requireContext(),
            "Permissão concedida! Toque no botão 'Continuar' para prosseguir.",
            Toast.LENGTH_SHORT
        ).show()
    }

    override fun onDestroyView() {
        super.onDestroyView()
```

```
        _binding = null
    }
}
```

APÊNDICE 23 – Código MainActivity.kt

```
package com.example.geofenceapp.ui

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.navigation.findNavController
import com.example.geofenceapp.R
import com.example.geofenceapp.util.Permissions
import dagger.hilt.android.AndroidEntryPoint

// Atividade de compatibilidade de aplicativos
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //verificar se tem permissão de localização
        if(Permissions.hasLocationPermission(this)){

findNavController(R.id.navHostFragment).navigate(R.id.action_permissionFrag
ment_to_mapsFragment)
        }

    }
}
```

APÊNDICE 24 – Pasta util - Código Constants.kt

```
package com.example.geofenceapp.util

// valores Constantes
object Constants {

    const val PERMISSION_LOCATION_REQUEST_CODE = 1
    const  val  PERMISSION_BACKGROUND_LOCATION_REQUEST_CODE  =
2

    const val PREFERENCE_NAME = "geofence_preference"
    const val PREFERENCE_FIRST_LAUNCH = "firstLaunch"

    const val DATABASE_TABLE_NAME = "geofence_table"
    const val DATABASE_NAME = "geofence_db"

    const val NOTIFICATION_CHANNEL_ID = "geofence_transition_id"
    const val NOTIFICATION_CHANNEL_NAME = "geofence_notification"
    const val NOTIFICATION_ID = 3

}
```

APÊNDICE 25 – Pasta util - Código ExtensionFunctions.kt

```
package com.example.geofenceapp.util

import android.view.View
import androidx.lifecycle.LifecycleOwner
import androidx.lifecycle.LiveData
import androidx.lifecycle.Observer

// Funções de Extensão, permitir, desabilitar, mostrar, ocultar
object ExtensionFunctions {

    fun View.enable(){
        this.isEnabled = true
    }

    fun View.disable(){
        this.isEnabled = false
    }

    fun View.show(){
        this.visibility = View.VISIBLE
    }

    fun View.hide(){
        this.visibility = View.GONE
    }

    fun <T> LiveData<T>.observeOnce(lifecycleOwner: LifecycleOwner,
observer: Observer<T>){
        observe(lifecycleOwner, object : Observer<T> {
            override fun onChanged(t: T) {
                observer.onChanged(t)
                removeObserver(this)
            }
        })
    }
}
```

APÊNDICE 26 – Pasta util - Código MyDiffUtil.kt

```
package com.example.geofenceapp.util

import androidx.recyclerview.widget.DiffUtil

// utilitário diferente
class MyDiffUtil<T>(
    private val oldList: List<T>,
    private val newList: List<T>
) : DiffUtil.Callback() {
    //obter o tamanho da lista antiga
    override fun getOldListSize(): Int {
        return oldList.size
    }
    //obter tamanho da lista nova
    override fun getNewListSize(): Int {
        return newList.size
    }

    // se intems forem iguais
    override fun areItemsTheSame(oldItemPosition: Int, newItemPosition: Int): Boolean {
        return oldList[oldItemPosition] === newList[newItemPosition]
    }

    //se o conteudos forem iguais
    override fun areContentsTheSame(oldItemPosition: Int, newItemPosition: Int): Boolean {
        return oldList[oldItemPosition] == newList[newItemPosition]
    }
}
```

APÊNDICE 27 – Pasta util - Código NetworkListener.kt

```
package com.example.geofenceapp.util
import android.content.Context
import android.net.ConnectivityManager
import android.net.Network
import android.net.NetworkCapabilities
import android.os.Build
import androidx.annotation.RequiresApi
import kotlinx.coroutines.flow.MutableStateFlow

//verificar a disponibilidade da rede, conectado ou não
class NetworkListener : ConnectivityManager.NetworkCallback() {

    private val isNetworkAvailable = MutableStateFlow(false)

    @RequiresApi(Build.VERSION_CODES.N)
    fun checkNetworkAvailability(context: Context): MutableStateFlow<Boolean>
{
        val connectivityManager =
            context.getSystemService(Context.CONNECTIVITY_SERVICE)        as
ConnectivityManager
        connectivityManager.registerDefaultNetworkCallback(this)

        var isConnected = false

        connectivityManager.allNetworks.forEach { network ->
            val                              networkCapability                                =
connectivityManager.getNetworkCapabilities(network)
            networkCapability?.let {
                if
(it.hasCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET)) {
                    isConnected = true
                    return@forEach
                }
            }
        }
        isNetworkAvailable.value = isConnected

        return isNetworkAvailable
    }
    override fun onAvailable(network: Network) {
        isNetworkAvailable.value = true
    }
    override fun onLost(network: Network) {
        isNetworkAvailable.value = false
    }
}
```

APÊNDICE 28 – Pasta util - Código Permissions.kt

```
package com.example.geofenceapp.util

import android.Manifest
import android.content.Context
import android.os.Build
import androidx.fragment.app.Fragment
import
com.example.geofenceapp.util.Constants.PERMISSION_BACKGROUND_LOC
ATION_REQUEST_CODE
import
com.example.geofenceapp.util.Constants.PERMISSION_LOCATION_REQUES
T_CODE
import com.vmadalin.easypermissions.EasyPermissions

object Permissions {

    //se tem permissão de localização
    fun hasLocationPermission(context: Context) =
        EasyPermissions.hasPermissions(
            context,
            Manifest.permission.ACCESS_FINE_LOCATION
        )

    // solicitar permissão de localização
    fun requestsLocationPermission(fragment: Fragment) {
        EasyPermissions.requestPermissions(
            fragment,
            "Este aplicativo não pode funcionar sem permissão de localização.",
            PERMISSION_LOCATION_REQUEST_CODE,
            Manifest.permission.ACCESS_FINE_LOCATION
        )
    }

    //se tem permissão de localização em segundo plano
    fun hasBackgroundLocationPermission(context: Context): Boolean {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            return EasyPermissions.hasPermissions(
                context,
                Manifest.permission.ACCESS_BACKGROUND_LOCATION
            )
        }
        return true
    }

    //solicitar permissão de localização em segundo plano
    fun requestsBackgroundLocationPermission(fragment: Fragment) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
```

```
            EasyPermissions.requestPermissions(
                fragment,
                "A permissão de localização em segundo plano é essencial para este
aplicativo. Sem ele não poderemos prestar-lhe o nosso serviço.",
                PERMISSION_BACKGROUND_LOCATION_REQUEST_CODE,
                Manifest.permission.ACCESS_BACKGROUND_LOCATION
            )
        }
    }
}
```

APÊNDICE 29 – Pasta viewmodels - Código SharedViewModel.kt

```
package com.example.geofenceapp.viewmodels

import android.annotation.SuppressLint
import android.app.Application
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.graphics.Bitmap
import android.location.LocationManager
import android.os.Build
import android.provider.Settings
import android.util.Log
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.asLiveData
import androidx.lifecycle.viewModelScope
import
com.example.geofenceapp.broadcastreceiver.GeofenceBroadcastReceiver
import com.example.geofenceapp.data.DataStoreRepository
import com.example.geofenceapp.data.GeofenceEntity
import com.example.geofenceapp.data.GeofenceRepository
import com.example.geofenceapp.util.Permissions
import com.google.android.gms.location.Geofence
import com.google.android.gms.location.GeofencingRequest
import com.google.android.gms.location.LocationServices
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.LatLngBounds
import com.google.maps.android.SphericalUtil
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.CompletableDeferred
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import javax.inject.Inject
import kotlin.math.sqrt

//Modelo de visualização compartilhada
//Receber dados inicializados sobre a Geofence
@HiltViewModel
class SharedViewModel @Inject constructor(
    application: Application,
    private val dataStoreRepository: DataStoreRepository,
    private val geofenceRepository: GeofenceRepository
) : AndroidViewModel(application) {

    val app = application
    private var geofencingClient =
LocationServices.getGeofencingClient(app.applicationContext)
```

```
var geoId: Long = 0L
var geoName: String = "Teste"
var geoCountryCode: String = ""
var geoLocationName: String = "Brasil"
var geoLatLng: LatLng = LatLng(0.0, 0.0)
var geoRadius: Float = 50.0f
var geoSnapshot: Bitmap? = null

var geoCitySelected = false
var geofenceReady = false
var geofencePrepared = false

//redefinir valores compartilhados
fun resetSharedValues() {
    geoId = 0L
    geoName = "Teste"
    geoCountryCode = ""
    geoLocationName = "Brasil"
    geoLatLng = LatLng(0.0, 0.0)
    geoRadius = 50.0f
    geoSnapshot = null

    geoCitySelected = false
    geofenceReady = false
    geofencePrepared = false
}

// DataStore(Banco de Dados)
val readFirstLaunch = dataStoreRepository.readFirstLaunch.asLiveData()

fun saveFirstLaunch(firstLaunch: Boolean) =
    viewModelScope.launch(Dispatchers.IO) {
        dataStoreRepository.saveFirstLaunch(firstLaunch)
    }

// Database (Base de Dados)
val readGeofences = geofenceRepository.readGeofences.asLiveData()

//chama função para adicionar Geofence
fun addGeofence(geofenceEntity: GeofenceEntity) =
    viewModelScope.launch(Dispatchers.IO) {
        geofenceRepository.addGeofence(geofenceEntity)
    }

//chama função para remover Geofence
fun removeGeofence(geofenceEntity: GeofenceEntity) =
    viewModelScope.launch(Dispatchers.IO) {
        geofenceRepository.removeGeofence(geofenceEntity)
    }
```

```
//adicionar a Geofence ao banco de dados
fun addGeofenceToDatabase(location: LatLng) {
    val geofenceEntity =
        GeofenceEntity(
            geoId,
            geoName,
            geoLocationName,
            location.latitude,
            location.longitude,
            geoRadius,
            geoSnapshot!!
        )
    addGeofence(geofenceEntity)
}

//definir intenção pendente
private fun setPendingIntent(geoId: Int): PendingIntent {
    val intent = Intent(app, GeofenceBroadcastReceiver::class.java)
    return PendingIntent.getBroadcast(
        app,
        geoId,
        intent,
        PendingIntent.FLAG_MUTABLE
        //PendingIntent.FLAG_UPDATE_CURRENT
    )
}

//Ligar Geofence
@SuppressLint("PermissãoAusente")
fun startGeofence(
    latitude: Double,
    longitude: Double
) {
    if (Permissions.hasBackgroundLocationPermission(app)) {
        val geofence = Geofence.Builder()
            .setRequestId(geoId.toString())
            .setCircularRegion(
                latitude,
                longitude,
                geoRadius
            )
            .setExpirationDuration(Geofence.NEVER_EXPIRE)
            .setTransitionTypes(
                Geofence.GEOFENCE_TRANSITION_ENTER
                        or Geofence.GEOFENCE_TRANSITION_EXIT
                        or Geofence.GEOFENCE_TRANSITION_DWELL
            )
            .setLoiteringDelay(5000)
            .build()
```

```
            val geofencingRequest = GeofencingRequest.Builder()
                .setInitialTrigger(
                    GeofencingRequest.INITIAL_TRIGGER_ENTER
                            or GeofencingRequest.INITIAL_TRIGGER_EXIT
                            or GeofencingRequest.INITIAL_TRIGGER_DWELL
                )
                .addGeofence(geofence)
                .build()

            geofencingClient.addGeofences(geofencingRequest,
setPendingIntent(geoId.toInt())).run {
                addOnSuccessListener {
                    Log.d("Geofence", "Adicionado com sucesso.")
                }
                addOnFailureListener {
                    Log.e("Geofence", it.message.toString())
                }
            }
        } else {
            Log.d("Geofence", "Permissão não concedida.")
        }
    }

    //parar Geofence
    suspend fun stopGeofence(geoIds: List<Long>): Boolean {
        return if (Permissions.hasBackgroundLocationPermission(app)) {
            val result = CompletableDeferred<Boolean>()

geofencingClient.removeGeofences(setPendingIntent(geoIds.first().toInt()))
                .addOnCompleteListener {
                    if (it.isSuccessful) {
                        result.complete(true)
                    } else {
                        result.complete(false)
                    }
                }
            result.await()
        } else {
            false
        }
    }

    // obter os limites da geofence
    fun getBounds(center: LatLng, radius: Float): LatLngBounds {
        val distanceFromCenterToCorner = radius * sqrt(2.0)
        val southWestCorner = SphericalUtil.computeOffset(center,
distanceFromCenterToCorner, 225.0)
        val northEastCorner = SphericalUtil.computeOffset(center,
distanceFromCenterToCorner, 45.0)
        return LatLngBounds(southWestCorner, northEastCorner)
```

```
    }

    //verifique as configurações de localização do dispositivo com base na
versão do aparelho
    fun checkDeviceLocationSettings(context: Context): Boolean {
        return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
            val locationManager =
                context.getSystemService(Context.LOCATION_SERVICE)          as
LocationManager
            locationManager.isLocationEnabled
        } else {
            val mode: Int = Settings.Secure.getInt(
                context.contentResolver,
                Settings.Secure.LOCATION_MODE,
                Settings.Secure.LOCATION_MODE_OFF
            )
            mode != Settings.Secure.LOCATION_MODE_OFF
        }
    }

}
```

APÊNDICE 30 – Pasta viewmodels - Código Step1ViewModel.kt

```
package com.example.geofenceapp.viewmodels

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

//Ver modelo passo 1
class Step1ViewModel: ViewModel() {

    //botão de proximo ativado e destivado
    private val _nextButtonEnabled = MutableLiveData(false)
    val nextButtonEnabled: LiveData<Boolean> get() = _nextButtonEnabled

    fun enableNextButton(enable: Boolean){
        _nextButtonEnabled.value = enable
    }

}
```

APÊNDICE 31 – Pasta viewmodels - Código Step2ViewModel.kt

```
package com.example.geofenceapp.viewmodels

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

//Ver modelo passo 2
class Step2ViewModel: ViewModel() {

    //botão de proximo ativado e destivado
    private val _nextButtonEnabled = MutableLiveData(false)
    val nextButtonEnabled: LiveData<Boolean> get() = _nextButtonEnabled

    private val _internetAvailable = MutableLiveData(true)
    val internetAvailable: LiveData<Boolean> get() = _internetAvailable

    fun enableNextButton(enable: Boolean){
        _nextButtonEnabled.value = enable
    }

    fun setInternetAvailability(online: Boolean){
        _internetAvailable.value = online
    }

}
```

APÊNDICE 32 – Código MyApplication.kt

```
package com.example.geofenceapp

import android.app.Application
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class MyApplication: Application()
```

APÊNDICE 33 – Pasta layout - Código activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.MainActivity">

    <fragment
        android:id="@+id/navHostFragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_graph" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

APÊNDICE 33 – Pasta layout - Código fragment_geofences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>

        <variable
            name="sharedViewModel"
            type="com.example.geofenceapp.viewmodels.SharedViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".ui.geofences.GeofencesFragment">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="?attr/actionBarTheme"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:navigationIcon="@drawable/ic_back"
            app:title="Geofences"
            app:titleTextColor="@color/white" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/geofences_recyclerView"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layoutAnimation="@anim/recyclerview_layout_animation"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/toolbar" />

        <ImageView
            android:id="@+id/empty_imageView"
            setVisibility="@{sharedViewModel.readGeofences}"
            android:layout_width="120dp"
            android:layout_height="120dp"
```

```
            android:alpha="0.5"
            android:src="@drawable/ic_hourglass"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.45" />

        <TextView
            android:id="@+id/empty_textView"
            setVisibility="@{sharedViewModel.readGeofences}"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:alpha="0.5"
            android:text="@string/no_geofence_found"
            android:textSize="18sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="@+id/empty_imageView"
            app:layout_constraintStart_toStartOf="@+id/empty_imageView"
            app:layout_constraintTop_toBottomOf="@+id/empty_imageView" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

APÊNDICE 34 – Pasta layout - Código fragment_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:context=".ui.maps.MapsFragment" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/add_geofence_fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="24dp"
        android:layout_marginBottom="24dp"
        android:background="@color/view_state_background_color"
        android:clickable="true"
        android:focusable="true"
        android:src="@drawable/ic_add"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:tint="@color/white" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/geofences_fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:background="@color/view_state_background_color"
        android:clickable="true"
        android:focusable="true"
        app:fabSize="mini"
        app:layout_constraintBottom_toTopOf="@+id/add_geofence_fab"
        app:layout_constraintEnd_toEndOf="@+id/add_geofence_fab"
        app:layout_constraintStart_toStartOf="@+id/add_geofence_fab"
```

```
        app:srcCompat="@drawable/ic_history"
        app:tint="@color/white" />

    <TextView
        android:id="@+id/infoMessage_textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:text="@string/long_press_on_the_map_n_to_add_a_geofence"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="22sp"
        android:textStyle="bold"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ProgressBar
        android:id="@+id/geofence_progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

APÊNDICE 35 – Pasta layout - Código fragment_permission.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/quantum_vanillagreenA200"
    tools:context=".ui.permission.PermissionFragment">


    <Button
        android:id="@+id/continue_button"
        android:layout_width="0dp"
        android:layout_height="70dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="32dp"
        android:text="@string/continue_btn"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/permission_description_textView"
        android:layout_width="347dp"
        android:layout_height="113dp"
        android:layout_marginBottom="48dp"
        android:text="@string/permission_description"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="17sp"
        android:textStyle="italic"
        app:layout_constraintBottom_toTopOf="@+id/continue_button"
        app:layout_constraintEnd_toEndOf="@+id/continue_button"
        app:layout_constraintStart_toStartOf="@+id/continue_button" />

    <TextView
        android:id="@+id/permission_required_textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:text="@string/permission_required"
        android:textAlignment="center"
        android:textColor="@color/red"
        android:textSize="26sp"
        android:textStyle="bold"
```

```
app:layout_constraintBottom_toTopOf="@+id/permission_description_textView"

app:layout_constraintEnd_toEndOf="@+id/permission_description_textView"

app:layout_constraintStart_toStartOf="@+id/permission_description_textView"
/>

    <ImageView
        android:id="@+id/welcome_imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="100dp"
        android:src="@drawable/ic_welcome_image"

app:layout_constraintBottom_toTopOf="@+id/permission_required_textView"
        app:layout_constraintEnd_toEndOf="@+id/permission_required_textView"

app:layout_constraintStart_toStartOf="@+id/permission_required_textView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

APÊNDICE 36 – Pasta layout - Código fragment_step1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="sharedViewModel"
            type="com.example.geofenceapp.viewmodels.SharedViewModel" />
        <variable
            name="step1ViewModel"
            type="com.example.geofenceapp.viewmodels.Step1ViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/quantum_vanillagreenA200"
        tools:context=".ui.addgeofence.Step1Fragment">

        <TextView
            android:id="@+id/geofenceOne_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="32dp"
            android:layout_marginTop="32dp"
            android:text="@string/geofence"
            android:textColor="@color/black"
            android:textSize="40sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/stepOne_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/one_three"
            android:textColor="@color/black"
            android:textSize="24sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/geofenceOne_textView"
            app:layout_constraintTop_toBottomOf="@+id/geofenceOne_textView"
/>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/geofence_name_textInputLayout"
```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:hint="@string/name"
        app:counterEnabled="true"
        app:counterMaxLength="20"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/geofence_name_et"
            enableNextButton="@{step1ViewModel}"
            updateGeofenceName="@{sharedViewModel}"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/white"
            android:maxLength="20" />

    </com.google.android.material.textfield.TextInputLayout>

    <TextView
        android:id="@+id/step1_next"
        nextButtonEnabled="@{step1ViewModel.nextButtonEnabled}"
        saveGeofenceId="@{sharedViewModel}"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="32dp"
        android:enabled="@{step1ViewModel.nextButtonEnabled}"
        android:text="@string/next"
        android:textColor="@color/view_state_background_color"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <TextView
        android:id="@+id/step1_back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginBottom="32dp"
        android:text="@string/back"
        android:textColor="@color/quantum_grey"
```

```
            android:textSize="18sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent" />

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"
            setProgressVisibility="@{step1ViewModel.nextButtonEnabled}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

APÊNDICE 37 – Pasta layout - Código fragment_step2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="sharedViewModel"
            type="com.example.geofenceapp.viewmodels.SharedViewModel" />
        <variable
            name="step2ViewModel"
            type="com.example.geofenceapp.viewmodels.Step2ViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/quantum_vanillagreenA200"
        tools:context=".ui.addgeofence.Step2Fragment">

        <TextView
            android:id="@+id/geofenceTwo_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="32dp"
            android:layout_marginTop="32dp"
            android:text="@string/geofence"
            android:textColor="@color/black"
            android:textSize="40sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/stepTwo_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/two_three"
            android:textColor="@color/black"
            android:textSize="24sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/geofenceTwo_textView"
            app:layout_constraintTop_toBottomOf="@+id/geofenceTwo_textView"
/>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/geofence_location_textInputLayout"
```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        handleNetworkConnection="@{step2ViewModel.internetAvailable}"
        handleRecyclerView="@{predictionsRecyclerView}"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:hint="@string/location"
        app:errorIconDrawable="@drawable/ic_wifi_off"
        app:errorTextColor="@color/red"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/geofence_location_et"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:drawableStart="@drawable/ic_search"
            android:drawablePadding="16dp"
            android:background="@color/white"
            android:text="@{sharedViewModel.geoLocationName}"/>

    </com.google.android.material.textfield.TextInputLayout>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/predictions_recyclerView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/quantum_white_100"
        android:elevation="2dp"
        android:layoutAnimation="@anim/recyclerview_layout_animation"
        app:layout_constraintHeight_max="200dp"

app:layout_constraintEnd_toEndOf="@+id/geofence_location_textInputLayout"

app:layout_constraintStart_toStartOf="@+id/geofence_location_textInputLayout
"

app:layout_constraintTop_toBottomOf="@+id/geofence_location_textInputLayo
ut" />

    <TextView
        android:id="@+id/step2_next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="32dp"
```

```
            android:layout_marginBottom="32dp"
            android:text="@string/next"
            android:enabled="@{step2ViewModel.nextButtonEnabled}"
            android:textColor="@color/view_state_background_color"
            android:textSize="18sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent" />

        <TextView
            android:id="@+id/step2_back"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="32dp"
            android:layout_marginBottom="32dp"
            android:text="@string/back"
            android:textColor="@color/quantum_grey"
            android:textSize="18sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent" />

        <TextView
            android:id="@+id/powered_by_google_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="16dp"
            android:alpha="0.5"
            android:text="@string/powered_by_google"

app:layout_constraintBottom_toTopOf="@+id/geofence_location_textInputLayo
ut"

app:layout_constraintEnd_toEndOf="@+id/geofence_location_textInputLayout"
            app:layout_constraintHorizontal_bias="1.0"

app:layout_constraintStart_toStartOf="@+id/geofence_location_textInputLayout
" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

APÊNDICE 38 – Pasta layout - Código fragment_step3.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="sharedViewModel"
            type="com.example.geofenceapp.viewmodels.SharedViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/quantum_vanillagreenA200"
        tools:context=".ui.addgeofence.Step3Fragment">

        <TextView
            android:id="@+id/geofenceThree_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="32dp"
            android:layout_marginTop="32dp"
            android:text="@string/geofence"
            android:textColor="@color/black"
            android:textSize="40sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/stepThree_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/three_three"
            android:textColor="@color/black"
            android:textSize="24sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/geofenceThree_textView"
            app:layout_constraintTop_toBottomOf="@+id/geofenceThree_textView"
/>

        <TextView
            android:id="@+id/slider_value_textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"
```

```
    android:text="@string/_50_0_m"
    android:textAlignment="center"
    android:textColor="@color/black"
    android:textSize="40sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/slider"
    app:layout_constraintEnd_toEndOf="@+id/slider"
    app:layout_constraintStart_toStartOf="@+id/slider" />

<com.google.android.material.slider.Slider
    android:id="@+id/slider"
    updateSliderValueTextView="@{sliderValueTextView}"
    getGeoRadius="@{sharedViewModel}"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"
    android:stepSize="50.0"
    android:value="50.0"
    android:valueFrom="50.0"
    android:valueTo="10000.0"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/geofence_radius_desc"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="@string/choose_a_radius_for_your_geofence"
    android:textAlignment="center"
    android:textColor="@color/black"
    app:layout_constraintEnd_toEndOf="@+id/slider"
    app:layout_constraintStart_toStartOf="@+id/slider"
    app:layout_constraintTop_toBottomOf="@+id/slider" />

<TextView
    android:id="@+id/step3_done"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="32dp"
    android:layout_marginBottom="32dp"
    android:text="@string/done"
    android:textColor="@color/view_state_background_color"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
```

```
            app:layout_constraintEnd_toEndOf="parent" />

        <TextView
            android:id="@+id/step3_back"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="32dp"
            android:layout_marginBottom="32dp"
            android:text="@string/back"
            android:textColor="@color/quantum_grey"
            android:textSize="18sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

APÊNDICE 39 – Pasta layout - geofences_row_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="geofencesEntity"
            type="com.example.geofenceapp.data.GeofenceEntity" />
    </data>

    <androidx.constraintlayout.motion.widget.MotionLayout
        android:id="@+id/motionLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        handleMotionTransition="@{deleteImageView}"
        app:layoutDescription="@xml/geofences_row_layout_scene">

        <View
            android:id="@+id/red_background"
            android:layout_width="wrap_content"
            android:layout_height="120dp"
            android:background="@color/red"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageView
            android:id="@+id/delete_imageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="36dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="@+id/red_background"
            app:layout_constraintHorizontal_bias="1.0"
            app:layout_constraintStart_toStartOf="@+id/red_background"
            app:layout_constraintTop_toTopOf="@+id/red_background"
            app:srcCompat="@drawable/ic_delete"
            app:tint="@color/white" />

        <View
            android:id="@+id/white_background"
            android:layout_width="wrap_content"
            android:layout_height="120dp"
            android:background="@color/white"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<ImageView
    android:id="@+id/snapshot_imageView"
    android:layout_width="120dp"
    android:layout_height="120dp"
    android:scaleType="centerCrop"
    loadImage="@{geofencesEntity.snapshot}"
    app:layout_constraintBottom_toBottomOf="@id/white_background"
    app:layout_constraintStart_toStartOf="@id/white_background"
    app:layout_constraintTop_toTopOf="@id/white_background"
    tools:srcCompat="@tools:sample/avatars" />

<TextView
    android:id="@+id/name_textView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="@{geofencesEntity.name}"
    android:textColor="@color/black"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="@id/white_background"
    app:layout_constraintStart_toEndOf="@+id/snapshot_imageView"
    app:layout_constraintTop_toTopOf="@id/white_background" />

<TextView
    android:id="@+id/locationName_textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:text="@string/location_txt"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="@+id/name_textView"
    app:layout_constraintTop_toBottomOf="@+id/name_textView" />

<TextView
    android:id="@+id/coordinates_textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/coordinates"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="@+id/locationName_textView"
```

```
            app:layout_constraintTop_toBottomOf="@+id/locationName_textView"
/>

        <TextView
            android:id="@+id/radius_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radius"
            android:textColor="@color/black"
            app:layout_constraintStart_toStartOf="@+id/coordinates_textView"
            app:layout_constraintTop_toBottomOf="@+id/coordinates_textView" />

        <TextView
            android:id="@+id/locationValue_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="46dp"
            android:text="@{geofencesEntity.location}"
            app:layout_constraintStart_toEndOf="@+id/locationName_textView"
            app:layout_constraintTop_toTopOf="@+id/locationName_textView" />

        <TextView
            android:id="@+id/latitude_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            parseCoordinates="@{geofencesEntity.latitude}"
            app:layout_constraintStart_toStartOf="@+id/locationValue_textView"
            app:layout_constraintTop_toBottomOf="@+id/locationValue_textView"
/>

        <TextView
            android:id="@+id/longitude_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            parseCoordinates="@{geofencesEntity.longitude}"
            app:layout_constraintBottom_toBottomOf="@+id/latitude_textView"
            app:layout_constraintStart_toEndOf="@+id/latitude_textView"
            app:layout_constraintTop_toTopOf="@+id/latitude_textView" />

        <TextView
            android:id="@+id/radiusValue_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{String.valueOf(geofencesEntity.radius)}"
            app:layout_constraintStart_toStartOf="@+id/latitude_textView"
            app:layout_constraintTop_toBottomOf="@+id/latitude_textView" />

    </androidx.constraintlayout.motion.widget.MotionLayout>
</layout>
```

APÊNDICE 40 – Pasta layout - predictions_row_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto">
<!--
   O widget de preenchimento automático é uma caixa de diálogo de pesquisa com a funcionalidade de preenchimento
   automático integrada. À medida que um usuário digita termos de pesquisa, o widget apresenta uma
   lista de locais previstos para escolha. Quando o usuário faz uma seleção, uma instância Place é
   retornada, e o app vai poder usá-la para ver detalhes sobre o lugar selecionado.
-->
   <data>
      <import
type="com.google.android.libraries.places.api.model.AutocompletePrediction"/>
      <variable
         name="prediction"
         type="AutocompletePrediction" />
   </data>

   <androidx.constraintlayout.widget.ConstraintLayout
      android:id="@+id/rootLayout"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:paddingBottom="16dp"
      android:paddingTop="16dp">

      <TextView
         android:id="@+id/city_textView"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_marginStart="16dp"
         android:layout_marginEnd="16dp"
         android:name                                                    =
"com.google.android.libraries.places.widget.AutocompleteSupportFragment"
         setCidade="@{prediction}"
         android:textSize="18sp"
         android:textStyle="bold"
         app:layout_constraintEnd_toEndOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent" />

      <TextView
         android:id="@+id/country_textView"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
```

```
        android:name                                                    =
"com.google.android.libraries.places.widget.AutocompleteSupportFragment"
        setPais="@{prediction}"
        app:layout_constraintEnd_toEndOf="@+id/city_textView"
        app:layout_constraintStart_toStartOf="@+id/city_textView"
        app:layout_constraintTop_toBottomOf="@+id/city_textView" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

APÊNDICE 41 – Pasta navigation - nav_graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/permissionFragment">

    <fragment
        android:id="@+id/permissionFragment"

android:name="com.example.geofenceapp.ui.permission.PermissionFragment"
        android:label="fragment_permission"
        tools:layout="@layout/fragment_permission" >
        <action
            android:id="@+id/action_permissionFragment_to_mapsFragment"
            app:destination="@id/mapsFragment"
            app:popUpTo="@id/permissionFragment"
            app:popUpToInclusive="true" />
        <action
            android:id="@+id/action_permissionFragment_to_add_geofence_graph"
            app:destination="@id/add_geofence_graph"
            app:popUpTo="@id/permissionFragment"
            app:popUpToInclusive="true" />
    </fragment>
    <fragment
        android:id="@+id/mapsFragment"
        android:name="com.example.geofenceapp.ui.maps.MapsFragment"
        android:label="fragment_maps"
        tools:layout="@layout/fragment_maps" >
        <action
            android:id="@+id/action_mapsFragment_to_geofencesFragment"
            app:destination="@id/geofencesFragment"
            app:enterAnim="@anim/from_right"
            app:exitAnim="@anim/to_left"
            app:popEnterAnim="@anim/from_left"
            app:popExitAnim="@anim/to_right" />
        <action
            android:id="@+id/action_mapsFragment_to_add_geofence_graph"
            app:destination="@id/add_geofence_graph"
            app:enterAnim="@anim/from_bottom"
            app:exitAnim="@anim/to_top"
            app:popEnterAnim="@anim/from_top"
            app:popExitAnim="@anim/to_bottom" />
        <argument
            android:name="geofenceEntity"
            app:argType="com.example.geofenceapp.data.GeofenceEntity"
            app:nullable="true"
```

```
            android:defaultValue="@null" />
    </fragment>
    <navigation
        android:id="@+id/add_geofence_graph"
        app:startDestination="@id/step1Fragment">
        <fragment
            android:id="@+id/step1Fragment"

android:name="com.example.geofenceapp.ui.addgeofence.Step1Fragment"
            android:label="fragment_step1"
            tools:layout="@layout/fragment_step1" >
            <action
                android:id="@+id/action_step1Fragment_to_step2Fragment"
                app:destination="@id/step2Fragment"
                app:enterAnim="@anim/from_right"
                app:exitAnim="@anim/to_left"
                app:popEnterAnim="@anim/from_left"
                app:popExitAnim="@anim/to_right"
                app:popUpTo="@id/step1Fragment"
                app:popUpToInclusive="true" />
        </fragment>
        <fragment
            android:id="@+id/step2Fragment"

android:name="com.example.geofenceapp.ui.addgeofence.Step2Fragment"
            android:label="fragment_step2"
            tools:layout="@layout/fragment_step2" >
            <action
                android:id="@+id/action_step2Fragment_to_step3Fragment"
                app:destination="@id/step3Fragment"
                app:enterAnim="@anim/from_right"
                app:exitAnim="@anim/to_left"
                app:popEnterAnim="@anim/from_left"
                app:popExitAnim="@anim/to_right"
                app:popUpTo="@id/step2Fragment"
                app:popUpToInclusive="true" />
            <action
                android:id="@+id/action_step2Fragment_to_step1Fragment"
                app:destination="@id/step1Fragment"
                app:enterAnim="@anim/from_left"
                app:exitAnim="@anim/to_right"
                app:popUpTo="@id/step2Fragment"
                app:popUpToInclusive="true" />
        </fragment>
        <fragment
            android:id="@+id/step3Fragment"

android:name="com.example.geofenceapp.ui.addgeofence.Step3Fragment"
            android:label="fragment_step3"
            tools:layout="@layout/fragment_step3" >
```

```
            <action
                android:id="@+id/action_step3Fragment_to_step2Fragment"
                app:destination="@id/step2Fragment"
                app:enterAnim="@anim/from_left"
                app:exitAnim="@anim/to_right"
                app:popUpTo="@id/step3Fragment"
                app:popUpToInclusive="true" />
        </fragment>
    </navigation>
    <fragment
        android:id="@+id/geofencesFragment"

android:name="com.example.geofenceapp.ui.geofences.GeofencesFragment"
        android:label="fragment_geofences"
        tools:layout="@layout/fragment_geofences" >
        <action
            android:id="@+id/action_geofencesFragment_to_mapsFragment"
            app:destination="@id/mapsFragment"
            app:launchSingleTop="true"
            app:popUpTo="@id/geofencesFragment"
            app:popUpToInclusive="true" />
    </fragment>
    <action
        android:id="@+id/action_step1Fragment_to_mapsFragment"
        app:destination="@id/mapsFragment"
        app:popUpTo="@id/step1Fragment"
        app:popUpToInclusive="true"
        app:launchSingleTop="true" />
    <action
        android:id="@+id/action_step3Fragment_to_mapsFragment"
        app:destination="@id/mapsFragment"
        app:popUpTo="@id/step3Fragment"
        app:popUpToInclusive="true"
        app:launchSingleTop="true" />
</navigation>
```

APÊNDICE 42 – Estilo do mapa - mapstyle.json

```
// Estilo do mapa
[
  {
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#f5f5f5"
      }
    ]
  },
  {
    "elementType": "labels.icon",
    "stylers": [
      {
        "visibility": "off"
      }
    ]
  },
  {
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#616161"
      }
    ]
  },
  {
    "elementType": "labels.text.stroke",
    "stylers": [
      {
        "color": "#f5f5f5"
      }
    ]
  },
  {
    "featureType": "administrative.land_parcel",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#bdbdbd"
      }
    ]
  },
  {
    "featureType": "poi",
    "elementType": "geometry",
    "stylers": [
```

```
    {
      "color": "#eeeeee"
    }
  ]
},
{
  "featureType": "poi",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#757575"
    }
  ]
},
{
  "featureType": "poi.park",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#e5e5e5"
    }
  ]
},
{
  "featureType": "poi.park",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#9e9e9e"
    }
  ]
},
{
  "featureType": "road",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#ffffff"
    }
  ]
},
{
  "featureType": "road.arterial",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#757575"
    }
  ]
},
```

```
{
  "featureType": "road.highway",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#dadada"
    }
  ]
},
{
  "featureType": "road.highway",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#616161"
    }
  ]
},
{
  "featureType": "road.local",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#9e9e9e"
    }
  ]
},
{
  "featureType": "transit.line",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#e5e5e5"
    }
  ]
},
{
  "featureType": "transit.station",
  "elementType": "geometry",
  "stylers": [
    {
      "color": "#eeeeee"
    }
  ]
},
{
  "featureType": "water",
  "elementType": "geometry",
  "stylers": [
    {
```

```
        "color": "#c9c9c9"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#9e9e9e"
      }
    ]
  }
]
```

APÊNDICE 43 – Pasta values - colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="blue_200">#80b4ff</color>
    <color name="blue_500">#4285F4</color>
    <color name="blue_transparent">#304285F4</color>
    <color name="blue_700">#0059c1</color>
    <color name="teal_200">#4285F4</color>
    <color name="teal_700">#0059c1</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>

    <color name="red">#F44336</color>
    <color name="gray">#CCCCCC</color>
</resources>
```

APÊNDICE 44 – Pasta values - google_maps_api.xml

```
<resources>
    <!--
   TODO: Antes de executar seu aplicativo, você precisa de uma chave de API do Google Maps.

     Para obter um, siga este link, siga as instruções e pressione "Criar" no final:


https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=73:9A:60:A6:C3:94:02:D9:A5:C6:9F:AC:B0:70: 3B:3B:DB:50:CD:7E%3Bcom.example.geofenceapp

     Você também pode adicionar suas credenciais a uma chave existente, usando estes valores:

     Nome do pacote:
     com.example.gefenceapp

     Impressão digital do certificado SHA-1:
     73:9A:60:A6:C3:94:02:D9:A5:C6:9F:AC:B0:70:3B:3B:DB:50:CD:7E

     Alternativamente, siga as instruções aqui:
     https://developers.google.com/maps/documentation/android/start#get-key

     Depois de ter sua chave (começa com "AIza"), substitua o "google_maps_key" Keu neste arquivo.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false"> google_maps_key</string>
</resources>
```

APÊNDICE 45 – Pasta values - strings.xml

```
<resources>
    <string name="app_name">GeofenceApp</string>

    <string name="permission_description">Para usar este aplicativo, primeiro precisamos solicitar permissão de localização de você. Caso contrário, não poderemos fornecer-lhe o nosso serviço. Se você concorda e deseja continuar, toque no botão abaixo.</string>
    <string name="permission_required">Permissão necessária</string>
    <string name="continue_btn">Continuar</string>
    <string name="back">Voltar</string>
    <string name="next">Próximo</string>
    <string name="name">Nome</string>
    <string name="geofence">Criando Geofence</string>
    <string name="one_three">1/3</string>
    <string name="two_three">2/3</string>
    <string name="location">Localização</string>
    <string name="powered_by_google">Powered by Google</string>
    <string name="done">Fim</string>
    <string name="three_three">3/3</string>
    <string name="_50_0_m">50.0 m</string>
    <string name="choose_a_radius_for_your_geofence">Escolha um raio para sua cerca geográfica.</string>

    <string name="display_meters">%1$s m</string>
    <string name="display_kilometers">%1$s km</string>
    <string name="long_press_on_the_map_n_to_add_a_geofence">Pressione e segure no mapa \n para adicionar uma cerca geográfica.</string>
    <string name="no_geofence_found">Nenhuma Geofence encontrada.</string>
    <string name="radius">Raio:</string>
    <string name="coordinates">Coordenadas:</string>
    <string name="location_txt">Localização:</string>
</resources>
```

APÊNDICE 46 – Pasta values - themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.GeofenceApp"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/blue_500</item>
        <item name="colorPrimaryVariant">@color/blue_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

APÊNDICE 47 – Pasta values - themes.xml/nigth

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.GeofenceApp"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/blue_200</item>
        <item name="colorPrimaryVariant">@color/blue_700</item>
        <item name="colorOnPrimary">@color/black</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

APÊNDICE 48 – geofences_row_layout_scene.xml

```
<?xml version="1.0" encoding="utf-8"?>
<MotionScene
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:motion="http://schemas.android.com/apk/res-auto">

   <Transition
      motion:constraintSetEnd="@+id/end"
      motion:constraintSetStart="@id/start"
      motion:duration="1000">
     <KeyFrameSet>
     </KeyFrameSet>
      <OnSwipe
         motion:touchAnchorId="@+id/white_background"
         motion:dragDirection="dragLeft"
         motion:touchAnchorSide="right" />
   </Transition>

   <ConstraintSet android:id="@+id/start">
   </ConstraintSet>

   <ConstraintSet android:id="@+id/end">
      <Constraint
         motion:layout_constraintEnd_toEndOf="parent"
         android:layout_width="wrap_content"
         android:layout_height="120dp"
         motion:layout_constraintBottom_toBottomOf="parent"
         motion:layout_constraintHorizontal_bias="0.5"
         motion:layout_constraintTop_toTopOf="parent"
         motion:layout_constraintStart_toStartOf="parent"
         android:id="@+id/white_background"
         android:layout_marginEnd="200dp" />
   </ConstraintSet>
</MotionScene>
```

APÊNDICE 49 – build.gradle (Project: GeofenceApp)

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.2.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.6.10"
        classpath "com.google.dagger:hilt-android-gradle-plugin:2.40.5"
        classpath "androidx.navigation:navigation-safe-args-gradle-plugin:2.5.0-alpha03"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

APÊNDICE 50 – build.gradle (Modulo: GeofenceApp.app)

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-kapt'
    id 'dagger.hilt.android.plugin'
    id 'androidx.navigation.safeargs.kotlin'
    id 'kotlin-parcelize'
}

android {
    compileSdkVersion 31
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.geofenceapp"
        minSdkVersion 21
        targetSdkVersion 31
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildFeatures {
        viewBinding true
        dataBinding true
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles     getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_11
        targetCompatibility JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = '11'
    }
}

dependencies {

//Bibliotecas de navegação e demais
```

```
    implementation 'androidx.core:core-ktx:1.8.0'
    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.navigation:navigation-fragment-ktx:2.5.2'
    implementation 'androidx.navigation:navigation-ui-ktx:2.5.2'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.android.gms:play-services-maps:18.1.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    // Ciclo da vida
    implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1"
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.5.1"
    implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.5.1"

    // Coroutines
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.6.1'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.1'

    // Bobina - Biblioteca de Carregamento de Imagens
    implementation("io.coil-kt:coil:1.3.2")

    // Places SDK Locais
    implementation 'com.google.android.libraries.places:places:2.6.0'
    implementation 'com.android.volley:volley:1.2.1'

    // Easy Permissions (Permissões fáceis)
    implementation 'com.vmadalin:easypermissions-ktx:1.0.0'

    // Componentes da Room
    implementation "androidx.room:room-runtime:2.4.3"
    kapt "androidx.room:room-compiler:2.4.3"
    implementation "androidx.room:room-ktx:2.4.3"
    androidTestImplementation "androidx.room:room-testing:2.4.3"

    // Dagger - Hilt
    implementation "com.google.dagger:hilt-android:2.38.1"
    implementation "androidx.hilt:hilt-lifecycle-viewmodel:1.0.0-alpha03"
    kapt "com.google.dagger:hilt-android-compiler:2.38.1"
    kapt "androidx.hilt:hilt-compiler:1.0.0"

    // Banco de dados
    implementation "androidx.datastore:datastore-preferences:1.0.0"

    // Utilidade
    implementation 'com.google.maps.android:android-maps-utils:2.2.0'
}
```