PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

VALIDAÇÃO PARA GEORREFERENCIAMENTO POR MEIO DE IMPRESSÃO DIGITAL

ANTÔNIO LAGO TEIXEIRA NETO

GOIÂNIA
2021

ANTÔNIO LAGO TEIXEIRA NETO

VALIDAÇÃO PARA GEORREFERENCIAMENTO POR MEIO DE IMPRESSÃO DIGITAL

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos Requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Marcelo Antonio Adad de Araújo

GOIÂNIA

2021

ANTÔNIO LAGO TEIXEIRA NETO

VALIDAÇÃO PARA GEORREFERENCIAMENTO POR MEIO DE IMPRESSÃO DIGITAL

Este trabalho de Conclusão de Curso julgado adequado para obtenção o título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da pontifícia Universidade Católica de Goiás, em 07 / 06 / 2021.

_____

Profa. Mestra Ludmila Reis Pinheiro dos Santos
Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

_____

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

_____

Membro 1: Prof. Mestre Carlos Alexandre Ferreira de Lima, M.E.E.

_____

Membro 2: Prof. Fernando Gonçalves Abadia, M.E.

GOIÂNIA
2021

AGRADECIMENTOS

A Deus por ter me dado saúde e força para vencer todas as dificuldades pelas quais eu passei.

A minha mãe Alexandra de Araújo, a meu pai Antônio Carlos, e a minha avó Vilma de Araújo por terem me dado o suporte familiar de que precisei durante a vida.

A minha namorada Juliana Cardoso por ter me ajudado nos momentos difíceis e me apoiado quando precisei.

Ao professor Marcelo Antonio Adad de Araújo, orientador acadêmico, pela confiança e pelo suporte depositados em minha pessoa.

RESUMO

Datiloscopia ou papiloscopia é a área do conhecimento que estuda a identificação de pessoas pela impressão digital. É uma das formas mais seguras de se identificar uma pessoa, já que não há nenhuma impressão digital igual a outra, tornado uma das maneiras mais difundidas de identificação individual, estando presente em documentos de identificação e em várias aplicações em que se necessite validar a identidade do usuário. O objetivo deste trabalho é o estudo e o desenvolvimento de um aplicativo para validar através da impressão digital a identidade dos usuários de um sistema de georreferenciamento para assim aumentar o nível de segurança desta aplicação. O aplicativo é desenvolvido na linguagem Kotlin por meio da IDE Android Studio e do Android SDK Tools. Os dados necessários para a análise foram obtidos por meio do desenvolvimento de uma aplicação para o sistema operacional Android, utilizando uma API de impressão digital. O desenvolvimento do aplicativo demonstrou a possibilidade de integração e do aumento de segurança para aplicação de georreferenciamento.

Palavras-Chave: 1. Impressão digital, 2. Android, 3. Georreferenciamento.

ABSTRACT

Dactyloscopy or papilloscopy is the area of knowledge that studies the identification of people by fingerprint. Is one of the safest ways to identify a person, since there is no fingerprint like another, it has become one of the most widespread ways of identification being present in identification documents and in various applications in which it is necessary to validate the user's identity. The objective of this work is to study and develop a application to validate through the fingerprint the identity of users of a georeferencing system in order to increase the security level of this application. The application is being developed in Kotlin language through the IDE Android Studio and the Android SDK Tools. The data necessary for the analysis were obtained through the development of an application for the Android operating system, using a fingerprint API. The development of the application demonstrated the possibility of integration and increased security for the application of georeferencing.

Keywords: 1. Fingerprint, 2. Android, 3. Georeferencing

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

API	Interface de Programação de Aplicações.
Ativ	*Activity*.
CMOS	metal-óxido-semicondutor de simetria complementar.
Frag	*Fragmet*.
IDE	Ambiente de Desenvolvimento Integrado.
M.E.	Mestre em Engenharia.
M.E.E.	Mestre em Engenharia Elétrica.
Ma.	Mestra.
Prof.	Professor.
Profa.	Professora.
SDK	Kit de desenvolvimento de software.
SO	Sistema Operacional.
XML	*Extensible Markup Language*.

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem por objetivo o estudo e desenvolvimento de um aplicativo para o sistema operacional Android que possibilite a validação do usuário por meio de sua impressão digital através de uma API para melhorar o nível de segurança de um sistema de georreferenciamento.

Para o desenvolvimento do aplicativo foram realizados estudos sobre softwares e linguagens, que estão descritos ao longo do trabalho, e serão utilizados para um melhor desenvolvimento do aplicativo.

Foram selecionadas as ferramentas que mais atenderam os melhores requisitos para o projeto.

1.1 Justificativa

Cada ser humano tem características que o definem como indivíduo único, algumas destas características são exclusivas como a íris, a voz, e impressões da pele nas extremidades, destas a mais usada na identificação de indivíduos é a impressão digital, que está presente nos documentos de identificação e em aplicações que necessitam de validação da identidade do usuário. Por esse motivo está forma de validação foi escolhida para ser estudada e ser desenvolvido um aplicativo que faça uso de incremento de segurança de uma aplicação.

1.2 Questões de pesquisa

Será possível o incremento da segurança do aplicativo de georreferenciamento por meio de um aplicativo de validação de usuário por meio de impressão digital?

Partindo-se do que foi estudado até o presente momento observa-se que plenamente possível unir estas tecnologias possibilitando uma melhor segurança para o usuário.

Será possível a integração entre os dois sistemas?

A partir do que foi estudado até o presente momento é possível fazer a integração dos dois sistemas.

1.3 Objetivos

Construir um aplicativo Android capaz de criar *geofences*, gerenciá-las, e validar o usuário por meio da impressão digital.

1.3.1 Objetivo geral

Verificar por meio de impressão digital a identidade do usuário, para juntamente com seu georreferenciamento possibilitar uma melhor identificação deste mesmo usuário.

1.3.2 Objetivos específicos

- Determinar as API's que mais atenda às necessidades do projeto.
- Determinar a IDE que mais atenda à necessidade do projeto.
- Construir a aplicação Android.
- Integrar os sistemas de validação de usuário e o sistema de georreferenciamento.

1.4 Procedimentos metodológicos

Para que seja atingido o objetivo deste trabalho e que sejam respondidas as questões de pesquisa, este trabalho é realizado da seguinte forma:

Desenvolvimento do aplicativo Android através da IDE Android Studio.

Utilização da API de impressão digital da Google para validar o usuário.

Verificar a possibilidade de integração com o sistema de georreferenciamento.

1.5 Organização do trabalho

Neste capítulo é apresentada a introdução ao tema do trabalho as questões a serem respondidas, os objetivos, os procedimentos a serem seguidos e a organização do trabalho.

No capítulo 2 será apresentado o referencial teórico utilizado para elaboração deste trabalho.

No capítulo 3 serão apresentadas as ferramentas utilizadas e os procedimentos realizados para conclusão do trabalho.

No capítulo 4 serão apresentadas as tecnologias e abordagens utilizadas no desenvolvimento do aplicativo Android.

No capítulo 5 serão apresentadas as Consideração finais sobre o trabalho. E após este capítulo as referências bibliográficas e os apêndices.

2. REFERENCIAL TEÓRICO

Neste capítulo é apresentado o referencial teórico utilizado na elaboração deste trabalho, onde são apresentadas informações sobre desenvolvimento de aplicativos Android, noções de impressão digital, e noções de georreferenciamento.

2.1 Sistema operacional Android

O Android é um sistema operacional de código aberto desenvolvido pela Android Inc., esta por sua vez foi adquirida pela Google em 2005. Em 2007, foi formada a *Open Handset Alliance*™ da qual a Google faz parte como companhia de software. Esta aliança de empresas operadoras de telefonia, fabricantes de aparelhos, companhias de semicondutores, companhias de software, e companhias de comercialização que têm por missão melhorar a experiência dos consumidores de aparelhos móveis.

A figura 1 representa de forma esquemática a arquitetura do SO Android, e será explicado ao longo do documento seguinte.

Figura 1 - Arquitetura do SO Android.

Fonte: **Arquitetura Android**. (*Android Open Source Project*, 2020)

Estrutura do aplicativo é uma parte em que os aplicativos fazem suas solicitações de serviços ao SO.

Binder Inter-Process Comunication é um mecanismo de intercomunicação entre as solicitações do aplicativo e os serviços do sistema. Esse mecanismo é oculto ao desenvolvedor de aplicativos, fazendo parecer que suas solicitações são atendidas diretamente pelos serviços do sistema.

Serviços do sistema são módulos de funcionalidade específica que fazem o acesso ao hardware. Estes são divididos em dois grupos mídia (como defletor de áudio e serviços de câmera) e sistema (gerenciador de janelas e serviço de busca).

A camada de abstração de hardware serve de interface padrão para os fornecedores de *hardware*, o que permite que o Android não tenha conhecimento do que foi implementado em

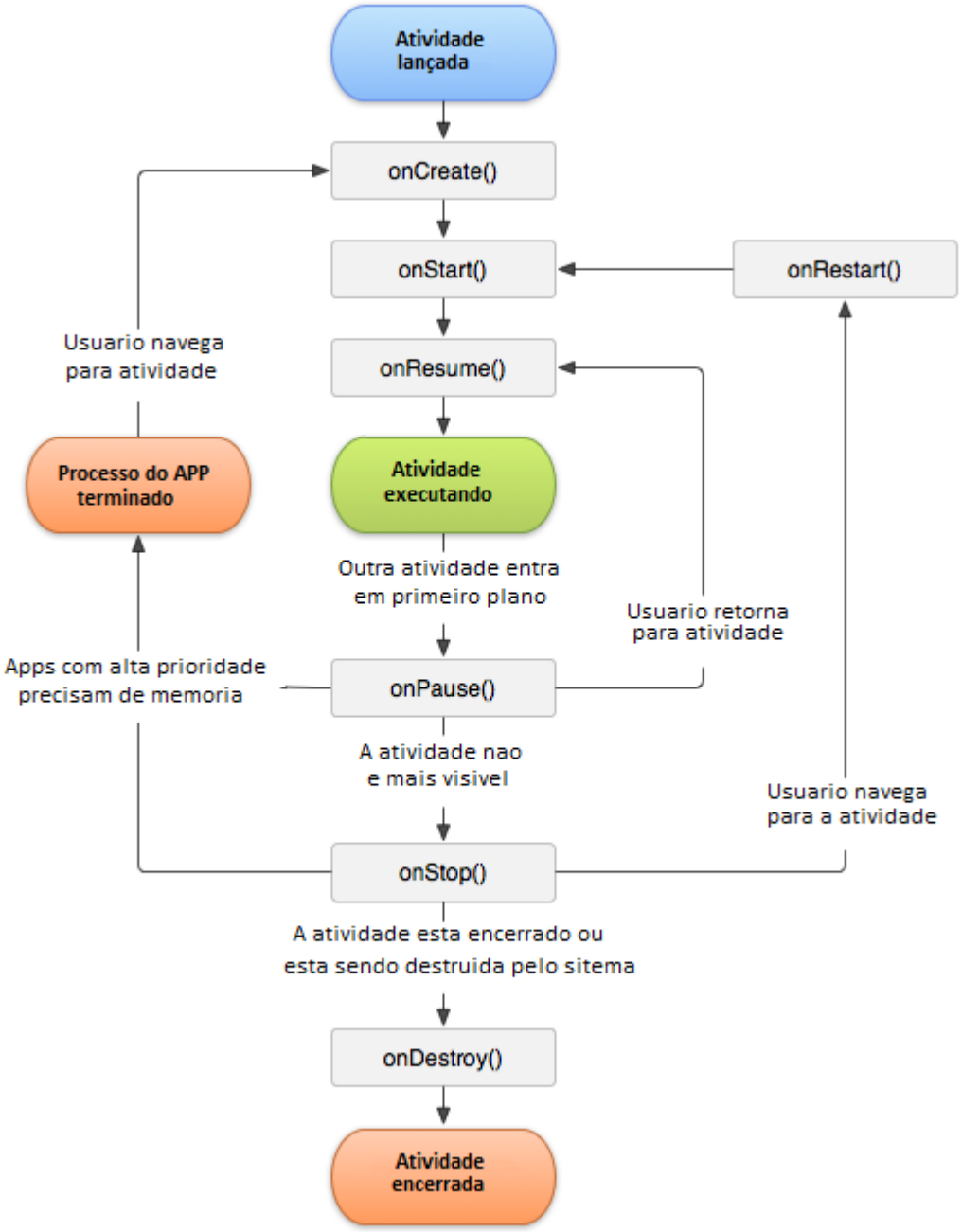
hardware, assim permitindo a implementação de diversos módulos de *hardware* sem alterar o sistema de nível superior.

O Android utiliza um Kernel Linux otimizado para a plataforma móvel, sendo que essa otimização não afeta o desenvolvimento dos driver. *(GOOGLE, 2020)*

A interface de usuário do Android é feita por meio da classe *Activity*, doravante chamada neste trabalho de ativ, e utilizando de suas subclasses como a *FragmentActivity* que foi escolhida para ser utilizada nesse trabalho, cria a tela com o qual o usuário vai interagir.

A ativ tem um ciclo de vida delimitado por funções que fazem a manutenção da ativ desde a sua criação até a sua destruição. Estas funções são *onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop(), onDestroy()*. Algumas dessas funções são criadas automaticamente quando o arquivo da ativ é criado outras, caso seja necessário modificar seu funcionamento, podem ser incluídas durante o desenvolvimento, e o seu funcionamento é demostrado pelo fluxograma da Figura 2. (ANDROID OPEN SOURCE PROJECT, 2021)

Figura 2 – Fluxograma da classe *Activity*.



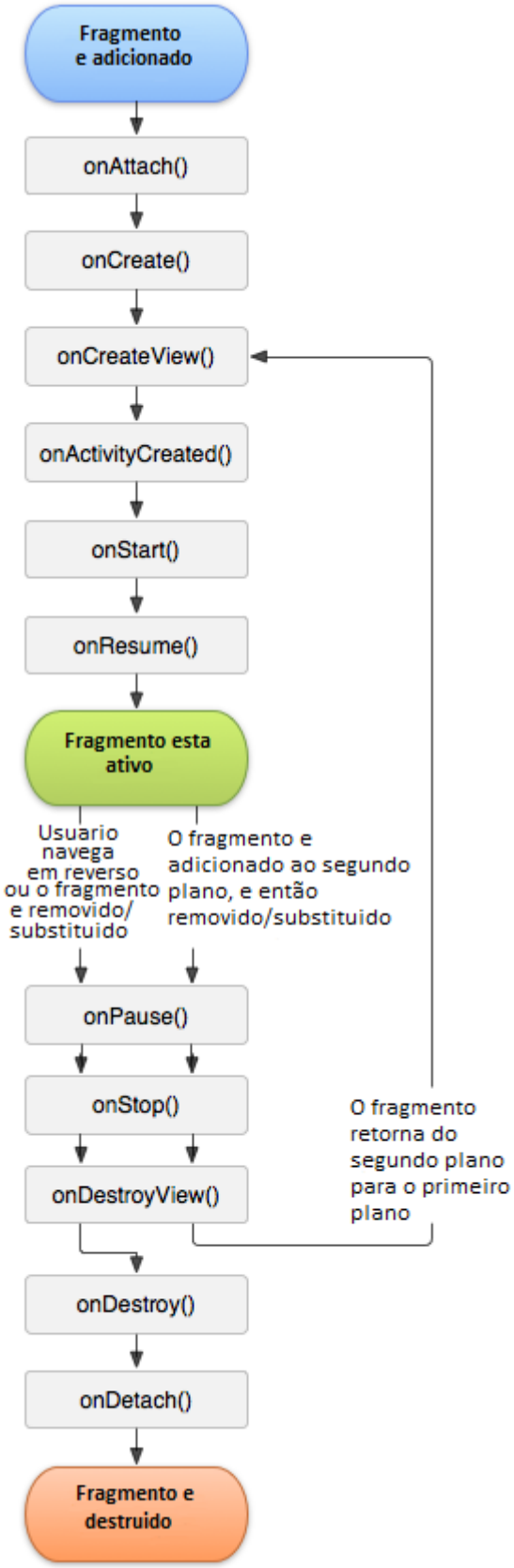
Fonte: Modificada de **Activity**. (ANDROID OPEN SOURCE PROJECT, 2021)

A função *onCreate()* é chamada pelo sistema. Quando a ativ é criada tudo que for escrito dentro desta função será executado durante a criação da ativ, todo o ciclo de vida da ativ fica entre as funções *onCreate()* e *onDestroy()*, que é chamada quando o a ativ é encerrada, a função *onStart()* é chamada quando a ativ se torna visível ao usuário e, por sua vez, esta fica visível até a chamada da função *onStop()* que encerra a visualização da ativ. Essa função normalmente é chamada quando uma nova ativ está sendo inicializada, e a ativ atual está a caminho de ser

encerrada. A função *onRestart()* é chamada quando a ativ foi parada e o usuário navega novamente para essa ativ, esta retorna e chama a função *onStart(),* a função *onResume()* é chamada quando a ativ começa a ser interativa para com o usuário neste ponto ela é a ativ que está no topo da pilha de ativs. A função *onPause()* é chamada quando perde seu estado de primeiro plano esta função pode ser seguida pela *onResume()* caso ativ retorne ou pela *onStop()* caso fique invisível ao usuário. (ANDROID OPEN SOURCE PROJECT, 2021)

A subclasse *FragmentActivity* pode fazer uso da classe *Fragment*, doravante neste trabalho chamada de frag, que permite que o aplicativo em desenvolvimento seja mais modular possibilitando que uma única ativ inclua múltiplos frags, a classe frag foi introduzida no SO Android a partir do da versão 3.0. Quando se utiliza frags a classe ativ fica apenas com sua função *onCreate()* sendo todo o funcionamento da interface gerenciado dentro do frag que será inflado na ativ, a função *onAttach()* é chamada quando o frag é anexado à ativ, sendo chamada a função *onDetach()* quando o frag é desanexado da ativ, as funções *onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop()* e *onDestroy()* são chamadas nos memos casos descritos na ativ porém agora desempenhando suas funções dentro da classe frag, a função *onCreateView()* é chamada quando o frag é instanciado como a interface de visualização do usuário. Se está função for chamada, é necessário que seja chamada a função *onDestroyView(),* a função *onActivityCreated()* é chamada quando a ativ já foi criada e o frag já foi hierarquicamente instanciado porém está função foi descontinuada sendo substituída pela função *onViewCreated(),* os frags ficam à mercê do ciclo de vida da ativ. Quando esta é destruída os frags também são destruídos. O frag também possui seu próprio ciclo de vida independente da ativ podendo ser pausado ou destruído sem interferir no funcionamento da ativ, assim como a ativ o frag tem suas funções que fazem sua manutenção, e seu ciclo de vida é demonstrado na figura 3. (ANDROID OPEN SOURCE PROJECT, 2019)

Figura 3 – Fluxograma da classe *Fragmet*.



Fonte: Modificada de **Fragmentos**. (ANDROID OPEN SOURCE PROJECT, 2019)

2.2 Android SDK

O Android SDK é um pacote de ferramentas essenciais para o desenvolvimento de aplicativos para o sistema Android. Dentro desse pacote tem-se as bibliotecas, as documentações, códigos de exemplo, tutoriais do Android, o depurador, e o simulador baseado em QEMU. (VAATI, 2020)

2.3 Interface de Programação de Aplicações (API)

São padrões de comunicação entre produtos e serviços distintos que retiram a necessidade de um entendimento pleno da implementação um do outro.
Uma das principais vantagens do uso de uma API é a dispensabilidade de se reescrever uma função a cada novo programa que necessite da mesma. Outra vantagem é que o desenvolvedor da API pode manter a sua solução oculta e disponibilizar para uso apenas os padrões de comunicação. (RED HAT, 2020)

2.4 Linguagem de programação e de marcação

Para a escrita do código desenvolvido foi escolhida a Linguagem Kotlin pois esta é compatível com o desenvolvimento Android e corrigiu vários problemas da linguagem Java (que é a linguagem de desenvolvimento original do Android) como a execução do ponteiro vazio e a excessiva verbosidade encontrada no código.

Kotlin é uma linguagem estaticamente tipada o que resulta em uma compilação mais rápida pois o compilador não precisa identificar qual o tipo das variáveis do código. (MOSKALA, 2017)

Uma linguagem de marcação é utilizada para demarcar onde cada elemento gráfico será exibido dentro de uma interface. A linguagem de marcação utilizada neste projeto foi a *Extensible Markup Language* ou XML. Esta é a linguagem de marcação utilizada para desenvolvimento da interface de aplicativos nativos do sistema Android.

2.5 Impressão Digital

Até alguns animais possuem característica únicas que os distinguem entre seus semelhantes e podem ser associadas com as impressões digitais do ser humano, como as

impressões nasais de cães, bovinos, caprinos, ovinos, entre outras impressões de outras espécies animais.

Existem relatos muito antigos do uso da impressão digital para identificação de indivíduos, porém uma das primeiras pesquisas científicas sobre a digital humana foi uma observação por meio de microscópio da digital por Marcello Malphighi em 1986 onde observou o relevo da digital.

Johannes Evangelista Purkinje publicou uma tese em 1823 onde citava nove padrões de impressões digitais.

Em 1894 o Fances Alphonse Bertillon adicionou a impressão digital a seu sistema de identificação.

William James Herschel fez uso da impressão da palma da mão em um acordo comercial na Índia. Após o acordo ser bem-sucedido, Herschel passou a utilizar impressão da palma e posteriormente do dedo médio em seus contratos, após observar seus contratos acumulados durante anos, estava convicto de que impressões digitais poderiam ser usadas na identificação de indivíduos. Em 1880 publicou no diário científico “Nature” um artigo falando de suas experiências.

Em 1870 o cirurgião britânico Henry Faulds observou impressões digitais em cerâmicas pré-históricas e após estudos reconheceu a importância do uso de impressões digitais para a identificação de indivíduos, assim como criou um método de classificá-las. Em 1880 publicou um artigo no diário científico “Nature” discutindo sobre o uso de impressões digitais para identificação e sobre o uso de tinta para a obtenção delas em papel, por exemplo.

Em 1880 o antropólogo britânico Francis Galton começou seu trabalho em impressões digitais com base nos trabalhos de Herschel e Faulds. Em 1892 publicou seu livro sobre impressões digitais. Galton comprovou cientificamente que as impressões digitais não se alteravam durante a vida e que não há duas impressões digitais iguais além de identificar as características usadas para a identificação de impressões digitais. Essas características são as minúcias, e podem também ser chamadas de detalhes de Galton.

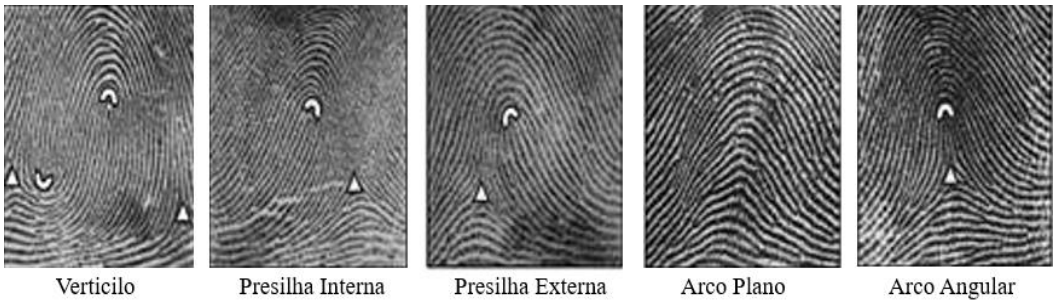
Em 1897 Sir. Edward Richard Henry criou o Sistema de Pontos singulares ou Detalhes Globais. (MÁRCICO,2002)

O Sistema de identificação das impressões digitais pode ser dividido em três abordagens: global (ou sistema de pontos singulares), local (ou minúcias), e fina.

2.5.1 *Abordagem Global*

A abordagem global ou sistema de pontos singulares utiliza para a identificação 5 classes sendo elas o arco plano, arco angular, presilha interna, presilha externa, e verticilo como pode ser visto na figura 4, sendo esta a abordagem mais comum de identificação das impressões digitais pois utiliza de estruturas maiores para a identificação, porém esta é a menos segura entre as três. (SOARES,.2009)

Figura 4 - Pontos Singulares.

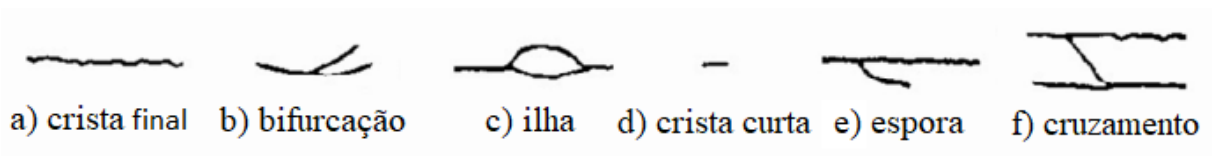


Fonte: Modificada de (MISHRA; DEHURI, 2019)

2.5.2 *Abordagem Local*

A abordagem local utiliza-se das minúcias que são padrões obtidos pela observação do comportamento das cristas da impressão digital sendo 6 esses padrões: a bifurcação, a crista final, a ilha, a crista curta a espora e o cruzamento, como pode ser visto na figura 5. Esta forma de identificação é a segunda mais usada pois garante mais segurança na identificação, porém a imagem gerada precisa ser de uma melhor resolução. (SOARES,.2009)

Figura 5 - Minúcias.



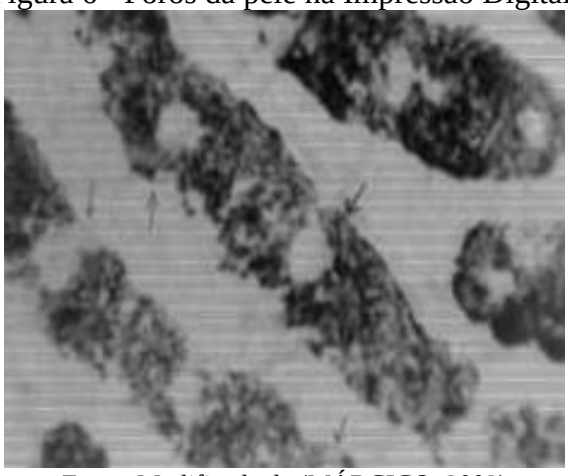
Fonte: Modificada de (Galton, 1982)

Uma outra abordagem utilizada é uma junção da abordagem global e da abordagem local assim aumentando o nível de segurança da identificação da impressão digital e reduzindo o número de correspondências de minúcias necessárias.

2.5.3 *Abordagem Fina*

A abordagem fina se utiliza, para identificação do padrão, dos poros na impressão digital. Esta abordagem é a menos usada dentre as três pois necessita de uma imagem de muito alta resolução capaz de identificar os poros da pele que medem cerca de 60 microns como pode ser visto na figura 6. Porém esta é a abordagem mais segura. (SOARES,.2009)

Figura 6 - Poros da pele na Impressão Digital.

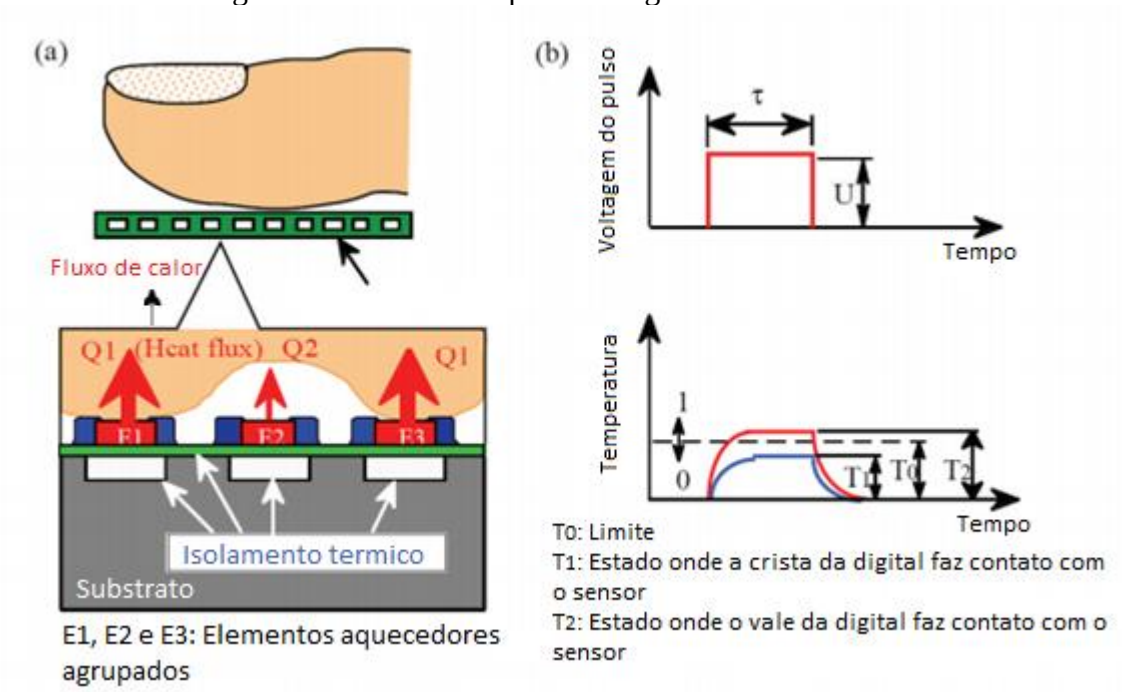


Fonte: Modificada de (MÁRCICO, 2002)

Existem quatro tipos de sensores para a obtenção da impressão digital. Estes sensores são: resistivo, capacitivo, óptico e ultrassônico.

O sensor resistivo tem seu princípio de funcionamento baseado em termorresistores, que identificam a variação entre a temperatura das cristas e dos sulcos ou vales na impressão digital assim montando uma impressão digital como pode ser visto na figura 7. Esse tipo de sensor quase não é usado pois tem uma precisão baixa quando a temperatura está alta e a diferença de temperatura entre os vales e as cristas da impressão é muito pequena. (LU, 2015)

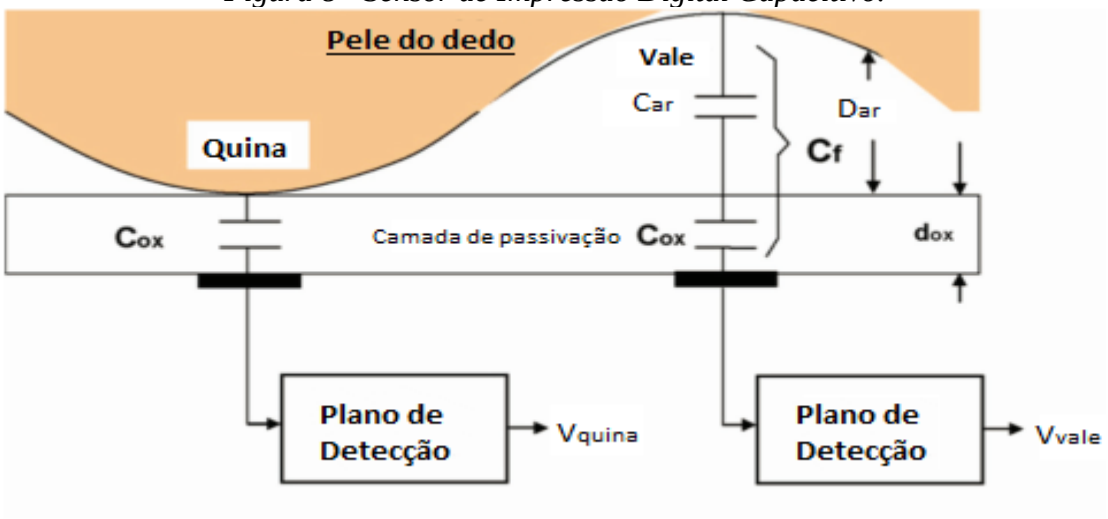
Figura 7 - Sensor de Impressão Digital Termorresistivo.



Fonte: Modificada de (LU, 2015)

O sensor capacitivo é o mais comum pois é baseado na tecnologia de circuito integrado CMOS. O sensor funciona com base em um vetor de eletrodos no padrão de processamento CMOS coberto por uma camada dielétrica. Quando o dedo é colocado sobre a superfície do sensor, a capacitância se altera entre a crista e o vale da digital, assim gerando uma diferença de tensão entre a crista e o vale permitindo que se monte uma impressão digital, como pode ser visto na figura 8. O sensor capacitivo está em desuso em smartfones desde 2018, sendo substituído por sensores ópticos e ultrassônicos pois sofre muita interferência de umidade e temperatura. (LU, 2015)

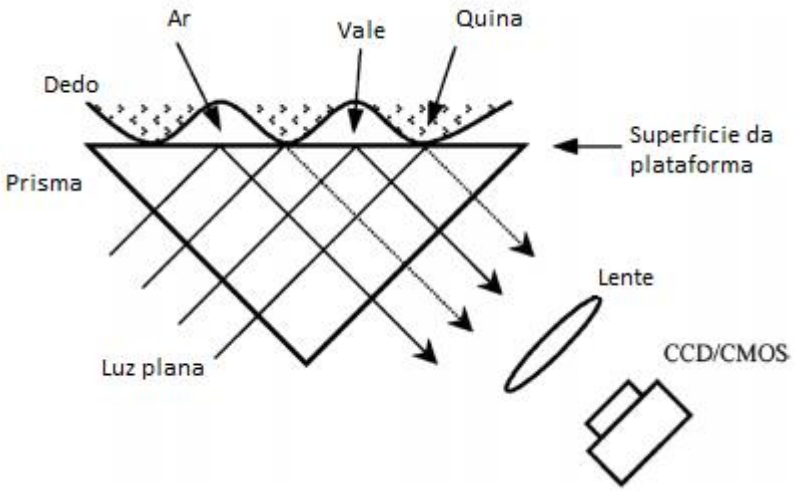
Figura 8 - Sensor de Impressão Digital Capacitivo.



Fonte: Modificada de (LU, 2015)

O sensor óptico funciona com base no fenômeno de reflexão interna total frustrada. Este fenômeno se dá quando o prisma emissor de luz tem um índice de refração similar ao da pele humana e o ângulo entre o feixe de luz e o vidro seja tal que quando em contato com o ar no vale da digital seja totalmente refletido, então, quando a luz entra em contato com a crista da digital é apenas parcialmente refletida assim sendo possível gerar uma impressão digital como pode ser visto na figura 9. Porém os sensores ópticos podem sofrer interferência luminosa em ambientes externos e baixa qualidade na imagem por causa da falta de umidade nos dedos. (LU, 2015)

Figura 9 - Sensor de Impressão Digital Óptico.

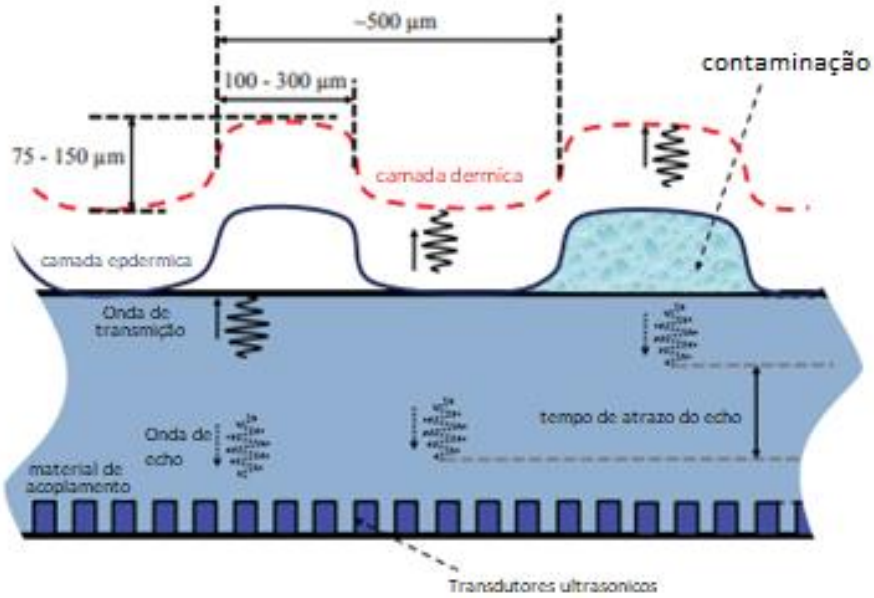


Fonte: Modificada de (LU, 2015)

O sensor ultrassônico utiliza ondas sonoras para mapear a digital identificando a diferença da distância entre o retorno das ondas sonoras em contato com as cristas e com os

vales da digital, como pode ser visto na figura 10. O sensor ultrassônico foi muito aclamado por sua possibilidade de ser mais seguro e eficiente que o sensor óptico, porém quando foi instalado no smartfone da Samsung S10 houve uma falha de segurança em que foi possível fazer uma réplica 3D de uma digital e usá-la para desbloquear o aparelho, sendo assim a empresa voltou a usar sensores ópticos em seus novos smartfones. (LU, 2015)

Figura 10 - Sensor de Impressão Digital Ultrassónico



Fonte: Modificada de *(LU, 2015)*

2.6 Localização Geográfica

Localizar-se no espaço é um problema enfrentado não somente pelo ser humano, a natureza como um todo enfrenta esse problema. Os animais desenvolveram muitas formas de se localizar como sonares dos golfinhos, brilho da lua para as mariposas, ondas ultrassônicas para os morcegos, marcadores químicos para as formigas, entre outros.

Os navegadores durante a época das grandes navegações precisaram se localizar no meio do oceano para chegar ao destino desejado, porém por meios visuais e sem nenhum instrumento isso não era possível, por isso eram utilizadas bússolas para marcar onde ficava o norte magnético da terra e o astrolábio para medir a longitude e a latitude com base no ângulo que o horizonte formava com um astro, como uma estrela.

A latitude e a longitude atualmente são demarcadas em um sistema de coordenadas geográficas que utiliza de um modelo 3D para determinar locais no globo terrestre.

As latitudes então divididas em linhas paralelas que circundam horizontalmente a terra e que vão de -90º a 90º sendo a latitude 0 a linha do equador.

As longitudes são formadas por linhas que circundam a terra verticalmente e vão de -180º a 180º sendo por convenção a longitude 0º o meridiano de Greenwich. É possível marcar qualquer ponto na superfície da terra com base nas latitudes e nas longitudes. (ALVES, 2018)

Com o surgimento de novas tecnologias de localização entre elas o sistema de posicionamento global ou GPS desenvolvido pelos Estados Unidos da América e o desenvolvimento de aplicativos baseados no uso de sistemas de localização global com o Google Maps, o Waze e o Uber. Porém existem outros concorrentes do sistema GPS como o Glonass Russo, o Beidou Chinês, que ficou plenamente operacional em 2020, e o Galileo Europeu.

O GPS é um sistema desenvolvido originalmente para uso militar pela força aérea e pela marinha dos Estados Unidos para auxiliar na movimentação de tropas, localização de inimigos, guiar mísseis entre outras aplicações. Em 1980 o presidente Ronald Regan autorizou o uso civil da tecnologia.

O GPS possui 24 satélites em operação e 4 satélites de reserva, e é programado para que de qualquer ponto do planeta seja possível ter acesso a pelo menos 4 satélites. Apenas 3 satélites são necessários para se calcular a altitude, latitude, e longitude, porém para que não ocorra erros no cálculo de tempo são utilizados 4 satélites. (CARVALHO, 2009)

Geofence é uma aplicação que faz uso da tecnologia GPS para criar fronteiras virtuais com base em coordenadas de um mapa digital. E quando um dispositivo monitorado adentra ou sai desta fronteira desencadeia uma ação designada pelo desenvolvedor da aplicação. (GOOGLE, 2020)

Dependendo da API utilizada, a fronteira pode ser uma área circular delimitada pelas coordenadas latitude, longitude, ou uma área customizável delimitada por vários pontos. A API utilizada no trabalho de referência foi a API do Google que utiliza uma área circular.

3. APRESENTAÇÃO DO TRABALHO

Neste capítulo serão apresentados os processos e ferramentas utilizados para realização deste projeto com base no referencial teórico.

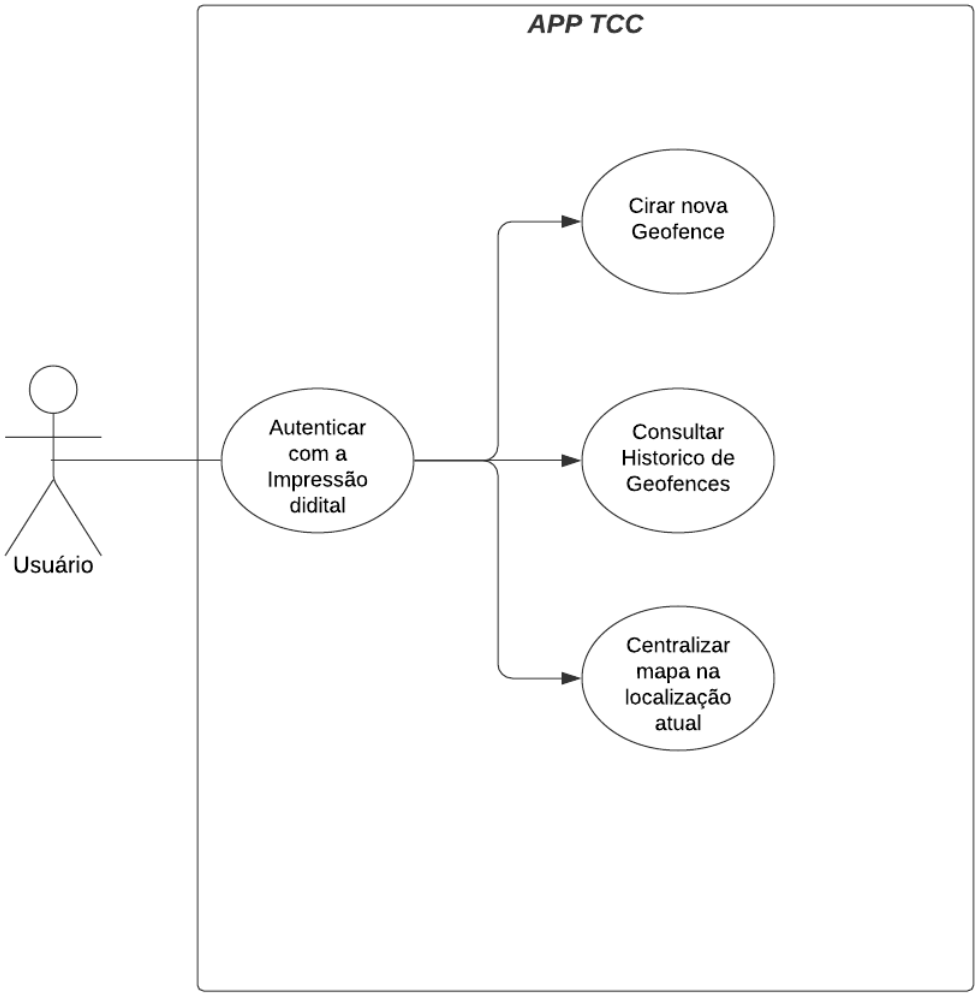
Este trabalho tem por objetivo a criação de um aplicativo de *geofence* que tenha o seu usuário validado por meio de sua impressão digital.

3.1 Aplicativo Android

O aplicativo desenvolvido integra a validação do usuário por meio de sua impressão digital e a criação e o monitoramento de *geofences*.

Na figura 11 é apresentado o diagrama de caso de uso do aplicativo, onde são demonstradas as opções que o usuário tem para utilizar o aplicativo após autenticar a identidade, com a impressão digital, de forma que o usuário navegue pelo mapa centralizando na sua posição atual, criar uma nova *geofence*, ou verificar se alguma já foi criada.

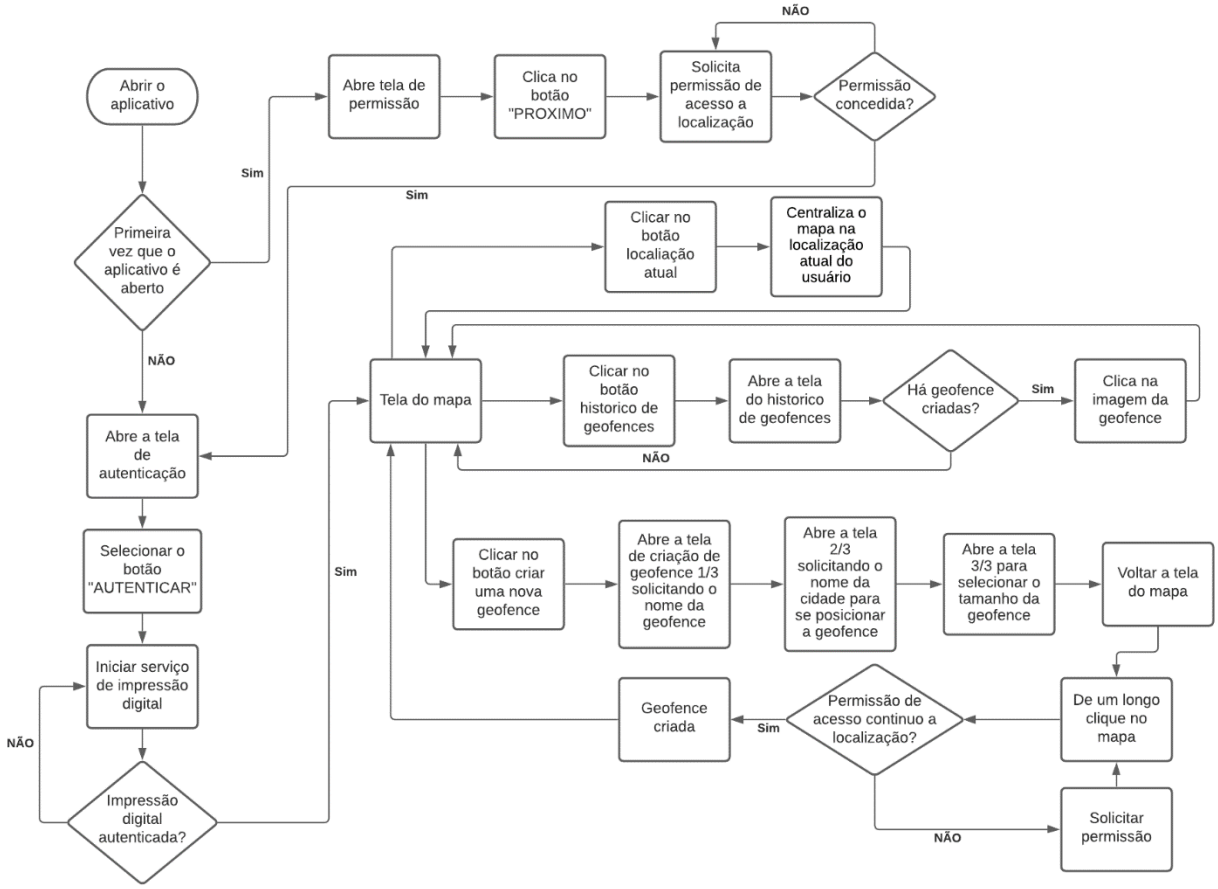
Figura 11 – Caso de uso



Fonte: Elaborado pelo Autor

A figura 12 é o Fluxograma que demonstra o funcionamento do aplicativo.

Figura 12 – Fluxograma do Aplicativo em *Loop* Infinito



Fonte: Elaborado pelo Autor

Ao abrir o aplicativo, se for a primeira vez que este é aberto, será iniciada a tela de permissão onde se encontra o botão “PRÓXIMO” que se selecionado solicita o acesso ao serviço de localização do aparelho. Se esta for negada, será solicitada novamente a permissão até que seja aceita, se for concedida será aberta a tela de autenticação. Na tela de autenticação se encontra o botão “AUTENTICAR” que se selecionado inicia o serviço de impressão digital, se a impressão for inválida inicia o serviço novamente, caso seja válida é aberta a tela do mapa, onde há três botões, se for selecionado o botão de localização atual, a tela irá centralizar na localização atual do aparelho. Se for selecionado o botão de histórico de *geofences*, será aberta a tela do histórico de *geofences*, se houver *geofences* criadas elas aparecerão na tela onde será possível selecionar uma delas. Se não houver é possível voltar a tela do mapa, ao clicar no botão de criar uma nova *geofence*, onde será aberta a primeira das três telas de criação da *geofence*. Será então solicitado o nome da *geofence*, na segunda tela será solicitado o nome da cidade onde se deseja posicionar a *geofence*, na terceira tela será solicitado a escolha do raio da *geofence*. Após esse procedimento volta-se a tela do mapa, onde será possível dar um clique longo na tela, na posição em que se deseja criar a *geofence*, se a aplicação não tiver a permissão

de acesso à localização será solicitada então, após a permissão ser aceita será possível criar a *geofence*. Uma explicação mais detalhada será realizada nas apresentações das telas.

3.2 Solicitação de Permissão

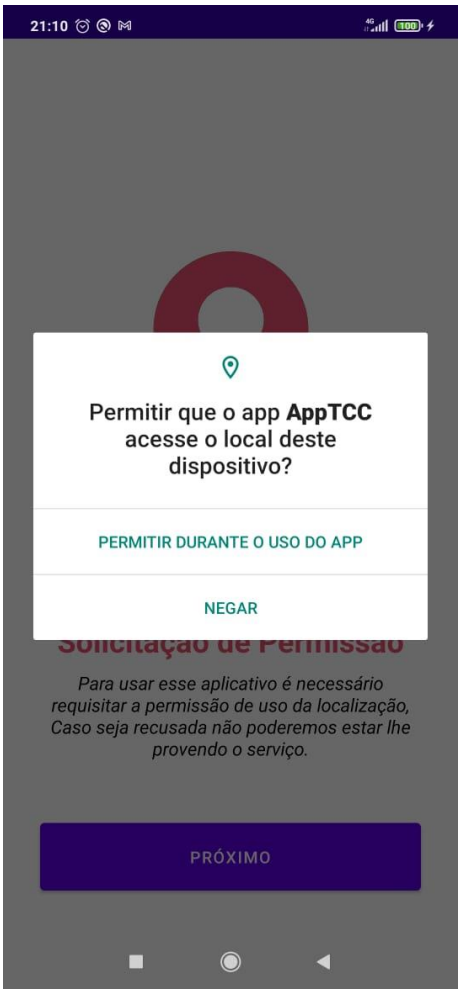
A primeira tela do aplicativo é uma tela que só aparece a primeira vez que ele é instalado já que sua função é apenas solicitar ao usuário que permita o acesso ao serviço de localização do aparelho.

Na tela pode ser encontrada uma mensagem solicitando o acesso e um botão de próximo que na primeira vez que é selecionado abrirá a caixa de solicitação de permissão como pode ser visto nas figuras 13 e 14.

Figura 13 – Tela de solicitação de permissão

Fonte: Elaborado pelo Autor

Figura 14 – Tela de solicitação de permissão com caixa de solicitação



Fonte: Elaborado pelo Autor

Se for negada a solicitação será informado ao usuário que não será possível utilizar o aplicativo sem essa permissão, e será solicitado novamente o uso do serviço de localização do dispositivo.

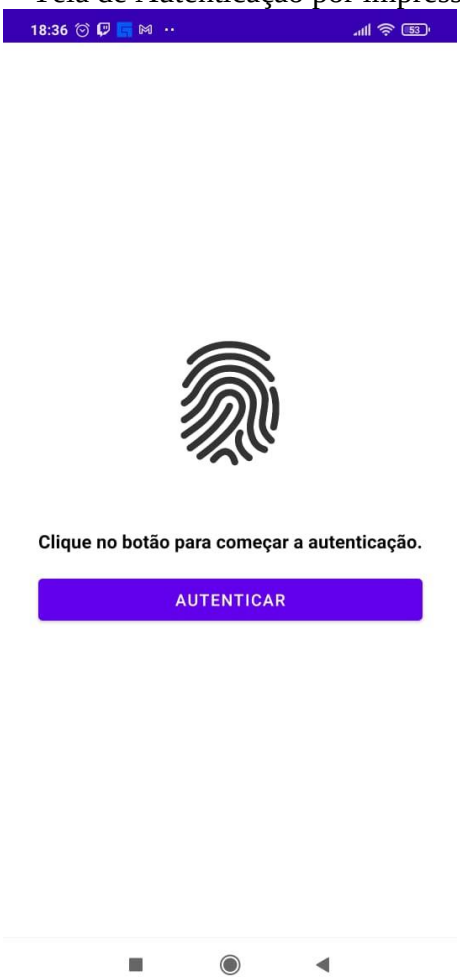
Se a permissão for concedida o botão próximo vai levar o usuário à tela de validação da identidade por impressão digital.

3.3 Validação da identidade do usuário

Para que o usuário seja validado pelo aplicativo é necessário que ele tenha um smartphone com sensor de impressão digital, e que a sua impressão digital esteja cadastrada no sistema de segurança do aparelho.

Na tela de validação da identidade uma mensagem informa ao usuário que toque no botão autenticar, o que irá fazer com que seja iniciada a API *Biometric* do Android que abrirá uma caixa de diálogo solicitando que o usuário toque no sensor de impressão digital. Se o acesso for aprovado o usuário será redirecionado à tela do mapa, como pode ser visto nas figuras 15 e 16.

Figura 15 – Tela de Autenticação por impressão digital



Fonte: Elaborado pelo Autor

Figura 16 – Tela de Autenticação por impressão digital com caixa de solicitação

Fonte: Elaborado pelo Autor

3.4 Mapa

Foi utilizada neste aplicativo a API de mapas da Google pois essa tem suporte nativo tanto no Android quanto na IDE Android Studio, que foi usada no desenvolvimento deste aplicativo e será abordada no tópico Desenvolvimento Android.

Nesta tela tem-se um Mapa interativo e três botões. No canto superior direito há um botão para centralizar a tela na localização atual do usuário, no canto inferior direito há dois botões, o botão que contém uma seta que leva ao histórico de *geofences* já criadas, e o botão com símbolo de mais que leva às telas de criação de uma nova *geofence*. Esta tela pode ser vista na figura 17.

Figura 17 – Tela do Mapa

Fonte: Elaborado pelo Autor

3.5 Criando a *geofence*

Para criar uma nova *geofence* primeiro são necessárias informações que serão solicitadas ao usuário em três telas.

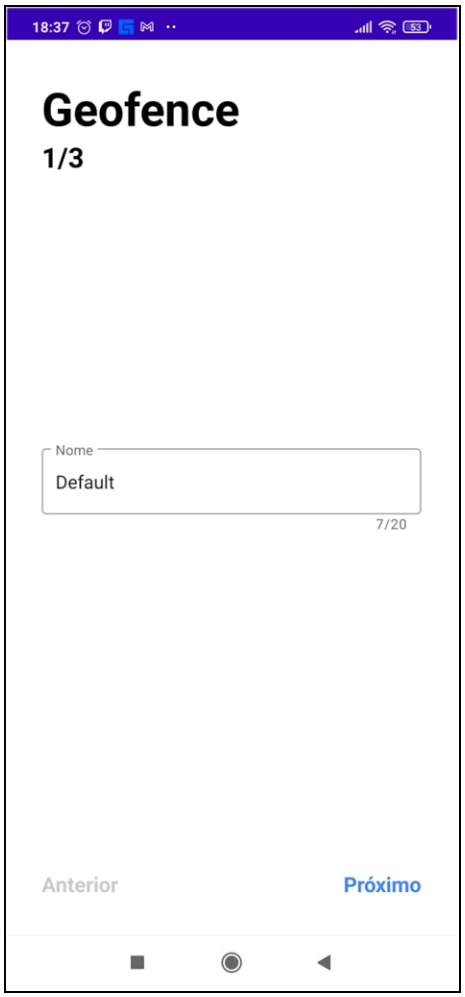
Na primeira tela será solicitado ao usuário que de um nome à *geofence*, caso o usuário não queira nomear a *geofence* o aplicativo está programado a pôr o nome "*Default"* para a *geofence* que em inglês significa padrão, nestas telas tem-se os botões "Anterior" e "Próximo" que levam respectivamente o usuário a tela do mapa e a próxima tela da criação da nova *geofence*, o botão "Próximo" é desabilitado enquanto a caixa de texto com o nome da *geofence* estiver vazia.

Na segunda tela é solicitado ao usuário que pesquise pela cidade onde será criada a nova *geofence*, quando o usuário começa a digitar é executada a API *Places* do Google que

autocompleta a pesquisa com os nomes das cidades do país em que o usuário está localizado, após selecionar um dos resultados da pesquisa o botão “Próximo” fica azul e é habilitado.

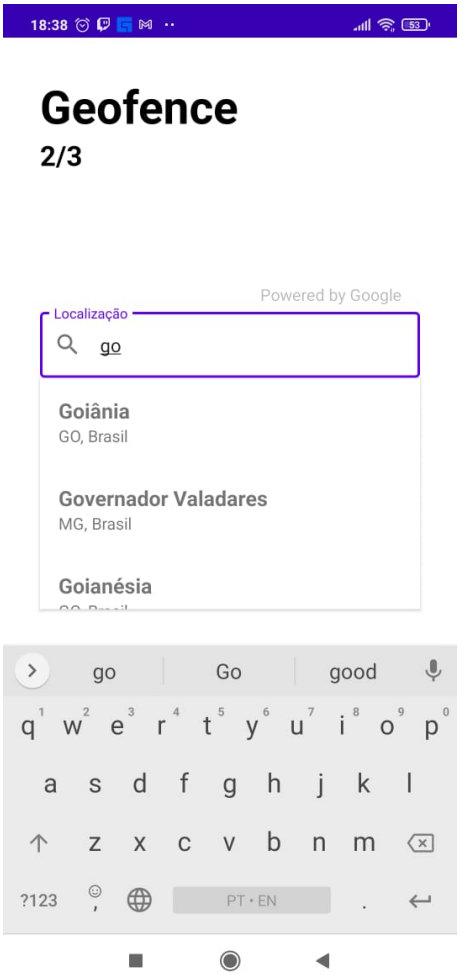
Na terceira tela é solicitado ao usuário que deslize uma barra horizontal para escolher o raio da *geofence*, que neste caso vai de 500 metros a 10 quilômetros, após isso o usuário navega pelo mapa e seleciona com um clique longo na tela um local para a nova *geofence*, caso esse seja a primeira *geofence* que o usuário está criando será solicitado que permita a utilização do serviço de localização o tempo todo, caso o usuário negar essa solicitação será informado que essa aplicação não funciona sem a utilização desta permissão, caso ele aceite será possível criar a *geofence* dado um clique longo na tela no local desejado, estas telas podem ser vistas nas figuras 18, 19, 20, 21, 22 e 23.

Figura 18 – Tela de nomeação da *geofence*



Fonte: Elaborado pelo Autor

Figura 19 – Tela de pesquisa de cidade



Fonte: Elaborado pelo Autor

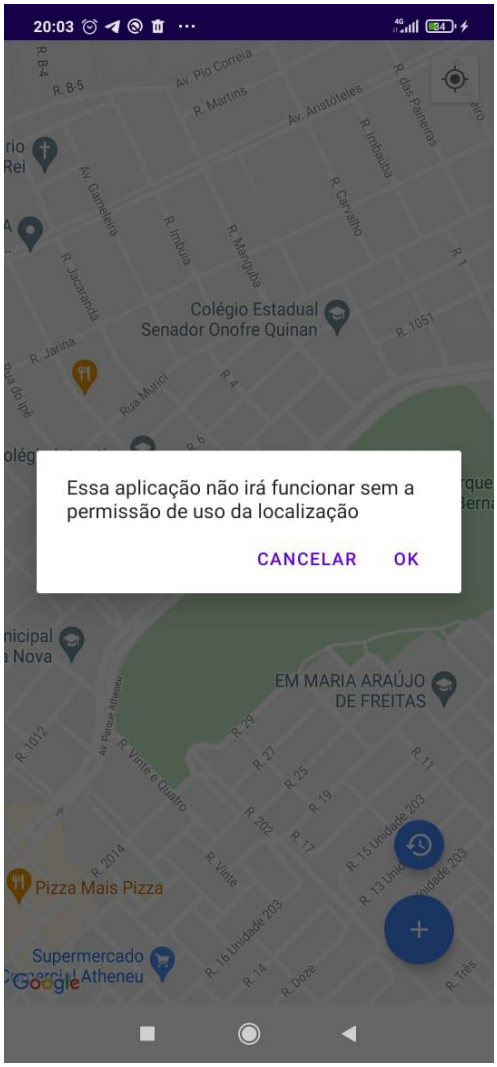
Figura 20 – Tela de seleção do raio da *geofence*

Fonte: Elaborado pelo Autor

Figura 21 – Tela do mapa com solicitação de permissão da localização o tempo todo

Fonte: Elaborado pelo Autor

Figura 22 – Tela do mapa com aviso após o usuário negar a permissão



Fonte: Elaborado pelo Autor

Figura 23 – Tela do mapa com *geofence* criada

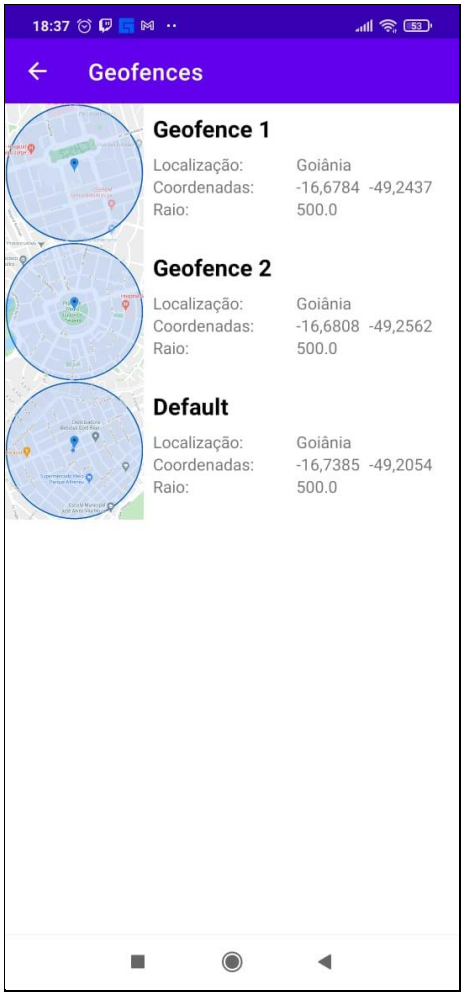
Fonte: Elaborado pelo Autor

3.6 Histórico de *geofences*

A tela de histórico de *geofences* é onde podem ser encontradas todas as *geofences* criadas neste dispositivo, elas são salvas em um banco de dados na memória do aparelho que só pode ser acessado após a primeira *geofence* ser criada.

Após a criação de uma *geofence* ela aparecerá nesta tela com seu nome e duas informações são carregadas do banco de dados, além de uma imagem que se selecionada levará o usuário ao local da *geofence* criada, para excluir a *geofence* é necessário deslizar a mesma para o lado onde aparecerá um fundo vermelho com uma imagem de lixeira que ao ser clicada excluirá a *geofence* tanto do mapa quanto do banco de dados, as imagens ilustrando esta explicação podem ser vistas nas figuras 24 e 25.

Figura 24 – Tela do histórico de *geofences*



Fonte: Elaborado pelo Autor

Figura 25 – Tela do histórico de *geofences* com procedimento de exclusão da *geofence*

Fonte: Elaborado pelo Autor

4. DESENVOLVIMENTO ANDROID

Para desenvolver este aplicativo foram estudadas as tecnologias disponíveis no mercado e selecionadas as que mais se enquadraram nas necessidades deste projeto neste capítulo serão apresentadas tais tecnologias.

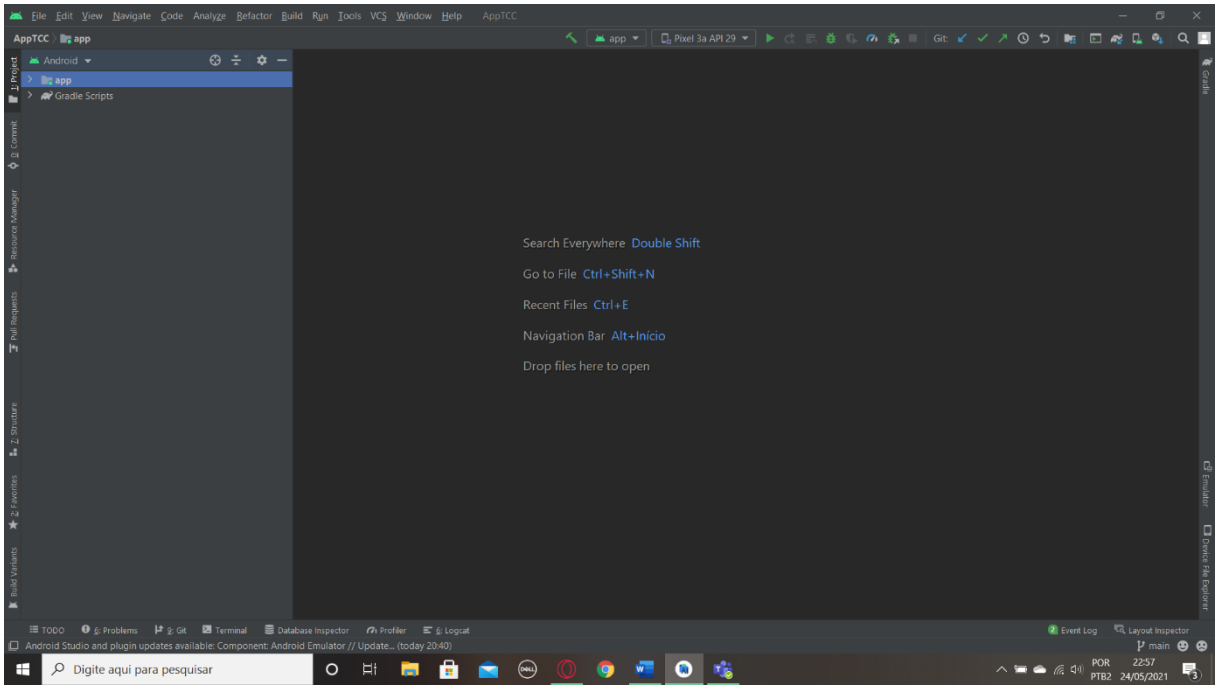
4.1 IDE

Para o desenvolvimento do aplicativo foi utilizada a IDE Android Studio pois é a IDE oficial recomendada pela Google desenvolvedora da plataforma Android e do Android SDK Tools. O desenvolvimento foi realizado em um desktop com o sistema operacional Windows 10.

Esta IDE possui um sistema de simulador de smartphone onde é possível testar o aplicativo em desenvolvimento, além de ser a plataforma de lançamento do aplicativo final até para outras tecnologias como as que desenvolvem aplicativos para Web, Android e IOS.

A figura 26 mostra a tela do Android Studio IDE que foi utilizada no desenvolvimento do aplicativo.

Figura 26 – Tela do Android Studio



Fonte: Elaborado pelo Autor

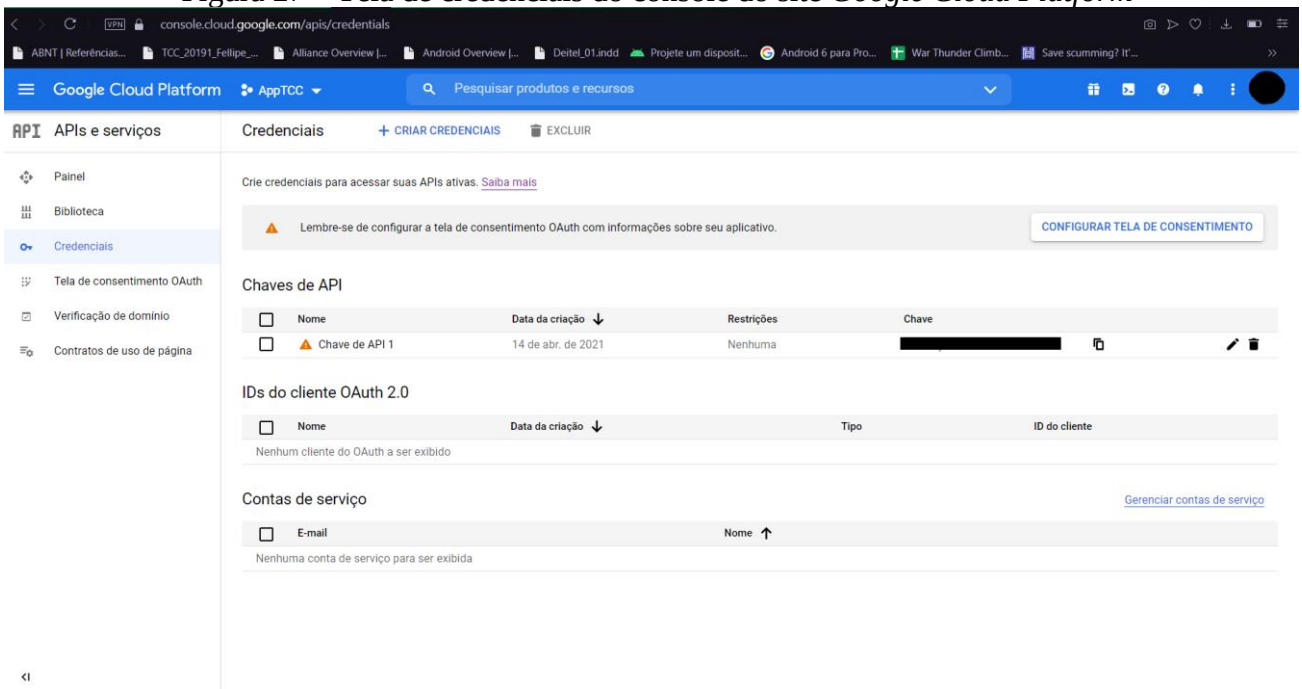
4.2 API's

Algumas API's foram utilizadas nesse projeto e serão descritas neste tópico.

Foi utilizada a API da Google *Biometric*, esta é utilizada na validação da identidade do usuário pois é compatível com qualquer aparelho Android que tenha um sensor de impressão digital. (GOOGLE, 2020)

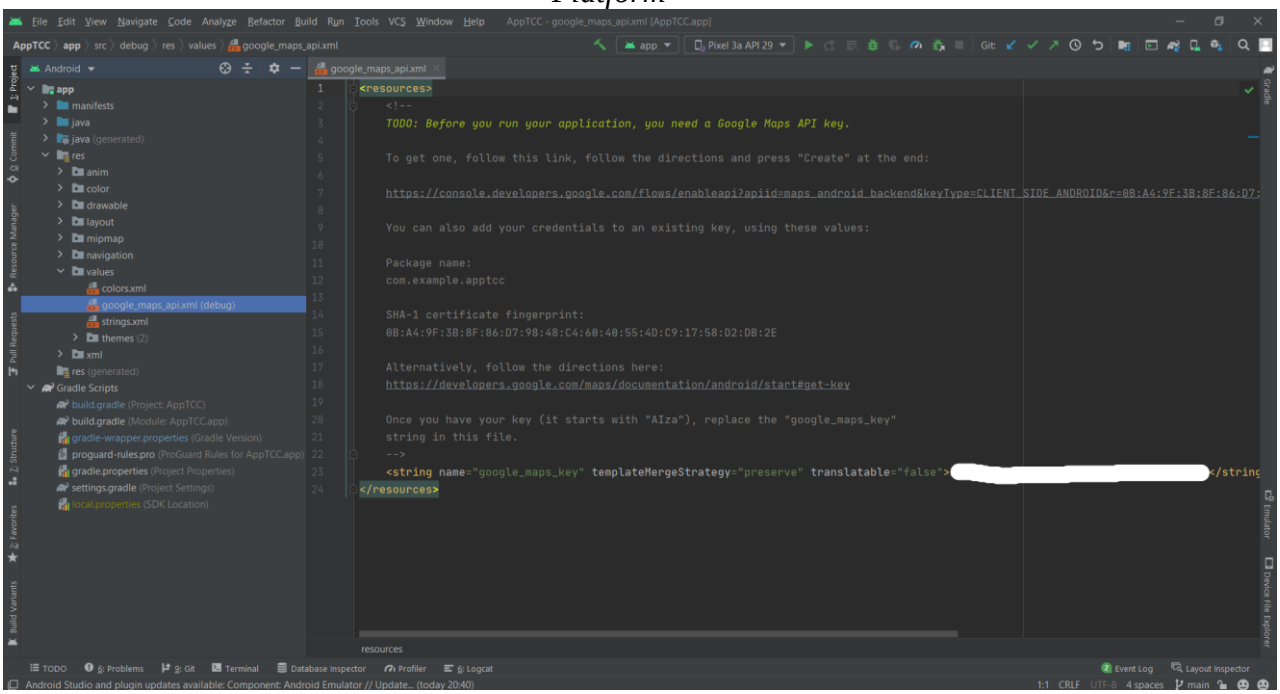
Foi utilizada a API Maps da Google pois essa é nativa e vem de fábrica com qualquer aparelho Android, para utilizar esta API é necessário gerar uma chave única de utilização no site "console.cloud.google.com" como é mostrado na figura 27, esta chave é única para o projeto e qualquer outra API do google a ser utilizada no projeto basta ser ativada no site, porém no Android Studio quando é criada uma ativ ou uma frag do MAPA é gerado um arquivo com um link que facilita a geração desta chave, este arquivo é mostrado na figura 28.

Figura 27 – Tela de credenciais do console do site *Google Cloud Platform*



Fonte: Elaborado pelo Autor

Figura 28 –Arquivo de criação de credenciais do console do site *Google Cloud Platform*



Fonte: Elaborado pelo Autor

Foi utilizada a API da Google *Places* para a detecção do país em que o usuário está no momento, e para sugerir cidades na pesquisa por cidades do aplicativo, esta API também necessita de ser ativada no site “console.cloud.google.com”.

4.3 Abordagens de desenvolvimento utilizadas

No presente trabalho foram utilizadas bibliotecas e abordagens para melhorar o desenvolvimento e são apresentadas na sequência.

Foi utilizada uma biblioteca chamada *easypermissions* com o intuito de facilitar a implementação da requisição de permissões dentro da plataforma Android, esta biblioteca possui um conjunto de instruções que permitem a criação da requisição em duas funções e evita a escrita de muitas funções.

Foi utilizada para a movimentação entre telas dentro do aplicativo a criação de um arquivo de movimentação “nav_graph.xml” que permite criar um grafo de movimentação entre os frags.

Foi utilizado para desenvolver o banco dados que armazena as *geofences* o *Jetpack DataStore*, que é uma solução de armazenamento de dados utilizando de buffers de protocolo

o qual armazena pares de chave-valor ou objetos tipados. (ANDROID OPEN SOURCE PROJECT, 2021)

4.4 Sistema de arquivos do projeto

O arquivo *MyApplication* apresenta em seu código apenas o construtor da aplicação. Os outros arquivos do projeto foram separados em pastas para uma melhor organização deste.

- Pasta *adapters* contém os arquivos:
    - *GeofenceAdapter*: Contém as funções que permitem a interação do usuário com cada *geofence* criada e exibida na tela do histórico de *geofences*.
    - *PredictionsAdapter*: Contém as funções que permitem o funcionamento da pesquisa por meio da Places API na tela dois de criação da *geofence*.
- Pasta *bindingadapters* contém os arquivos:
    - *GeofenceBindings.kt*: Vincula os dados a serem apresentados na tela do histórico de *geofence*.
    - *PredictionsBindings.kt*: Vincula os dados a serem apresentados no autocompletar da pesquisa na tela dois de criação da *geofence*.
    - *Step1Bindings.kt*: Vincula o nome digitado à *geofence*, habilita o botão para a próxima tela se houver algo digitado e torna a barra de progresso visível.
    - *Step2Bindings.kt*: Verifica a conexão com a internet e manda mensagem caso está desligada.
    - *Step3Bindings.kt*: Torna funcional a barra de rolagem e altera de metros para quilômetros a unidade de medida exibida conforme a barra é rolada
- Pasta *broadcastreciver* contém o arquivo:
    - *GeofenceBroadcastReceiver*: monitora a posição do aparelho e notifica se este entrou, se mantém, ou saiu da área da *geofence*,
- Pasta *data* contém os arquivos:
    - *Converters*: Converte o formato Bitmap da imagem da *geofence* que é exibida na tela do histórico de *geofences* para um *ByteArray* para ser salvo no banco de dados.
    - *DataStoreRepository.kt*: armazena e acessa os dados usando chaves.
    - *GeofenceDao*: Contém a função de leitura do banco de dados e pausa as funções de escrita e remoção do banco de dados.

- *GeofenceDatabase*: Conecta a classe *GeofenceEntity* e a interface *GeofenceDao* ao banco de dados.
- *GeofenceEntity*: Contém a classe *GeofenceEntity* e seus atributos.
- *GeofenceRepository*: Faz a leitura do banco de dados e pausa as funções de escrita e remoção do banco de dados.

- Pasta *di* contém o arquivo:
  - *DatabaseModule*: Contém o objeto *DatabaseModule* que contém a função de criação do banco de dados.
- Pasta *ui* contém as patas e o arquivo *MainActivity*:
  - *addgeofence*: Contém os arquivos:
    - *Step1Fragment*: Contém as funções de conexão do frag com a ativ, atualiza o nome da *geofence* no banco de dados, inicialização da Places API e localização do país onde o aparelho está localizado.
    - *Step2Fragment*: Contém as funções de conexão do frag com a ativ, inicialização da *Places* API, recebe a informação do país coletadas anteriormente, para gerar as opções de cidades que aparecem na pesquisa de cidades, insere a cidade selecionada no banco de dados, e verificar se existe a conexão com a internet.
    - *Step3Fragment*: Contém as funções de conexão do frag com a ativ, atualiza o valor do raio da *geofence* no banco de dados, e informa que a *geofence* está pronta.
  - *fingerprint*: Contém o arquivo:
    - *FingerprintFragment*: Contém as funções de conexão do frag com a ativ e inicialização e execução da API *Biometric*.
  - *geofences*: Contém o arquivo:
    - *GeofenceFragment*: Contém as funções de conexão do frag com a ativ, conexão com *geofencesAdapter* e leitura do banco de dados.
  - *maps*: Contém o arquivo:
    - *MapsFragment*: Contém as funções de conexão do frag com a ativ, verificação da *geofence*, aproximação de uma área determinada do mapa, leitura do banco de dados, verificação de longo clique no mapa, requisição de permissão de localização o tempo todo, criação da *geofence* no banco e dados se a localização estiver ativa.
  - *permission*: Contém o arquivo:

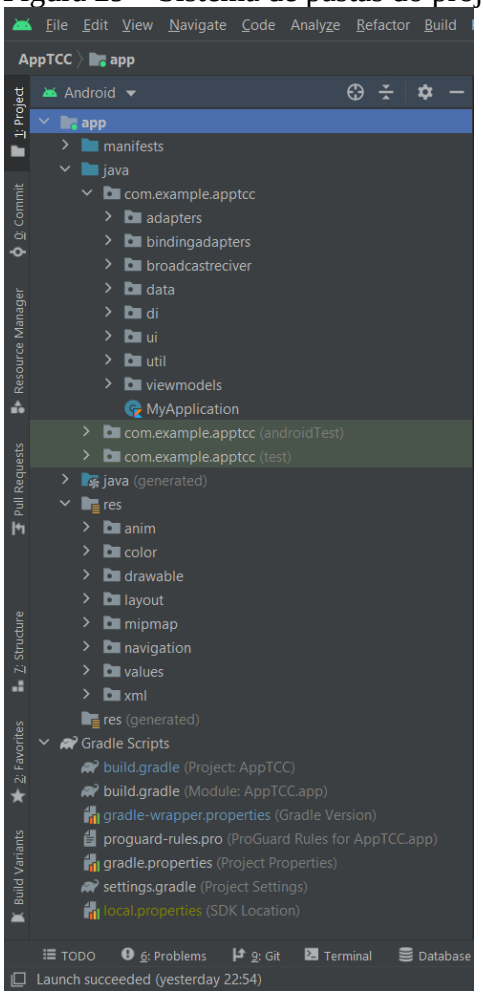
        - *PermissionFragment*: Contém as funções de conexão do frag com a ativ, verificação se a permissão de acesso a localização foi concedida ou negada e verificação se é a primeira vez que o aplicativo é aberto.
    - *MainActivity*: Verifica se a permissão de localização foi concedida se foi passa para a tela de autenticação do usuário.
- Pasta *util* contém os arquivos:
    - *Constants*: Contém todas as constantes usadas no projeto.
    - *ExtensionFunctions*: Contém as funções que habilitam e desabilitam a visualização, que mostram ou dispensam, e de observação do ciclo de vida.
    - *MyDiffUtil*: Contém funções que recebem o tamanho de duas listas e realocam os itens se estes e seus conteúdos forem os mesmos.
    - *NetworkListener*: Contém funções que verificam a conexão com a internet, e que mudam o valor da variável *isNetworkAvailable* entre verdadeiro ou falso caso internet esteja conectado ou desconectada.
    - *Permissions*: Contém as funções padrões da biblioteca *easypermissions*, que verificam se a permissão está concedida ou requisitam a permissão.
- Pasta *viewmodels* contém os arquivos:
    - *SharedViewModel*: Inicializa as variáveis do banco de dados com valores iniciais, e contém funções que reseta essas variáveis, salva a primeira vez que o aplicativo é aberto, adiciona a *geofence* ao banco de dados, remove a *geofence* do banco de dados, utiliza a verificação de posição do usuário, inicializa a *geofence*, para a *geofence* para removê-la, calcula a aproximação da câmera quando foca em uma *geofence*, verifica as configurações de localização do dispositivo.
    - *Step1ViewModel*: Contém a função que habilita o botão de passar para a próxima tela.
    - *Step2ViewModel*: Contém as funções que habilita o botão de passar para a próxima tela e de informar o valor online para a variável *_internetAvailable*.
- *MyApplication* é o arquivo que contém o construtor da aplicação como já foi mencionado anteriormente.

Dentro da pasta “res” toda a parte gráfica do aplicativo dividida em pastas para organizar o projeto.

- A pasta *anim* contém todas as animações de tela utilizadas na aplicação.
- A pasta *color* contém as cores do plano de fundo do aplicativo.
- A pasta *drawable* contém as imagens utilizadas no projeto.
- A pasta *layout* contém os arquivos gráficos das telas do aplicativo.
- A pasta *mipmap* contém a imagem do atalho do aplicativo que vai aparecer o dispositivo do usuário.
- A pasta *navigation* contém o grafo de navegação entre telas do aplicativo.
- A pasta *values* contém variáveis utilizadas na constrição da interface como cores e *strings*.
- A pasta *xml* contém um arquivo de movimentação na tela do histórico de *geofences*.

Os códigos do projeto estão nos apêndices de A até AI, na ordem em que foram implementados, para um melhor entendimento das informações passadas anteriormente, a figura 29 permite visualizar esse sistema de pastas.

Figura 29 – Sistema de pastas do projeto



Fonte: Elaborado pelo Autor

5. CONSIDERAÇÕES FINAIS E PERSPECTIVAS FUTURAS

Esse trabalho objetivou a criação de um aplicativo de criação e manutenção de *geofences* e integrar a este aplicativo uma validação do usuário por meio de sua impressão digital.

O Android Studio tornou possível o desenvolvimento do aplicativo por conter as bibliotecas que tornam possível desenvolver uma gama de aplicativos.

Foram feitos testes para verificar o funcionamento do aplicativo e a possibilidade de integração da validação do usuário e do sistema de *geofence* e obteve-se um resultado satisfatório respondendo as questões de pesquisa, mostrando ser possível a integração dos sistemas.

O projeto proporcionou um conhecimento em diversas áreas como desenvolvimento para Android com Kotlin, utilização da API's da Google *Places*, *Biometric*, *Maps*, a criação de um banco de dados na memória interna do aparelho, e as diversas maneiras de se validar uma impressão digital.

O projeto apresentou um grande potencial de desenvolvimento de aplicativos que necessitem de validação do usuário por meio de impressão digital e que utilizem a tecnologia *geofence*.

O projeto se viabilizou fisicamente e até agora se mostrou uma ferramenta de uso importante no ambiente Android.

Para projetos futuros sugere-se implementar um banco de dados que possa ser compartilhado com outros aparelhos, um sistema de criptografia ponta a ponta na comunicação do banco de dados com o aparelho, monitorar mais de um aparelho se movimentando no mapa, entre outras possíveis aplicações.

REFERÊNCIAS

ALVES, Paulo.**VERIFICAÇÃO AUTOMÁTICA GEORREFERENCIADA**. Orientador: Prof. M.E.E. Marcelo Antônio Adad de Araújo. 2018. 77 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação) - Escola de Ciências Exatas e da Computação, Pontifícia Universidade Católica de Goiás., Goiânia, 2018.

BHATNAGAR, Mayank. **Magic lies here - Statically vs Dynamically Typed Languages**. [*S. l.*], 9 set. 2018. Disponível em: https://android.jlelse.eu/magic-lies-here-statically-typed-vs-dynamically-typed-languages-d151c7f95e2b. Acesso em: 13 nov. 2020.

CARVALHO, Edilson; ARAÚJO, Paulo. **Noções básicas de sistema de posicionamento global GPS**. [*S. l.*:*s. n.*], 2009. 28 p. ISBN 9788572735254. Disponível em: http://www.ead.uepb.edu.br/arquivos/cursos/Geografia_PAR_UAB/Fasciculos%20-%20Material/Leituras_Cartograficas_II/Le_Ca_II_A08_MZ_GR_260809.pdf. Acesso em: 12 nov. 2020.

CASTRO, Thiago. **Identificação de Impressões Digitais Baseada na Extração de Minúcias**. Orientador: Prof. Augusto Santiago Cerqueira e Prof. David Sérgio Adães Gouvêa. 2008. 118 p. Dissertação (Mestrado em Engenharia elétrica) - Faculdade de Engenharia, Universidade Federal de Juiz de Fora, Juiz de Fora, MG-Brasil, 2008. Disponível em: http://repositorio.ufjf.br:8080/jspui/bitstream/ufjf/3518/1/thiagodasilvacastro.pdf. Acesso em: 10 nov. 2020.

DEITEL, Paul; DEITEL, Harvey; WALD, Alexander. **Android 6 para Programadores**: Uma Abordagem Baseada em Aplicativos. 3. ed. aum. [*S. l.*]: Bookman Editora, 2016. 456 p. ISBN 8582604122, 9788582604120.*E-book*.

DSS SWFT FINGERPRINTING PROGRAM. **History of Fingerprinting**. [*S. l.*], 2009. Disponível em: https://www.fieldprintswft.com/SubPage_FullWidth.aspx?ChannelID=309. Acesso em: 12 nov. 2020.

GALTON, Francis. **Finger Prints**. London: Macmillan and CO., 1892. 247 p. Disponível em: http://www.biometricbits.com/Galton-Fingerprints-1892.pdf. Acesso em: 14 nov. 2020.

ANDROID OPEN SOURCE PROJECT. **Arquitetura Android**. [*S. l.*], 1 set. 2020. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://source.android.com/devices/architecture. Acesso em: 26 out. 2020.

ANDROID OPEN SOURCE PROJECT. **Criar e monitorar fronteiras geográficas virtuais**. [*S. l.*], 25 set. 2020. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://developer.android.com/training/location/geofencing. Acesso em: 11 out. 2020.

ANDROID OPEN SOURCE PROJECT. **Activity**. [*S. l.*], 18 maio 2021. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://developer.android.com/reference/android/app/Activity?hl=pt-br. Acesso em: 20 maio 2021.

ANDROID OPEN SOURCE PROJECT. **Guia para a arquitetura do app**. [S. l.], 6 maio 2021. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://developer.android.com/jetpack/guide?hl=pt-br. Acesso em: 20 maio 2021.

ANDROID OPEN SOURCE PROJECT. **Mostrar caixa de diálogo de autenticação biométrica**. [*S. l.*], 25 set. 2020. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://developer.android.com/training/sign-in/biometric-auth?hl=pt-br. Acesso em: 11 out. 2020.

ANDROID OPEN SOURCE PROJECT. **DataStore**. [*S. l.*], 3 maio 2021. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://developer.android.com/topic/libraries/architecture/datastore?hl=pt-br. Acesso em: 22 maio 2021.

ANDROID OPEN SOURCE PROJECT. **Fragmentos**. [*S. l.*], 27 dez. 2019. used according to terms described in the Creative Commons 2.5 Attribution License. Disponível em: https://developer.android.com/guide/components/fragments?hl=pt-br. Acesso em: 20 maio 2021.

ITIMU, Kiruti. **Renaissance on Flagship Phones**. [*S. l.*], 10 maio 2019. Disponível em: https://techweez.com/2019/05/10/capacitive-fingerprint-scanners-need-to-come-back/#comment-15325. Acesso em: 12 nov. 2020.

LEÃO, Márcia. **Instrumentação para Ensino**: Reflexão Interna Total Frustrada ou Penetração de Barreira Óptica. [*S. l.*], 2014. Disponível em: https://sites.ifi.unicamp.br/lunazzi/files/2014/03/009289Marcia_Orlando_F809_RF.pdf. Acesso em: 12 nov. 2020.

LU, Yipeng. **Piezoelectric Micromachined Ultrasonic Transducers for Fingerprint Sensing**. 2015. 162 p. Tese para Ph.D. (Doutorado em filosofia em mecânica e engenharia aeroespacial) - Escritório de estudos de graduação, Universidade da Califórnia, [*S. l.*], 2015. Disponível em: https://www.researchgate.net/publication/285770473_Piezoelectric_Micromachined_Ultrasonic_Transducers_for_Fingerprint_Sensing. Acesso em: 12 nov. 2020.

MÁRCICO, José. **Papiloscopia**. [*S. l.*], 2002. Disponível em: http://www.papiloscopia.com.br. Acesso em: 11 nov. 2020.

MISHRA, Annapurna; DEHURI, Satchidananda. Real-time online fingerprint image classification using adaptive hybrid techniques. **International Journal of Electrical and Computer Engineering (IJECE)**, [*S. l.*], v. 9, n. 5, p. 4372-4381, out. 2019. DOI 10.11591/ijece.v9i5.pp4372-4381. Disponível em: https://www.researchgate.net/publication/336886519_Real-time_online_fingerprint_image_classification_using_adaptive_hybrid_techniques. Acesso em: 12 nov. 2020.

MOSKALA, Marcin; WOJDA, Igor. **Android Development with Kotlin**: Enhance your skills for Android development using Kotlin. 1. ed. Livery Place, 35 Livery Street,

Birmingham B3 2PB, UK: Packt Publishing, 2017. 440 p. ISBN 1787128989, 9781787128989.*E-book*.

MURTHY, Anitaa. **5 steps to implement Biometric authentication in Android**. [*S. l.*], 11 jul. 2018. Disponível em: https://proandroiddev.com/5-steps-to-implement-biometric-authentication-in-android-dbeb825aeee8. Acesso em: 12 nov. 2020.

OLIVEIRA, Gabriel. **UM ESTUDO SOBRE IMPRESSÃO DIGITAL**. Orientador: Prof. Wagner Felippe Pacheco. 2016. 56 p. Trabalho de Conclusão de Curso (Graduação em Química Industrial) - Instituto de Química, Universidade Federal Fluminense, Niterói, RJ, 2016. Disponível em: https://app.uff.br/riuff/bitstream/1/6474/1/Monografia%20-%20Gabriel%20Duarte%20de%20Oliveira.pdf. Acesso em: 10 nov. 2020.

RED HAT, INC. **INTERFACE DE PROGRAMAÇÃO DE APLICAÇÕES**: O que é API?. [*S. l.*], 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 6 nov. 2020.

SILVA, Fellipe. **Sistema de identificação biométrica baseado em extração de minúcias e redes neurais artificiais**. Orientador: Prof. Dra. Janaína Gonçalves Guimarães. 2019. 66 p. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) - Departamento de Engenharia de Controle e Automação e Computação, Universidade Federal de Santa Catarina, Blumenau, 2019. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/197848/TCC_20191_Fellipe_Silva.pdf?sequence=1&isAllowed=y. Acesso em: 10 nov. 2020.

SOARES, Alexandre; BOTELHO, Pedro; AIEX, Nicolau. **Biometria**. [*S. l.*], 6 mar. 2009. Disponível em: https://www.gta.ufrj.br/grad/09_1/versao-final/biometria/index.html. Acesso em: 12 nov. 2020.

VAATI, Esther.**What Is the Android SDK and How to Start Using It**. [*S. l.*], 2 jul. 2020. Disponível em: https://code.tutsplus.com/tutorials/the-android-sdk-tutorial--cms-34623. Acesso em: 13 nov. 2020.

WINDER, Davey. **Samsung Galaxy S10 Fingerprint Scanner Hacked - Here‘s What You Need To Know**. [*S. l.*], 6 abr. 2019. Disponível em: https://www.forbes.com/sites/daveywinder/2019/04/06/samsung-galaxy-s10-fingerprint-scanner-hacked-heres-what-you-need-to-know/?sh=14074865d423. Acesso em: 12 nov. 2020.

WOODFORD, Chris. **Biometric fingerprint scanners**. [*S. l.*], 6 nov. 2020. Disponível em: https://www.explainthatstuff.com/fingerprintscanners.html#:~:text=The%20scanner%20uses%20a%20light,turn%20it%20into%20a%20code. Acesso em: 12 nov. 2020.

APÊNDICE A - Código build.gradle(Project: AppTCC)

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
   repositories {
      google()
      jcenter()
   }
   dependencies {
      classpath "com.android.tools.build:gradle:4.1.3"
      classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.4.31"
      classpath "com.google.dagger:hilt-android-gradle-plugin:2.31.2-alpha"
      classpath "androidx.navigation:navigation-safe-args-gradle-plugin:2.3.4"

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

allprojects {
   repositories {
      google()
      jcenter()
   }
}

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

APÊNDICE B - Código build.gradle(Module: AppTCC.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 30
   buildToolsVersion "30.0.3"

   defaultConfig {
      applicationId "com.example.apptcc"
      minSdkVersion 21
      targetSdkVersion 30
      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_1_8
      targetCompatibility JavaVersion.VERSION_1_8
   }
   kotlinOptions {
      jvmTarget = '1.8'
   }
}

dependencies {

   implementation "org.jetbrains.kotlin:kotlin-stdlib:1.4.31"
   implementation 'androidx.core:core-ktx:1.3.2'
   implementation 'androidx.appcompat:appcompat:1.2.0'
```

```
    implementation 'com.google.android.material:material:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.navigation:navigation-fragment-ktx:2.3.4'
    implementation 'androidx.navigation:navigation-ui-ktx:2.3.4'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    // Lifecycle
    implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
    implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.3.1"

    // Coroutines
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.4.2'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.4.2'

    // Coil - Image Loading Library
    implementation("io.coil-kt:coil:1.1.1")

    // Places SDK
    implementation 'com.google.android.libraries.places:places:2.4.0'

    // Easy Permissions
    implementation 'com.vmadalin:easypermissions-ktx:0.1.0'

    // Room components
    implementation "androidx.room:room-runtime:2.2.6"
    kapt "androidx.room:room-compiler:2.2.6"
    implementation "androidx.room:room-ktx:2.2.6"
    androidTestImplementation "androidx.room:room-testing:2.2.6"

    // Dagger - Hilt
    implementation "com.google.dagger:hilt-android:2.31.2-alpha"
    implementation "androidx.hilt:hilt-lifecycle-viewmodel:1.0.0-alpha03"
    kapt "com.google.dagger:hilt-android-compiler:2.31.2-alpha"
    kapt "androidx.hilt:hilt-compiler:1.0.0-beta01"

    // DataStore
    implementation "androidx.datastore:datastore-preferences:1.0.0-alpha08"

    // Util
    implementation 'com.google.maps.android:android-maps-utils:2.2.0'

    //Biometric
    implementation "androidx.biometric:biometric:1.1.0"
}
```

APÊNDICE C - Código GeofencesAdapter

```
package com.example.apptcc.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.apptcc.R
import com.example.apptcc.data.GeofenceEntity
import com.example.apptcc.databinding.GeofencesRowLayoutBinding
import com.example.apptcc.ui.geofences.GeofencesFragmentDirections
import com.example.apptcc.util.MyDiffUtil
import com.example.apptcc.viewmodels.SharedViewModel
import com.google.android.material.snackbar.Snackbar
import kotlinx.coroutines.launch

//Está classe e suas funções tem por objetivo gerenciar as geofences que aparecem na tela do
historico de geofences
class GeofencesAdapter(private val sharedViewModel: SharedViewModel) :
   RecyclerView.Adapter<GeofencesAdapter.MyViewHolder>() {

   private var geofenceEntity = mutableListOf<GeofenceEntity>()

   //Recebe as opções a serem exibidas na tela e exibe elas
   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)
         }
      }
   }

   //Cria o suporte de visualização
   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
      return MyViewHolder.from(parent)
   }

   //Passa as opções a serem exibidas como parametro para a função bind
   override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
      val currentGeofence = geofenceEntity[position]
```

```
        holder.bind(currentGeofence)

        //Se clicar na imagem da lixeira chama a função que remove a geofence
        holder.binding.deleteImageView.setOnClickListener {
            removeItem(holder, position)
        }

        //Se clicar na imagem da geofence vai para a tela do mapa e centraliza na geofence
        holder.binding.snapshotImageView.setOnClickListener {
            val action =

GeofencesFragmentDirections.actionGeofencesFragmentToMapsFragment(currentGeofence)
            holder.itemView.findNavController().navigate(action)
        }
    }

    //Remove a geofence tanto da tela quanto do banco de dados
    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 NOT REMOVED!")
            }
        }
    }

    //Informa que a geofence foi removida e da a opção de cancelar a ação
    private fun showSnackBar(
        holder: MyViewHolder,
        removedItem: GeofenceEntity
    ) {
        Snackbar.make(
            holder.itemView,
            "Removida " + removedItem.name,
            Snackbar.LENGTH_LONG
        ).setAction("UNDO") {
            undoRemoval(holder, removedItem)
        }.show()
    }

    //Cancela 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 a tamanho do dados da geofence armazenados
    override fun getItemCount(): Int {
        return geofenceEntity.size
    }

    //Atualiza a lista de geofences exibidas na tela
    fun setData(newGeofenceEntity: MutableList<GeofenceEntity>) {
        val geofenceDiffUtil = MyDiffUtil(geofenceEntity, newGeofenceEntity)
        val diffUtilResult = DiffUtil.calculateDiff(geofenceDiffUtil)
        geofenceEntity = newGeofenceEntity
        diffUtilResult.dispatchUpdatesTo(this)
    }
}
```

APÊNDICE D - Código PredictionsAdapter

```
package com.example.apptcc.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.apptcc.R
import com.example.apptcc.data.GeofenceEntity
import com.example.apptcc.databinding.GeofencesRowLayoutBinding
import com.example.apptcc.ui.geofences.GeofencesFragmentDirections
import com.example.apptcc.util.MyDiffUtil
import com.example.apptcc.viewmodels.SharedViewModel
import com.google.android.material.snackbar.Snackbar
import kotlinx.coroutines.launch

//Está classe e suas funções tem por objetivo gerenciar as geofences que aparecem na tela do
historico de geofences
class GeofencesAdapter(private val sharedViewModel: SharedViewModel) :
   RecyclerView.Adapter<GeofencesAdapter.MyViewHolder>() {

   private var geofenceEntity = mutableListOf<GeofenceEntity>()

   //Recebe as opções a serem exibidas na tela e exibe elas
   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)
         }
      }
   }

   //Cria o suporte de visualização
   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
      return MyViewHolder.from(parent)
   }

   //Passa as opções a serem exibidas como parametro para a função bind
   override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
      val currentGeofence = geofenceEntity[position]
```

```
        holder.bind(currentGeofence)

        //Se clicar na imagem da lixeira chama a função que remove a geofence
        holder.binding.deleteImageView.setOnClickListener {
            removeItem(holder, position)
        }

        //Se clicar na imagem da geofence vai para a tela do mapa e centraliza na geofence
        holder.binding.snapshotImageView.setOnClickListener {
            val action =

GeofencesFragmentDirections.actionGeofencesFragmentToMapsFragment(currentGeofence)
            holder.itemView.findNavController().navigate(action)
        }
    }

    //Remove a geofence tanto da tela quanto do banco de dados
    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 NOT REMOVED!")
            }
        }
    }

    //Informa que a geofence foi removida e da a opção de cancelar a ação
    private fun showSnackBar(
        holder: MyViewHolder,
        removedItem: GeofenceEntity
    ) {
        Snackbar.make(
            holder.itemView,
            "Removida " + removedItem.name,
            Snackbar.LENGTH_LONG
        ).setAction("UNDO") {
            undoRemoval(holder, removedItem)
        }.show()
    }

    //Cancela 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 a tamanho do dados da geofence armazenados
    override fun getItemCount(): Int {
        return geofenceEntity.size
    }

    //Atualiza a lista de geofences exibidas na tela
    fun setData(newGeofenceEntity: MutableList<GeofenceEntity>) {
        val geofenceDiffUtil = MyDiffUtil(geofenceEntity, newGeofenceEntity)
        val diffUtilResult = DiffUtil.calculateDiff(geofenceDiffUtil)
        geofenceEntity = newGeofenceEntity
        diffUtilResult.dispatchUpdatesTo(this)
    }
}
```

APÊNDICE E - Código GeofencesBindings.kt

```
package com.example.apptcc.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.apptcc.R
import com.example.apptcc.data.GeofenceEntity
import com.example.apptcc.util.ExtensionFunctions.disable
import com.example.apptcc.util.ExtensionFunctions.enable

/*
 Se não houver geofences na tela do historico de
 geofence aparece uma imagem e uma menssagem informando
 que não foi encontrada nenhuma geofence, caso contrario elas ficam invisiveis.
 */
@BindingAdapter("setVisibility")
fun View.setVisibility(data: List<GeofenceEntity>) {
   if (data.isNullOrEmpty()) {
      this.visibility = View.VISIBLE
   } else {
      this.visibility = View.INVISIBLE
   }
}
//Cria a movimentação da geofence para o o lado esquerdo para mostrar a opção de deletar a
geofence.
@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()
         }
      }
   })
}

//Carrega a imagem da geofence que aparece na tela do historico de geofences.
@BindingAdapter("loadImage")
fun ImageView.loadImage(bitmap: Bitmap) {
   this.load(bitmap)
```

```
}

//Limita a quantidade de casas que aparecem no numero da latitude e longitude.
@BindingAdapter("parseCoordinates")
fun TextView.parseCoordinates(value: Double) {
   val coordinate = String.format("%.4f", value)
   this.text = coordinate
}
```

APÊNDICE F - Código PredictionsBindings.kt

```
package com.example.apptcc.bindingadapters

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

//Passa o tipo do atributo cidade recebido para string.
@BindingAdapter("setCity")
fun TextView.setCity(prediction: AutocompletePrediction){
   this.text = prediction.getPrimaryText(null).toString()
}

//Passa o tipo do atributo país recebido para string.
@BindingAdapter("setCountry")
fun TextView.setCountry(prediction: AutocompletePrediction){
   this.text = prediction.getSecondaryText(null).toString()
}
```

APÊNDICE G - Código Step1Bindings.kt

```
package com.example.apptcc.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.apptcc.R
import com.example.apptcc.viewmodels.SharedViewModel
import com.example.apptcc.viewmodels.Step1ViewModel
import com.google.android.material.textfield.TextInputEditText

/*
   Salva o nome da geofence e se o nome não estiver vasio habilita
   o botão para passar para proxima tela.
*/
@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)
   }
}

//Se o botão para passar para proxima tela estiver habilitado e for clicado passa para a tela step2.
@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)
      }
   }
}

/*
   Se o botão para passar para proxima tela estiver habilitado desaparece a barra de carregamento
```

```
   caso contrario ela fica visivel.
*/
@BindingAdapter("setProgressVisibility")
fun ProgressBar.setProgressVisibility(nextButtonEnabled: Boolean){
   if(nextButtonEnabled){
      this.visibility = View.GONE
   }else {
      this.visibility = View.VISIBLE
   }
}
```

APÊNDICE H - Código Step2Bindings.kt

```
package com.example.apptcc.bindingadapters

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

/*
   Informa se não há conexão com a internet e desabilita o apêndice da pesquisa
   caso haja conexão com a internet habilita o apêndice.
*/
@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 I - Código Step3Bindings.kt

```
package com.example.apptcc.bindingadapters

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

/*
   Habilata o movimento da barra para seleção do raio da geofence, salva o raio,
   e chama a função de atualização da unidade de maedida aprensentada na tela.
 */
@BindingAdapter("updateSliderValueTextView", "getGeoRadius", requireAll = true)
fun Slider.updateSliderValue(textView: TextView, sharedViewModel: SharedViewModel){
   this.updateSliderValueTextView(sharedViewModel.geoRadius, textView)
   this.addOnChangeListener { _, value, _ ->
      sharedViewModel.geoRadius = value
      updateSliderValueTextView(sharedViewModel.geoRadius, textView)
   }
}

//Atualiza da unidade de medida aprensentada na tela.
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 J - Código GeofenceBroadcastReceiver

```
package com.example.apptcc.broadcastreciver

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.apptcc.R
import com.example.apptcc.util.Constants.NOTIFICATION_CHANNEL_ID
import com.example.apptcc.util.Constants.NOTIFICATION_CHANNEL_NAME
import com.example.apptcc.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

class GeofenceBroadcastReceiver: BroadcastReceiver() {
  /*
    Verifica a posição do aparelho e chama a função de notificação caso ele
    entre, saia, se mantenha, ou ocorra um erro na identificação da movimentação do usuario.
   */
  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
    }

    when(geofencingEvent.geofenceTransition){
      Geofence.GEOFENCE_TRANSITION_ENTER -> {
        Log.d("BroadcastReceiver", "Entrou na Geofence")
        displayNotification(context, "Entrou na Geofence")
      }

      Geofence.GEOFENCE_TRANSITION_EXIT -> {
        Log.d("BroadcastReceiver", "Saiu da Geofence")
        displayNotification(context, "Saiu na Geofence")
      }

      Geofence.GEOFENCE_TRANSITION_DWELL -> {
        Log.d("BroadcastReceiver", "Permanece na Geofence")
        displayNotification(context, "Permanece na Geofence")
      }
      else -> {
        Log.d("BroadcastReceiver", "Tipo Invalido")
        displayNotification(context, "Tipo Invalido")
```

```
            }
        }
    }

    /*
        Função que chama a função de criação do canal de notificação
        e mostra uma notificação no aparelho do usuario.
    */
    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())
    }

    //Ciar o canal de notificações.
    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 K - Código Converters

```
package com.example.apptcc.data

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

class Converters {

   //Converte o tipo de Bitmap para ByteArray para salvar a imagem no banco de dados.
   @TypeConverter
   fun fromBitmap(bitmap: Bitmap): ByteArray{
      val outputStream = ByteArrayOutputStream()
      bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream)
      return outputStream.toByteArray()
   }

   /*
      Converte o tipo de ByteArray da Bitmap para exibir a
      imagem na tela do historico de geofences.
   */
   @TypeConverter
   fun toBitmap(byteArray: ByteArray): Bitmap {
      return BitmapFactory.decodeByteArray(byteArray, 0, byteArray.size)
   }
}
```

APÊNDICE L - Código DataStoreRepository.kt

```
package com.example.apptcc.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.apptcc.util.Constants.PREFERENCE_FIRST_LAUNCH
import com.example.apptcc.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

private val Context.dataStore by preferencesDataStore(PREFERENCE_NAME)

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

   //A variável first launch recebe a chave PreferenceKey passando por parâmetro o nome
“firstlaunch”.
   private object PreferenceKey {
      val firstLaunch = booleanPreferencesKey(PREFERENCE_FIRST_LAUNCH)
   }

   //a variável dataStore recebe o ponteiro do banco de dados.
   private val dataStore: DataStore<Preferences> = context.dataStore

   //Pausa a função saveFirstLaunch.
   suspend fun saveFirstLaunch(firstLaunch: Boolean) {
      dataStore.edit { preference ->
         preference[PreferenceKey.firstLaunch] = firstLaunch
      }
   }

   //A variável readFirstLaunch recebe a exceção e o estado de preferência do banco de dados.
   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 M - Código GeofenceDao

```
package com.example.apptcc.data

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

@Dao
interface GeofenceDao {

   //Cria a função de leitura do banco de dados.
   @Query("SELECT * FROM geofence_table ORDER BY id ASC")
   fun readGeofences(): Flow<MutableList<GeofenceEntity>>

   //Pausa a inserção de geofences no banco de dados.
   @Insert(onConflict = OnConflictStrategy.REPLACE)
   suspend fun addGeofence(geofenceEntity: GeofenceEntity)

   //Pausa a remoção de geofences no banco de dados.
   @Delete
   suspend fun  removeGeofence(geofenceEntity: GeofenceEntity)
}
```

APÊNDICE N - Código GeofenceDatabase

```
package com.example.apptcc.data

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

//Marca a classe GeofenceEntity como as entidades do banco de dados.
@Database(
   entities = [GeofenceEntity::class],
   version = 1,
   exportSchema = false
)

/*
   Cria a classe GeofenceDatabase como banco de dados interno e a função
   geofenceDao como uma interface GeofenceDao.
*/
@TypeConverters(Converters::class)
abstract class GeofenceDatabase: RoomDatabase() {

   abstract fun geofenceDao(): GeofenceDao

}
```

APÊNDICE O - Código GeofenceEntity

```
package com.example.apptcc.data

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

//Cria a classe GeofenceEntity e seus parametros que se tornaram as entidades do banco de dados.
@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 P - Código GeofenceRepository

```
package com.example.apptcc.data

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

@ViewModelScoped
class GeofenceRepository @Inject constructor(private  val geofenceDao: GeofenceDao) {

   //Faz a leitura do banco de dados recebendo uma lista.
   val readGeofences: Flow<MutableList<GeofenceEntity>> = geofenceDao.readGeofences()

   //Pausa a inserção de geofences no banco de dados.
   suspend fun addGeofence(geofenceEntity: GeofenceEntity){
      geofenceDao.addGeofence(geofenceEntity)
   }

   //Pausa a remoção de geofences no banco de dados.
   suspend fun removeGeofence(geofenceEntity: GeofenceEntity) {
      geofenceDao.removeGeofence(geofenceEntity)
   }
}
```

APÊNDICE Q – Código DatabaseModule

```
package com.example.apptcc.di

import android.content.Context
import androidx.room.Room
import com.example.apptcc.data.GeofenceDatabase
import com.example.apptcc.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

@Module
@InstallIn(SingletonComponent::class)
object DatabaseModule {

   //Construtor do banco de dados
   @Singleton
   @Provides
   fun provideDatabase(
         @ApplicationContext context: Context
   ) = Room.databaseBuilder(
         context,
         GeofenceDatabase::class.java,
         DATABASE_NAME
   ).build()

   //O banco de dados recebe a classe de interface geofenceDao.
   @Singleton
   @Provides
   fun provideDao(database: GeofenceDatabase) = database.geofenceDao()

}
```

APÊNDICE R - Código Step1Fragment

```
package com.example.apptcc.ui.addgeofence

import android.annotation.SuppressLint
import android.location.Geocoder
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import com.example.apptcc.R
import com.example.apptcc.databinding.FragmentStep1Binding
import com.example.apptcc.viewmodels.SharedViewModel
import com.example.apptcc.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


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

    //Inicializa a Places API, cria o client, e recebe o codigo da região.
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Places.initialize(requireContext(), getString(R.string.google_maps_key))
        placesClient = Places.createClient(requireContext())
        geocoder = Geocoder(requireContext())
    }

    /*
        Anexa este frag a ativ,
        se o botão de voltar for clicado chama a função onStep1BackClicked
        e chama a função getCountryCodeFromCurrentLocation para
```

```
    receber o país onde o usuario está localizado.
*/
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    // Inflate the layout for this fragment
    _binding = FragmentStep1Binding.inflate(layoutInflater, container, false)
    binding.sharedViewModel = sharedViewModel
    binding.step1ViewModel = step1ViewModel
    binding.lifecycleOwner = this

    binding.step1Back.setOnClickListener{
        onStep1BackClicked()
    }

    getCountryCodeFromCurrentLocation()

    return binding.root
}

//Volta a tela do mapa.
private fun onStep1BackClicked() {
    findNavController().navigate(R.id.action_step1Fragment_to_mapsFragment)
}

/*
    Recebe na variavel geoCountryCode o código do país onde o aparelho está localizado
    se for recebido chama a função que habilita o botão de prassar para a proxima tela.
*/
@SuppressLint("MissingPermission")
private fun getCountryCodeFromCurrentLocation() {
    lifecycleScope.launch{
        val placeFields = listOf(Place.Field.LAT_LNG)
        val request: FindCurrentPlaceRequest = FindCurrentPlaceRequest.newInstance(placeFields)

        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
                    )
                    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()
            }
        }
    }
  }

  //Habilita o botão de prassar para a proxima tela se a variavel geoName não estiver vazia.
  private fun enableNextButton(){
    if (sharedViewModel.geoName.isNotEmpty()) {
        step1ViewModel.enableNextButton(true)
    }
  }

  //Quando o frag estiver sendo destruido passa null a variavel _binding.
  override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
  }
}
```

APÊNDICE S - Código Step2Fragment

```
package com.example.apptcc.ui.addgeofence

import android.os.Build
import android.os.Bundle
import android.util.Log
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.Fragment
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.apptcc.R
import com.example.apptcc.adapters.PredictionsAdapter
import com.example.apptcc.databinding.FragmentStep2Binding
import com.example.apptcc.util.ExtensionFunctions.hide
import com.example.apptcc.util.NetworkListener
import com.example.apptcc.viewmodels.SharedViewModel
import com.example.apptcc.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.FindAutocompletePredictionsRequest
import com.google.android.libraries.places.api.net.PlacesClient
import kotlinx.coroutines.flow.collect
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch

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()

    private lateinit var placesClient: PlacesClient

    private lateinit var networkListener: NetworkListener
```

```
//Inicializa a Places API e cria o client
override fun onCreate(savedInstanceState: Bundle?) {
   super.onCreate(savedInstanceState)
   Places.initialize(requireContext(), getString(R.string.google_maps_key))
   placesClient = Places.createClient(requireContext())
}

/*
   Anexa este frag a ativ,
   chama a função checkInternetConnection que verifica a conexão com a internet.
*/
@RequiresApi(Build.VERSION_CODES.N)
override fun onCreateView(
   inflater: LayoutInflater, container: ViewGroup?,
   savedInstanceState: Bundle?
): View {
   // Inflate the layout for this fragment
   _binding = FragmentStep2Binding.inflate(layoutInflater, container, false)
   binding.sharedViewModel = sharedViewModel
   binding.step2ViewModel = step2ViewModel
   binding.lifecycleOwner = this

   checkInternetConnection()

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

   //Quando a caixa de texto for modificada chama as funções handleNextButton e getPlaces.
   binding.geofenceLocationEt.doOnTextChanged { text, _, _, _ ->
      handleNextButton(text)
      getPlaces(text)
   }

   //Volta a tela do frag Step1
   binding.step2Back.setOnClickListener {
      findNavController().navigate(R.id.action_step2Fragment_to_step1Fragment)
   }

   //Vai para a tela do frag Step3
   binding.step2Next.setOnClickListener {
      findNavController().navigate(R.id.action_step2Fragment_to_step3Fragment)
   }

   subscribeToObservers()

   return binding.root
}

//Se a caixa de texto estiver vazia desabilita o botão de passar para a proxima tela.
private fun handleNextButton(text: CharSequence?) {
   if (text.isNullOrEmpty()) {
      step2ViewModel.enableNextButton(false)
```

```
    }
}

/*
    Verifica se a vaiavel placeId não esta vazia,
    se não estiver passa ela como parametro para a função onCitySelected.
*/
private fun subscribeToObservers() {
    lifecycleScope.launch {
        predictionsAdapter.placeId.collectLatest { placeId ->
            if (placeId.isNotEmpty()) {
                onCitySelected(placeId)
            }
        }
    }
}

/*
    Quando uma cidade é selecionada as vaiavies recebem informações dessa cidade,
    desabilita o apendice, e habilita o botão de proximo.
*/
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.length)
            binding.predictionsRecyclerView.hide()
            step2ViewModel.enableNextButton(true)
        }
        .addOnFailureListener { exception ->
            Log.e("Step2Fragment", exception.message.toString())
        }

}

/*
    Verifica se o serviço de localização está ativo,
    se estiver verifica se a caixa de texto não está vazia,
    se não estiver passa as opções que batem com o que foi digitado,
    caso o serviso de localização esteja desativado é solicitado que se ative a localização.
*/
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(),
            "Porfavor ative a localização",
            Toast.LENGTH_SHORT
        ).show()
    }
}

//Verifica a conexão com a internet.
@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)
                }
            }
    }
}
```

```
    //Quando o frag estiver sendo destruido passa null a variavel _binding.
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

APÊNDICE T - Código Step3Fragment

```
package com.example.apptcc.ui.addgeofence

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

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 {
        // Inflate the layout for this fragment
        _binding =  FragmentStep3Binding.inflate(inflater, container, false)
        binding.sharedViewModel = sharedViewModel

        //Se o botão de voltar for clicado volta a tela Step2
        binding.step3Back.setOnClickListener {
            findNavController().navigate(R.id.action_step3Fragment_to_step2Fragment3)
        }

        /*
            Se o botão de finalizar a criação da geofence for selecionado a variavel geoRadius
            recebe o valor da barra de rolagem, a variavel geofenceReady que informa se a geofence
            está pronta recebe true, e passa para a tela do mapa.
        */
        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
    }

    //Quando o frag estiver sendo destruido passa null a variavel _binding.
```

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

APÊNDICE U - Código FingerprintFragment

```
package com.example.apptcc.ui.fingerprint

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.biometric.BiometricPrompt
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.apptcc.R
import com.example.apptcc.databinding.FragmentFingerprintBinding
import java.util.concurrent.Executor


class FingerprintFragment : Fragment() {

   private var _binding : FragmentFingerprintBinding? = null
   private val binding get() = _binding!!
   private lateinit var executor: Executor
   private lateinit var biometricPrompt: BiometricPrompt
   private lateinit var promptInfo: BiometricPrompt.PromptInfo

   override fun onCreateView(
      inflater: LayoutInflater, container: ViewGroup?,
      savedInstanceState: Bundle?
   ): View {
      // Inflate the layout for this fragment
      _binding = FragmentFingerprintBinding.inflate(inflater, container, false)

      executor = ContextCompat.getMainExecutor(requireContext())

      biometricPrompt = BiometricPrompt(this, executor, object :
BiometricPrompt.AuthenticationCallback() {

         //Se houver erro na autenticação é exibida uma mensagem de erro.
         override fun onAuthenticationError(errorCode: Int, errString: CharSequence) {
            super.onAuthenticationError(errorCode, errString)
            binding.authStatusTv.text = "Erro na autenticação : $errString"
            Toast.makeText(requireContext(), "Erro na autenticação : $errString",
Toast.LENGTH_SHORT).show()
         }

         //Se a digital não for reconhecida na autenticação é exibida uma mensagem de falha.
         override fun onAuthenticationFailed() {
            super.onAuthenticationFailed()
            binding.authStatusTv.text = "Falha na autenticação"
            Toast.makeText(requireContext(), "Falha na autenticação", Toast.LENGTH_SHORT).show()
         }
```

```
        //Se a autenticação for um sucesso é exibida uma mensagem de sucesso e avança para a tela
do mapa.
        override fun onAuthenticationSucceeded(result: BiometricPrompt.AuthenticationResult) {
            super.onAuthenticationSucceeded(result)
            binding.authStatusTv.text = "Autenticado com sucesso"
            Toast.makeText(requireContext(), "Autenticado com sucesso",
Toast.LENGTH_SHORT).show()
            findNavController().navigate(R.id.action_fingerprintFragment_to_mapsFragment)
        }
    })

    //Passa as mensagem a serem exibidas na caixa de autenticação.
    promptInfo = BiometricPrompt.PromptInfo.Builder()
        .setTitle("Autenticação Biometrica")
        .setSubtitle("Login usando autenticação por impressão digital")
        .setNegativeButtonText("Utilize a senha")
        .build()

    //Quando o botão de autenticação for clicado inicia o serviço de autenticação.
    binding.authBtn.setOnClickListener {
        biometricPrompt.authenticate(promptInfo)
    }
    return binding.root
  }

  //Quando o frag estiver sendo destruído passa null a variável _binding.
  override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
  }
}
```

APÊNDICE V - Código GeofencesFragment

```
package com.example.apptcc.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.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.apptcc.R
import com.example.apptcc.adapters.GeofencesAdapter
import com.example.apptcc.databinding.FragmentGeofencesBinding
import com.example.apptcc.viewmodels.SharedViewModel


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 {
        // Inflate the layout for this fragment
        _binding = FragmentGeofencesBinding.inflate(inflater, container, false)
        binding.sharedViewModel = sharedViewModel

        setupToolbar()
        setupRecyclerView()
        observeDatabase()

        return binding.root
    }

    //Se clicar na seta de voltar será direcionado a tela do mapa
    private fun setupToolbar() {
        binding.toolbar.setNavigationOnClickListener {
            findNavController().navigate(R.id.action_geofencesFragment_to_mapsFragment)
        }
    }

    /*
        Recebe o tipo de exibição das geofences na tela e
        passa a variável da classe GeofenceAdapter como parâmetro.
    */
```

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

    //Faz a leitura do banco de dados e exibe as geofences na tela.
    private fun observeDatabase() {
        sharedViewModel.readGeofences.observe(viewLifecycleOwner, {
            geofencesAdapter.setData(it)
            binding.geofencesRecyclerView.scheduleLayoutAnimation()
        })
    }

    //Quando o frag estiver sendo destruído passa null a variável _binding.
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

APÊNDICE W - Código MapsFragment

```
package com.example.apptcc.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.apptcc.R
import com.example.apptcc.databinding.FragmentMapsBinding
import com.example.apptcc.util.ExtensionFunctions.disable
import com.example.apptcc.util.ExtensionFunctions.enable
import com.example.apptcc.util.ExtensionFunctions.hide
import com.example.apptcc.util.ExtensionFunctions.show
import com.example.apptcc.util.Permissions.hasBackgroundLocationPermission
import com.example.apptcc.util.Permissions.requestBackgroundLocationPermission
import com.example.apptcc.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

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)

        //Vai para a tela de criação da geofence.
        binding.addGeofenceFab.setOnClickListener{
            findNavController().navigate(R.id.action_mapsFragment_to_add_geofence_graph)
        }

        //Vai para o historico de geofences.
        binding.geofencesFab.setOnClickListener {
            findNavController().navigate(R.id.action_mapsFragment_to_geofencesFragment)
        }

        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)
    }

    /*
        Quando o mapa estiver pronto habilita o botão de localização atual,
        habilita o clique longo no mapa, chama as funções onGeofenceReady,
        observeDatabase e backFromGeofencesFragment.
    */
    @SuppressLint("MissingPermission")
    override fun onMapReady(googleMap: GoogleMap?) {
        map = googleMap!!
        map.isMyLocationEnabled = true
        map.setOnMapLongClickListener (this)
        map.uiSettings.apply {
            isMyLocationButtonEnabled = true
            isMapToolbarEnabled = false
        }
        onGeofenceReady()
        observeDatabase()
        backFromGeofencesFragment()
    }

    /*
        Verifica se alguma geofence foi criada,
        se foi criada passa false para a variavel geofenceReady e true
        para a variavel geofencePrepared e chama as funções displayInfoMessage
        e zoomToSelectedLocation
    */
    private fun onGeofenceReady() {
        if(sharedViewModel.geofenceReady) {
            sharedViewModel.geofenceReady = false
```

```
        sharedViewModel.geofencePrepared = true
        displayInfoMessage()
        zoomToSelectedLocation()
    }
}

/*
    Informa ao usuário que de um clique longa na tela
    para que seja gerado o círculo da geofence no mapa.
*/
private fun displayInfoMessage() {
    lifecycleScope.launch {
        binding.infoMessageTextView.show()
        delay(2000)
        binding.infoMessageTextView.animate().alpha(0f).duration = 800
        delay(1000)
        binding.infoMessageTextView.hide()
    }
}

//Aproxima a tela da geofence criada.
private fun zoomToSelectedLocation() {
    map.animateCamera(
        CameraUpdateFactory.newLatLngZoom(sharedViewModel.geoLatLng, 10f),200, null
    )
}

/*
    Faz uma leitura no banco de dados buscando as geofence criadas e
    chama as funções drawCircle e drawMarker.
*/
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)
        }
    })
}

/*
    Se houver geofences no banco de dados recebe a latitude e a longitude da geofence
    selecionada e salva na variável selectedGeofence e
    chama a função zoomToGeofence passando como parâmetro a variável e o raio da geofence.
*/
private fun backFromGeofencesFragment() {
    if (args.geofenceEntity != null) {
        val selectedGeofence = LatLng(
            args.geofenceEntity!!.latitude,
            args.geofenceEntity!!.longitude
        )
```

```
        zoomToGeofence(selectedGeofence, args.geofenceEntity!!.radius)
    }
}

/*
    Se a permissão de uso do serviço de localização o no plano de fundo estiver ativa,
    verifica se a variável geofencePrepared recebeu true e
    se a localização recebida é diferente de null,
    se não é informado ao usuário que cria primeiro uma geofence,
    se não houver permissão de localização ativa é
    chamada a função requestBackgroundLocationPermission.
 */
override fun onMapLongClick(location: LatLng?) {
    if (hasBackgroundLocationPermission(requireContext())){
        if (sharedViewModel.geofencePrepared && location != null){
            setupGeofence(location)
        }else{
            Toast.makeText(
                requireContext(),
                "Primeiro crie uma nova Geofence.",
                Toast.LENGTH_SHORT
            ).show()
        }
    }else{
        requestBackgroundLocationPermission(this)
    }
}

//Verifica o resultado da permissão.
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, this)
}

/*
    Verifica se a permissão foi negada,
    se foi permanentemente negada é informado ao usuário que
    não poderá utilizar o aplicativo, se foi apenas negada é requisitada novamente.
*/
override fun onPermissionsDenied(requestCode: Int, perms: List<String>) {
    if (EasyPermissions.somePermissionPermanentlyDenied(this, perms[0])){
        SettingsDialog.Builder(requireActivity()).build().show()
    }else{
        requestBackgroundLocationPermission(this)
    }
}

/*
```

```
    Se a permissão for concedida chama a função onGeofenceReady e
    informa que o usuário de um longo clique na tela
*/
override fun onPermissionsGranted(requestCode: Int, perms: List<String>) {
    onGeofenceReady()
    Toast.makeText(
        requireContext(),
        "Permissão concedida! Pressione mapa a ser marcado até que apareça o marcador",
        Toast.LENGTH_SHORT
    ).show()
}

/*
    Verifica se o serviço de localização está ativo,
    desabilita o botão de histórico de geofences,
    desabilita a o botão de criação da geofence,
    mostra uma barra de progresso,
    chama as funções drawCircle, drawMarker, zoomToGeofence,
    captura uma foto do círculo da geofence,
    chama a função addGeofenceToDatabase que salva a geofence no banco de dados,
    chama as funções startGeofence e resetSharedValues,
    habilita o botão de histórico de geofences,
    habilita a o botão de criação da geofence,
    desabilita a barra de progresso,
    se o serviço de localização não estiver ativo é solicitado que o usuário ative.
*/
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(),
                "Por favor ative o serviço de localização",
```

```
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}

//Desenhe o círculo da geofence na tela.
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))
    )
}

//Coloca o marcador da geofence.
private fun drawMarker(location: LatLng, name: String) {
    map.addMarker(
        MarkerOptions().position(location).title(name)
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE))
    )
}

//Aproxima a tela da geofence criada.
private fun zoomToGeofence(center: LatLng, radius: Float) {
    map.animateCamera(
        CameraUpdateFactory.newLatLngBounds(
            sharedViewModel.getBounds(center, radius), 10
        ), 1000, null
    )
}

//Passa a foto tirada para a variável geoSnapshot
override fun onSnapshotReady(snapshot: Bitmap?) {
    sharedViewModel.geoSnapshot = snapshot
}

//Quando o frag estiver sendo destruído passa null a variável _binding.
override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
}
}
```

APÊNDICE X - Código PermissionFragment

```
package com.example.apptcc.ui.permission

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

@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 {
        // Inflate the layout for this fragment
        _binding = FragmentPermissionBinding.inflate(inflater, container, false)

        /*
            Se o botão de continue for clicado,
            verifica se foi concedida a permissão de
            acesso ao serviço de localização,
            e foi concedida chama a função checkFirstLaunch,
            se for negada é solicitada novamente.
        */
        binding.continueButton.setOnClickListener {
            if (Permissions.hasLocationPermission(requireContext())) {
                checkFirstLaunch()
            } else {
                Permissions.requestLocationPermission(this)
            }
        }

        return binding.root
```

```
}

/*
   Verifica se foi a primeira vez que o aplicativo foi aberto,
   se for a primeira vez passa false para a variável saveFirstLaunch,
   e passa para a tela de autenticação,
   caso não seja a primeira passa direto para a tela de autenticação.
*/
private fun checkFirstLaunch() {
   sharedViewModel.readFirstLaunch.observeOnce(viewLifecycleOwner, {firstLaunch ->
      if (firstLaunch) {
         findNavController().navigate(R.id.action_permissionFragment_to_fingerprintFragment)
         sharedViewModel.saveFirstLaunch(false)
      } else {
         findNavController().navigate(R.id.action_permissionFragment_to_fingerprintFragment)
      }
   })
}

//Verifica o resultado da permissão.
override fun onRequestPermissionsResult(
   requestCode: Int,
   permissions: Array<out String>,
   grantResults: IntArray
) {
   super.onRequestPermissionsResult(requestCode, permissions, grantResults)
   EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, this)
}

/*
   Verifica se a permissão foi negada,
   se foi permanentemente negada é informado ao usuário que
   não poderá utilizar o aplicativo, se foi apenas negada é requisitada novamente.
*/
override fun onPermissionsDenied(requestCode: Int, perms: List<String>) {
   if (EasyPermissions.somePermissionPermanentlyDenied(this, perms[0])){
      SettingsDialog.Builder(requireActivity()).build().show()
   }else {
      Permissions.requestLocationPermission(this)
   }
}

/*
   Se a permissão for concedida chama a função onGeofenceReady e
   informa que o usuário de um longo clique na tela
*/
override fun onPermissionsGranted(requestCode: Int, perms: List<String>) {
   Toast.makeText(
      requireContext(),
      "Permissão Concedida! pressione o botão 'Próximo' para prosseguir",
      Toast.LENGTH_SHORT
   ).show()
```

```
    }

    //Quando o frag estiver sendo destruído passa null a variável _binding.
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

APÊNDICE Y Código MainActivity

```
package com.example.apptcc.ui

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

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

      //Se a permissão de acesso a localização já tiver sido concedida passa para a tela de
autenticação.
      if (Permissions.hasLocationPermission(this)){

findNavController(R.id.navHostFragment).navigate(R.id.action_permissionFragment_to_fingerprintFr
agment)
      }
   }
}
```

APÊNDICE Z – Código Constants

```
package com.example.apptcc.util

//Todas as constantes usadas no projeto
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 AA – Código ExtensionFunctions

```
package com.example.apptcc.util

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

object  ExtensionFunctions {

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

   //Desabilita a visualização.
   fun View.disable(){
      this.isEnabled = false
   }

   //Habilita a visualização.
   fun View.show(){
      this.visibility = View.VISIBLE
   }

   //Desabilita a visualização.
   fun  View.hide(){
      this.visibility = View.GONE
   }

   //Quando o observador for modificado remove o observador.
   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 AB – Código MyDiffUtil

```
package com.example.apptcc.util

import androidx.recyclerview.widget.DiffUtil

class MyDiffUtil<T>(
      private val oldList: List<T>,
      private val newList: List<T>
): DiffUtil.Callback() {

   //Recebe o tamanho da lista antiga
   override fun getOldListSize(): Int {
      return oldList.size
   }

   //Recebe o tamanho da nova lista
   override fun getNewListSize(): Int {
      return newList.size
   }

   //Verifica se os items da lista são o mesmo e coloca em uma nova posição.
   override fun areItemsTheSame(oldItemPosition: Int, newItemPosition: Int): Boolean {
      return oldList[oldItemPosition] === newList[newItemPosition]
   }

   //Verifica se os conteudos dos items da lista são o mesmo e coloca em uma nova posição.
   override fun areContentsTheSame(oldItemPosition: Int, newItemPosition: Int): Boolean {
      return oldList[oldItemPosition] == newList[newItemPosition]
   }
}
```

APÊNDICE AC – Código NetworkListener

```
package com.example.apptcc.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

class NetworkListener: ConnectivityManager.NetworkCallback() {

   private val isNetworkAvailable = MutableStateFlow(false)

   //Verifica a capacidade de conexão com a internet do dispositivo, e este está conectado.
   @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
   }

   //Quando a conexão estiver ativa a variavel isNetworkAvailable recebe true
   override fun onAvailable(network: Network) {
      isNetworkAvailable.value = true
   }

   //Quando a conexão estiver desativada a variavel isNetworkAvailable recebe false
   override fun onLost(network: Network) {
      isNetworkAvailable.value = false
   }
}
```

APÊNDICE AD – Código Permissions

```
package com.example.apptcc.util
import android.Manifest
import android.content.Context
import androidx.fragment.app.Fragment
import com.example.apptcc.util.Constants.PERMISSION_BACKGROUND_LOCATION_REQUEST_CODE
import com.example.apptcc.util.Constants.PERMISSION_LOCATION_REQUEST_CODE
import com.vmadalin.easypermissions.EasyPermissions

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

   //Solicita a permissão de localização
   fun requestLocationPermission(fragment: Fragment){
      EasyPermissions.requestPermissions(
         fragment,
         "Essa aplicação não irá funcionar sem a permissão de uso da localização",
         PERMISSION_LOCATION_REQUEST_CODE,
         Manifest.permission.ACCESS_FINE_LOCATION
      )
   }

   //Verifica se tem a permissão de localização no plano de fundo
   fun hasBackgroundLocationPermission(context: Context) =
      EasyPermissions.hasPermissions(
         context,
         Manifest.permission.ACCESS_BACKGROUND_LOCATION
      )

   //Solicita a permissão de localização no plano de fundo
   fun requestBackgroundLocationPermission(fragment: Fragment){
      EasyPermissions.requestPermissions(
         fragment,
         "Essa aplicação não irá funcionar sem a permissão de uso da localização",
         PERMISSION_BACKGROUND_LOCATION_REQUEST_CODE,
         Manifest.permission.ACCESS_BACKGROUND_LOCATION
      )
   }
}
```

APÊNDICE AE – Código SharedViewModel

```
package com.example.apptcc.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.apptcc.broadcastreciver.GeofenceBroadcastReceiver
import com.example.apptcc.data.DataStoreRepository
import com.example.apptcc.data.GeofenceEntity
import com.example.apptcc.data.GeofenceRepository
import com.example.apptcc.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

@HiltViewModel
class SharedViewModel @Inject constructor(
   application: Application,
   private val dataStoreRepository: DataStoreRepository,
   private val geofenceRepository: GeofenceRepository
): AndroidViewModel(application) {

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

   var geoId: Long = 0L
   var geoName: String = "Default"
   var geoCountryCode: String = ""
   var geoLocationName: String = "Procurar Cidade"
   var geoLatLng: LatLng = LatLng(0.0, 0.0)
   var geoRadius: Float = 500f
   var geoSnapshot: Bitmap? = null
```

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

fun resetSharedValues() {
   geoId = 0L
   geoName = "Default"
   geoCountryCode = ""
   geoLocationName = "Procurar Cidade"
   geoLatLng = LatLng(0.0, 0.0)
   geoRadius = 500f
   geoSnapshot = null

   geoCitySelected = false
   geofenceReady = false
   geofencePrepared = false
}

//Recebe a primeira leitura do banco de dados na variavel readFirstLaunch
val readFirstLaunch = dataStoreRepository.readFirstLaunch.asLiveData()

//Salva a primeria leitura do banco de dados
fun saveFirstLaunch(firstLaunch: Boolean) =
   viewModelScope.launch(Dispatchers.IO) {
      dataStoreRepository.saveFirstLaunch(firstLaunch)
   }

//Recebe a leitura do banco de dados na variavel readGeofences
val readGeofences = geofenceRepository.readGeofences.asLiveData()

//Adiciona a geofence ao banco de dados.
fun addGeofence(geofenceEntity: GeofenceEntity) =
   viewModelScope.launch(Dispatchers.IO) {
      geofenceRepository.addGeofence(geofenceEntity)
   }

//Remove a geofence do banco de dados
fun removeGeofence(geofenceEntity: GeofenceEntity) =
   viewModelScope.launch(Dispatchers.IO) {
      geofenceRepository.removeGeofence(geofenceEntity)
   }

//Recebe as entidades da geofence e chama a função addGeofence
fun addGeofenceToDatabase(location: LatLng) {
   val geofenceEntity =
      GeofenceEntity(
         geoId,
         geoName,
         geoLocationName,
         location.latitude,
         location.longitude,
```

```
            geoRadius,
            geoSnapshot!!
        )
    addGeofence(geofenceEntity)
}

//Retorna a localização do usuario em referencia a geofence.
private fun setPendingIntent(geoId: Int): PendingIntent {
    val intent = Intent(app, GeofenceBroadcastReceiver::class.java)
    return PendingIntent.getBroadcast(
        app,
        geoId,
        intent,
        PendingIntent.FLAG_UPDATE_CURRENT
    )
}

/*
    Verifica a permissão de localização no plano e fundo,
    se for concedida é inicalizada a geofence e sua requisição,
    verifica se a geofenec foi criada com sucesso, se foi é informado o sucesso,
    se não é informada a falha,
    se não for concedida a permissão de localização no plano e fundo
    o usuario é informado que a permissão não foi concedidada.
*/
@SuppressLint("MissingPermission")
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.d("Geofence", it.message.toString())
            }
        }
    } else {
        Log.d("Geofence", "Permissão não concedida.")
    }
}

//Remove a geofence e verifica e foi removida com sucesso.
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
    }
}

//Calcula os cantos 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)
}

//Verifica as configurações de localização do dispositivo.
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 AF – Código Step1ViewModel

```
package com.example.apptcc.viewmodels

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

class Step1ViewModel: ViewModel() {

   private val _nextButtonEnabled = MutableLiveData(false)
   val nextButtonEnabled: LiveData<Boolean> get() = _nextButtonEnabled

   //Habilida o botão para passar para proxima tela.
   fun enableNextButton(enable: Boolean){
      _nextButtonEnabled.value = enable
   }
}
```

APÊNDICE AG – Código Step2ViewModel

```
package com.example.apptcc.viewmodels

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

class Step2ViewModel: ViewModel() {

    private val _nextButtonEnabled = MutableLiveData(false)
    val nextButtonEnabled: LiveData<Boolean> get() = _nextButtonEnabled

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

    //Habilida o botão para passar para proxima tela.
    fun enableNextButton(enable: Boolean){
        _nextButtonEnabled.value = enable
    }

    //Passa o valor online como parametro para a variavel _internetAvailable
    fun setInternetAvailability(online: Boolean){
        _internetAvailable.value = online
    }
}
```

APÊNDICE AH – Código MyApplication

```
package com.example.apptcc

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

@HiltAndroidApp
//Construtor da aplicação
class MyApplication: Application()
```

APÊNDICE AI – Códigos da Interface

Pasta anim - from_bottom.xml

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

   <translate
      android:duration="300"
      android:fromYDelta="100%"
      android:toYDelta="0%" />

</set>
```

Pasta anim - from_left.xml

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

   <translate
      android:duration="300"
      android:fromXDelta="-100%"
      android:toXDelta="0%" />

</set>
```

Pasta anim - from_right.xml

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

   <translate
      android:duration="300"
      android:fromXDelta="100%"
      android:toXDelta="0%" />

</set>
```

Pasta anim - from_top.xml

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

   <translate
      android:duration="300"
      android:fromYDelta="-100%"
      android:toYDelta="0%" />

</set>
```

Pasta anim - recyclerview_item_animation.xml

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

   <alpha
      android:duration="400"
      android:fromAlpha="0"
      android:toAlpha="1" />
   <translate
      android:duration="300"
      android:fromYDelta="-100%"
      android:toYDelta="0%" />

</set>
```

Pasta anim - recyclerview_layout_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
   android:animation="@anim/recyclerview_item_animation"
   android:animationOrder="normal"
   android:delay="15%"/>
```

Pasta anim - to_bottom.xml

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

   <translate
      android:duration="300"
      android:fromYDelta="0%"
      android:toYDelta="100%" />

</set>
```

Pasta anim – to_left.xml

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

   <translate
      android:duration="300"
      android:fromXDelta="0%"
      android:toXDelta="-100%" />

</set>
```

Pasta anim - to_right.xml

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

   <translate
```

```
        android:duration="300"
        android:fromXDelta="0%"
        android:toXDelta="100%" />

</set>
```

Pasta anim - to_top.xml

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

    <translate
        android:duration="300"
        android:fromYDelta="0%"
        android:toYDelta="-100%" />

</set>
```

Pasta color - view_state_background_color.xml

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

    <item android:color="@color/gray" android:state_enabled="false"/>
    <item android:color="@color/blue_500"/>

</selector>
```

Pasta layout - 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>
```

Pasta layout - fragment_fingerprint.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.permission.PermissionFragment">


   <ImageView
      android:id="@+id/imageFingerprint"
      android:layout_width="168dp"
      android:layout_height="185dp"
      android:layout_marginBottom="150dp"
      android:contentDescription="@string/finger_content"
      android:src="@drawable/fingerprint_free_download_png"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />


   <TextView
      android:id="@+id/authStatusTv"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="16dp"
      android:text="@string/authStatusTv"
      android:textAlignment="center"
      android:textSize="16sp"
      android:textColor="@color/black"
      android:textStyle="bold"
      app:layout_constraintEnd_toEndOf="@+id/imageFingerprint"
      app:layout_constraintStart_toStartOf="@+id/imageFingerprint"
      app:layout_constraintTop_toBottomOf="@+id/imageFingerprint" />

   <Button
      android:id="@+id/authBtn"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_marginTop="16dp"
      android:text="@string/autenticar"
      app:layout_constraintEnd_toEndOf="@+id/authStatusTv"
      app:layout_constraintStart_toStartOf="@+id/authStatusTv"
      app:layout_constraintTop_toBottomOf="@+id/authStatusTv" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Pasta layout - 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.apptcc.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>
```

Pasta layout - 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="32dp"
    android:layout_marginBottom="32dp"
    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: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/pressione_e_segure_para_adicionar_uma_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>
```

Pasta layout - 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"
   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="0dp"
      android:layout_height="wrap_content"
      android:layout_marginBottom="48dp"
      android:text="@string/permission_description"
      android:textAlignment="center"
      android:textColor="@color/black"
      android:textSize="16sp"
      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_require_textView"
      android:layout_width="wrap_content"
      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="181dp"
        android:layout_height="219dp"
        android:layout_marginBottom="100dp"
        android:src="@drawable/mapicon"
        app:layout_constraintBottom_toTopOf="@+id/permission_require_textView"
        app:layout_constraintEnd_toEndOf="@+id/permission_require_textView"
        app:layout_constraintStart_toStartOf="@+id/permission_require_textView"
        android:contentDescription="@string/image_location_todo" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Pasta layout - 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.apptcc.viewmodels.SharedViewModel" />
        <variable
            name="step1ViewModel"
            type="com.example.apptcc.viewmodels.Step1ViewModel" />

    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        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/nome"
    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="57dp"
        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/proximo"
    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/anterior"
```

```
        android:textColor="@color/gray"
        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>
```

Pasta layout - fragment_step2.xml

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

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

  <androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    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="57dp"
        android:drawableStart="@drawable/ic_search"
        android:drawablePadding="16dp"
        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/white"
    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_textInputLayout" />
```

```
        <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/proximo"
            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/anterior"
            android:textColor="@color/gray"
            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_textInputLayout"
            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>
```

Pasta layout - 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.apptcc.viewmodels.SharedViewModel"/>
```

```
</data>

<androidx.constraintlayout.widget.ConstraintLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   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/_500_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="500.0"
        android:value="500.0"
        android:valueFrom="500.0"
        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"
        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:enabled="true"
        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/anterior"
        android:textColor="@color/gray"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

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

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.apptcc.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"
    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"
    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"
    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>
```

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">

  <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"
      setCity="@{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"
      setCountry="@{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>
```

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.apptcc.ui.permission.PermissionFragment"
    android:label="fragment_permission"
    tools:layout="@layout/fragment_permission" >
    <action
      android:id="@+id/action_permissionFragment_to_fingerprintFragment"
      app:destination="@id/fingerprintFragment"
      app:popUpTo="@id/permissionFragment"
      app:popUpToInclusive="true" />
  </fragment>
  <fragment
```

```
    android:id="@+id/fingerprintFragment"
    android:name="com.example.apptcc.ui.fingerprint.FingerprintFragment"
    android:label="fragment_fingerprint"
    tools:layout="@layout/fragment_fingerprint" >
    <action
        android:id="@+id/action_fingerprintFragment_to_mapsFragment"
        app:destination="@id/mapsFragment"
        app:enterAnim="@anim/from_right"
        app:exitAnim="@anim/to_left"
        app:popEnterAnim="@anim/from_left"
        app:popExitAnim="@anim/to_right"
        app:popUpTo="@id/fingerprintFragment"
        app:popUpToInclusive="true" />
</fragment>
<fragment
    android:id="@+id/mapsFragment"
    android:name="com.example.apptcc.ui.maps.MapsFragment"
    android:label="fragment_maps"
    tools:layout="@layout/fragment_maps" >
    <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"
        app:popUpTo="@id/mapsFragment"
        app:popUpToInclusive="true" />
    <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"
        app:popUpTo="@id/mapsFragment"
        app:popUpToInclusive="true" />
    <argument
        android:name="geofenceEntity"
        app:argType="com.example.apptcc.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.apptcc.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.apptcc.ui.addgeofence.Step2Fragment"
      android:label="fragment_step2"
      tools:layout="@layout/fragment_step2" >
      <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" />
      <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" />
    </fragment>
    <fragment
      android:id="@+id/step3Fragment"
      android:name="com.example.apptcc.ui.addgeofence.Step3Fragment"
      android:label="fragment_step3"
      tools:layout="@layout/fragment_step3" >
      <action
        android:id="@+id/action_step3Fragment_to_step2Fragment3"
        app:destination="@id/step2Fragment"
        app:enterAnim="@anim/from_left"
        app:exitAnim="@anim/to_right"
        app:popUpToInclusive="true" />
    </fragment>
    <action
      android:id="@+id/action_global_mapsFragment"
      app:destination="@id/mapsFragment"
      app:popUpTo="@id/add_geofence_graph"
      app:popUpToInclusive="true" />
  </navigation>
  <fragment
    android:id="@+id/geofencesFragment"
    android:name="com.example.apptcc.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:enterAnim="@anim/from_left"
            app:exitAnim="@anim/to_right"
            app:launchSingleTop="true"
            app:popUpTo="@id/geofencesFragment"
            app:popUpToInclusive="true" />
    </fragment>
    <action
        android:id="@+id/action_step1Fragment_to_mapsFragment"
        app:destination="@id/mapsFragment"
        app:enterAnim="@anim/from_top"
        app:exitAnim="@anim/to_bottom"
        app:launchSingleTop="true"
        app:popUpTo="@id/step1Fragment"
        app:popUpToInclusive="true" />
    <action
        android:id="@+id/action_step3Fragment_to_mapsFragment"
        app:destination="@id/mapsFragment"
        app:enterAnim="@anim/from_right"
        app:exitAnim="@anim/to_left"
        app:launchSingleTop="true"
        app:popUpTo="@id/step3Fragment"
        app:popUpToInclusive="true" />

</navigation>
```

Pasta values - colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <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">#eb4f6f</color>
    <color name="gray">#CCCCCC</color>
</resources>
```

Pasta values - strings.xml

```
<resources>
    <string name="app_name">AppTCC</string>
    <string name="hello_blank_fragment">Hello blank fragment</string>
```

```
    <string name="permission_description">
        Para usar esse aplicativo é necessário requisitar a permissão de uso da localização,
        Caso seja recusada não poderemos estar lhe provendo o serviço.
    </string>
    <string name="toque_no_sensor">Toque no Sensor</string>
    <string name="permission_required">Solicitação de Permissão</string>
    <string name="continue_btn">Próximo</string>
    <string name="autenticar">Autenticar</string>
    <string name="authStatusTv">Clique no botão para começar a autenticação.</string>
    <string name="anterior">Anterior</string>
    <string name="proximo">Próximo</string>
    <string name="nome">Nome</string>
    <string name="one_three">1/3</string>
    <string name="geofence">Geofence</string>
    <string name="finger_content">finger_content</string>
    <string name="img_laction">img_laction</string>
    <string name="image_location_todo">TODO</string>
    <string name="two_three">2/3</string>
    <string name="location">Localização</string>
    <string name="powered_by_google">Powered by Google</string>
    <string name="three_three">3/3</string>
    <string name="done">Criar</string>
    <string name="_500_0_m">500.0 m</string>
    <string name="choose_a_radius_for_your_geofence">Escolha o raio da sua geofence</string>

    <string name="display_meters">%1$s m</string>
    <string name="display_kilometers">%1$s km</string>
    <string name="pressione_e_segure_para_adicionar_uma_geofence">Pressione e segure para
adicionar uma geofence</string>
    <string name="no_geofence_found">Não foi encontrada nenhuma geofence</string>
    <string name="location_txt">Localização:</string>
    <string name="coordinates">Coordenadas:</string>
    <string name="radius">Raio:</string>
</resources>
```

Pasta xml - 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>
```