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

GEOFENCING AUXILIANDO OS PRODUTORES RURAIS

SHIRLANO CANDIDO DIAS FILHO

GOIÂNIA
2021

SHIRLANO CANDIDO DIAS FILHO

GEOFENCING AUXILIANDO OS PRODUTORES RURAIS

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

Orientador(a):

Prof. M.E.E. Marcelo Antonio Adad de Araújo

GOIÂNIA
2021

SHIRLANO CANDIDO DIAS FILHO

GEOFENCING AUXILIANDO OS PRODUTORES RURAIS

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 Politécnica, da Pontifícia Universidade Católica de Goiás, em 08/12/2021.

Prof. MSc. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca examinadora:

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

Prof. MSc Pedro Araújo Valle

Prof. M.E.E Carlos Alexandre Ferreira de Lima

GOIÂNIA
2021

AGRADECIMENTOS

A Deus pela minha vida, pelo conhecimento e pelas oportunidades.

Em especial ao meus pais, pois é graça aos seus esforços que hoje posso concluir meu curso, e que são pilares na minha formação como ser humano. Meus maiores e melhores orientadores da vida.

Ao meu Professor MEE. Marcelo Antonio Adad de Araújo pelo apoio e confiança depositada e por ter aceitado acompanhar-me neste projeto. O seu empenho foi imprescindível para a minha motivação à medida que as dificuldades iam surgindo ao longo do caminho.

Gostaria também de agradecer aos membros da banca examinadora, em ênfase ao Prof. MSc. Pedro Araújo Valle e ao Prof. MEE Carlos Alexandre Ferreira de Lima, por ter aceito convite de participação nesse momento tão importante para mim.

À minha noiva cuja presença foi essencial para a conclusão deste trabalho. Grato pelo apoio incondicional oferecido em todos os aspectos e pela sua compreensão com as minhas horas de ausência.

À coordenação da Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás por todo apoio que deram durante toda a minha formação acadêmica.

Aos professores do Curso de Engenharia da Computação que me forneceram inúmeras bases necessárias para a realização deste trabalho, agradeço com profunda gratidão pelo vosso profissionalismo.

A todos que direta ou indiretamente colaboraram para materialização deste trabalho.

RESUMO

Buscando auxiliar os produtores rurais, o presente trabalho visa a construção de um aplicativo para dispositivos móveis que possa gerir as diversas atividades presentes em um âmbito rural. Com a utilização de um sistema georreferenciado, o aplicativo reconhece a localização do usuário, verifica se o mesmo adentrou em uma determinada região de sua propriedade, e de maneira automática, retorna todas as informações, alocadas em um banco de dados local, sobre aquela determinada área de sua propriedade. Deste modo o aplicativo busca manter o proprietário sempre bem-informado a respeito de cada linha de produção de sua fazenda.

Palavra-Chave: *Android; Geofencing; Mapeamento*

ABSTRACT

Seeking to assist productive producers, the present work aims to build an application for devices that can manage the various activities present in a rural environment. With the use of a georeferenced system, the application recognizes the user's location, verifies whether the user has entered a certain region, already provided by the user, and automatically returns all information, allocated in a local database, about that specific area of your property. In this way, the application seeks to keep the owner always feeling well about each production line on his farm.

Keywords: *Android; Geofencing; Mapeamento*

LISTA DE FIGURAS

Figura 1 – Linhas Longitudinais e Latitudinais..... 16
Figura 2 – Linhas Latitudinais..... 19
Figura 3 – Linhas Longitudinais..... 18
Figura 4 – Distribuição das 6 órbitas com os 24 satélites..... 20
Figura 5 – Estações de Controle do GPS..... 21
Figura 6 – Receptores de Usuários do GPS..... 21
Figura 7 – Diagrama de blocos do Receptor GPS..... 22
Figura 8 – Arquitetura do Sistema operacional Android..... 27
Figura 9 – Montagem operacional Java..... 29
Figura 10 – Estrutura de Pasta *Android Studio*..... 35
Figura 11 – Interface inicial do *Google* Mapas..... 40
Figura 12 – Interface *Google* Mapa de Satélite..... 41
Figura 13 –Exemplo do *onMapReady()*..... 42
Figura 14 –Marcador do Maps na PUC-GO..... 45
Figura 15 –Declaração de Permissões..... 44
Figura 16 – Página Educacional de Permissão..... 45
Figura 17 – Permissão de Localização em Primeiro Plano..... 46
Figura 18 – Permissão de Localização em Segundo Plano..... 47
Figura 19 –Requerer Permissão Novamente..... 47
Figura 20 – Alterar Perdições em Informações do App..... 48
Figura 21 – Aba Informações..... 48
Figura 22 – Botão Adicionar *Geofencing*..... 49
Figura 23 – Tipo, Nome e Raio da *Geofencing*..... 50
Figura 24 – Dados Galinheiro..... 51
Figura 25 – Dados Mangueiro..... 52
Figura 26 –Dados Gado de Corte..... 53
Figura 27 –Dados Gado de Leite..... 54
Figura 28 –Dados Cultura..... 55
Figura 29 – *Geofencing* Criada..... 56
Figura 30 – Projeto *Firebase*..... 57
Figura 31 – Página de *Login*..... 58
Figura 32 – Página de Criação de Conta..... 59

Figura 33 – Página *Firebase* de Usuários. 60
Figura 34 – Lista de *Geofence*s..... 61
Figura 35 – Dados da *Geofence*..... 62
Figura 36 – Estrutura Final do *Realtime Database*..... 64
Figura 37 – *Broadcast Geofence Receiver*..... 66

LISTA DE SIGLAS

API	*Application Programming Interface*
PIB	Produto Interno Bruto
GPS	*Global Positioning System*
RF	Rádio frequência
FI	Frequência intermediária
CI	Circuito integrado
AGPS	*Assisted Global Positioning System*
Wi-Fi	*Wireless Fidelity*
GSM	*Global System for Mobile Communications*
APP	Aplicativos
ART	*Android Runtime*
NDK	*Native Development Kit*
API	*Application Programming Interface*
JVM	Máquina virtual Java
OO	Orientada a objeto
FP	Programação funcional
IDE	Ambiente de Desenvolvimento Integrado
SGBD	Sistemas Gerenciadores de Banco de Dados
SQL	*Structured Query Language*
NoSQL	*Not Only Structured Query Language*
TCC2	Trabalho de Conclusão de Curso para o segundo semestre.
TCC	Trabalho de Conclusão de Curso
SDK	*Software Development Kit*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Justificativa 12
1.2 Questão de Pesquisa 13
1.3. Objetivos 14
1.3.1. Objetivo Geral 14
1.3.2. Objetivos Específicos 14
1.4. Procedimentos Metodológicos 14
1.5 Estrutura do Trabalho 15
2 ESTADO DA ARTE 16
2.1 Coordenadas Geográficas 16
2.2 Sistema De Posicionamento Global (GPS) 18
2.2.1 GPS e seus segmentos 19
2.2.2 Funcionamento do GPS 23
2.2.3 Geolocalização em *smartphones* 24
2.3 Geofencing 25
2.4 Sistema Operacional *Android* 26
2.5 *Application Programming Interface* (API) 28
2.6 Linguagem Java 29
2.7 Linguagem *Kotlin* 30
2.8 Ambiente de Desenvolvimento Integrado (IDE) 31
2.9 Banco de Dados 31
3 PROPOSTA DE SOLUÇÃO 33
3.1 *Kotlin* 33
3.2 *Android* Studio 34
3.3 *Android SDK* 36

3.4 *Google* Mapas API 37
3.5 *Google Firebase* 37
3.6 Linguagem XML 38
3.7 Desenvolvimento da Aplicação 39
3.7.1 Implementação do Mapa 39
3.7.2 Implementação das Permissões 43
3.7.3 Implementação das *Geofencing* 49
3.7.4 Implementação do *Login* do Usuário 57
3.7.5 Implementação da Exibição dos Dados 61
3.7.6 Armazenamento da *Geofencing* no Banco de Dados 63
3.7.7 Carregamento das *Geofences* do Banco de Dados 65
3.7.8 Monitoramento da *Geofencing* 66
4 CONSIDERAÇÕES FINAIS 68
4.1 Trabalhos Futuros 69
REFERÊNCIAS 70
APÊNDICES 75
APÊNDICE A - Código build.gradle(Project) 75
APÊNDICE B - Código build.gradle(Module) 75
APÊNDICE C – Código AddGeofenceFragment.kt 78
APÊNDICE D – Código CreateAccountFragment.kt 81
APÊNDICE E – Código EditCulturaFragment.kt 83
APÊNDICE F – Código EditGadoDeCorteFragment.kt 85
APÊNDICE F – Código EditGadoDeLeiteFragment.kt 87
APÊNDICE G – Código EditGalinheiroFragment.kt 89
APÊNDICE H – Código EditGeofenceFragment.kt 90
APÊNDICE I – Código EditMangueiroFragment.kt 95
APÊNDICE J – Código GeofenceReceiver.kt 97

APÊNDICE K – Código GeofencesFragment.kt 103
APÊNDICE L – Código Geofencing.kt 107
APÊNDICE M – Código Live.kt 109
APÊNDICE N – Código LiveAdapter.kt 109
APÊNDICE O – Código LoginFragment.kt 111
APÊNDICE P – Código MainActivity.kt 114
APÊNDICE Q – Código MapsFragment.kt 115
APÊNDICE R – Código PermissionFragment.kt 123
APÊNDICE S – Código de Interface 125

1 INTRODUÇÃO

O presente trabalho tem como objetivo o estudo e desenvolvimento de um aplicativo *mobile* para o sistema operacional Android, que seja capaz de auxiliar na gestão de informações e atividades usuais de uma propriedade rural típica, tomando como base a geolocalização na propriedade do usuário que será fornecida por uma *Application Programming Interface* (APIs) vinculada a um dispositivo móvel, com o intuito de retornar de forma automática as informações pertinentes a uma área específica, das diversas partes da propriedade rural ou região identificável da propriedade, permitindo ao usuário buscar às informações das diversas partes e itens que compõem a propriedade agrária, de forma manual no *smartphone* tais informações se apresentarão no momento em que o usuário se aproximar de uma das estruturas da fazenda ou chácara, apresentando as informações pertinentes.

A aplicação irá contar com um banco de dados, onde será armazenada as informações pertinentes, utilizando-se de uma *geofencing* pré-determinada da propriedade rural em questão. As informações irão variar de acordo com as áreas relacionadas às partes da propriedade. Supondo-se que na propriedade exista uma área destinada à criação de gado para a produção leiteira, o *geofencing* determinado para essa área conterá dados alocados pertinentes aos animais, alimentação, peso e espaçamento métrico desta construção, todos devidamente alocados no banco de dados relacionado a área de produção leiteira. A aplicação deverá ser capaz de identificar quando o usuário entrar em cada *geofencing* específica e com isso notificá-lo com informações pertinentes a cada parte da propriedade.

1.1 Justificativa

Sabe-se que pelo menos 25% do PIB (Produto Interno Bruto) do Brasil é representado por negócios no meio rural. Objetiva-se então, desenvolver um aplicativo para *smartphone* de forma que possa auxiliar o produtor rural de médio e pequeno porte e até de grande porte parcialmente, para que se possa cuidar melhor de sua propriedade utilizando-se do histórico das partes componentes do agronegócio, e assim poder tomar as melhores decisões (VECCHIO, 2018).

Um dos grandes problemas apresentados na agricultura é a dificuldade de lidar com vários ambientes diferentes de produção. Um agricultor pode lidar com várias áreas de produção

em sua propriedade. Esses espaços vão desde diversos tipos de lavouras até diferentes tipos de animais e diversos tipos de manejo. Esta diversificação de produtos gera grande volume de informações que muitas vezes não é gerenciado de forma correta ou coesa, o que acaba impactando diretamente no lucro envolvido.

Uma propriedade que tem como fonte de renda a produção de leite, ovos, carne bovina, produção de bezerros para comercialização dentre outros, tem um volume grande de dados que necessitam ser armazenados e gerenciados de forma a fornecer informações para futuras tomada de decisões.

Observa-se que cada parte da propriedade tem um conjunto de dados específicos que torna ainda mais complexo o gerenciamento das informações. Na simples produção de ovos, se o agricultor não possuir um relatório mensal como se saberá a quantidade de ovos produzidos, de consumo de insumos, consumo de alimento, despesas com medicação dos animais dentre outros, indica-se o uso de um banco de dados relacional a cada área de produção na propriedade.

É imprescindível para qualquer produtor rural manter um registro fiel de sua propriedade. Se este registro puder ser automaticamente acessado quando o produtor se aproxima de qualquer das partes de sua posse fica praticamente impossível se saber no final do mês se a produção deu lucro ou prejuízo ou até mesmo quantificar este resultado. Isso será muito melhorado com uso da geolocalização pois à medida que se aproximar com um *smartphone* o histórico será apresentado para cada uma das partes.

O uso dos *smartphones* na zona rural cresceu chegando a aproximadamente 20% dos celulares com internet no Brasil (CARVALHO, 2015).

Atualmente sete em cada dez produtores rurais gaúchos usam *smartphone* como ferramenta de gestão de algum assunto relacionado a propriedade rural. (COLUSSI, 2018)

Com base nestas afirmações conclui-se que aplicações móveis vêm ganhando cada vez mais espaço neste ramo. Um dos recursos presente nos *smartphone*s é o *Global Positioning System* (GPS) que será utilizado para o desenvolvimento da ferramenta do presente trabalho.

Neste contexto o trabalho de TCC se justifica ao propor o desenvolvimento de um sistema de gestão rural, para aplicativos móveis, que seja capaz de gerenciar de forma prática e fácil as várias áreas em uma propriedade de forma abrangente.

1.2 Questão de Pesquisa

O questionamento da pesquisa é observar e desenvolver um aplicativo o qual seja capaz de reconhecer uma *geofence* pré-determinada dentro de uma propriedade rural, e através deste

reconhecimento, possibilitar a tomada de ações na tela do *smartphone* do usuário, usando as informações estocadas em um banco de dados pertinentes aquele local.

1.3. Objetivos

Desenvolver um aplicativo *Android* com base na tecnologia *geofencing*, com intuito de gerenciar atividades e informações de uma propriedade rural, tomando como base a localização do usuário, e as localizações dos diversos ambientes que compõem a propriedade, como por exemplo a maternidade, o curral, a granja e outros. De forma a trazer na tela do *smartphone* do usuário, as informações pertinentes aos vários ambientes, como por exemplo nascimentos de crias, quantidade de animais, quantidade de alimentos utilizados, histórico do animal dentre outros relacionadas àquela área.

1.3.1. Objetivo Geral

Desenvolver um sistema para celular capaz de identificar uma área de produção pré-determinada da propriedade rural e com isso retornar ao usuário os dados e informações daquela mesma área e desenvolver um banco de dados com as diversas informações da propriedade rural.

1.3.2. Objetivos Específicos

- Utilizar de forma impactante o *Geofencing*.
- Implementar um aplicativo *Android* utilizando *Geofencing*
- Produzir banco de dados a ser gerenciado pelo aplicativo e relacionado a propriedade rural em questão.
- Desenvolver o sistema de forma a abarcar esses objetivos específicos e atribuir mais funcionalidades.

1.4. Procedimentos Metodológicos

O presente estudo se inicia com uma revisão Bibliográfica detalhada em meios de comunicações confiáveis como artigos científicos, monografias, dissertações de mestrados,

revistas, livros dentre outros. Com intuito de aprofundar no assunto do presente trabalho, sabe-se que dessa forma é possível conhecer o que já foi publicado em relação aos assuntos que serão abordados pelo presente trabalho no decorrer do desenvolvimento. (SANTOS, 2018)

Conjuntamente a revisão bibliográfica, será descrito a solução do trabalho e seu estado da arte. Com intuito de agregar conhecimento trazendo informações atualizadas sobre cada parte do assunto.

Por fim serão mostrados todos os passos a serem tomados na implementado da solução e respectivamente as conclusões dos resultados obtidos.

1.5 Estrutura do Trabalho

O trabalho contará com 4 capítulos aonde:

- O capítulo 1 tem caráter introdutório buscando descrever de forma ampla, sucinta e explicativa o tema e a solução encontrada.
- O capítulo 2 apresenta a fundamentação teórica sobre todos os conceitos utilizados para a solução do problema.
- O capítulo 3 descreve detalhadamente todo o processo de desenvolvimento que envolve a solução do problema.
- O capítulo 4 relata as considerações finais com análise nos resultados obtidos e sugestão de trabalhos futuros baseado na solução implementada, e conclusões a respeito do tema abordado.
- E as referências bibliográficas aplicada ao trabalho.

2 ESTADO DA ARTE

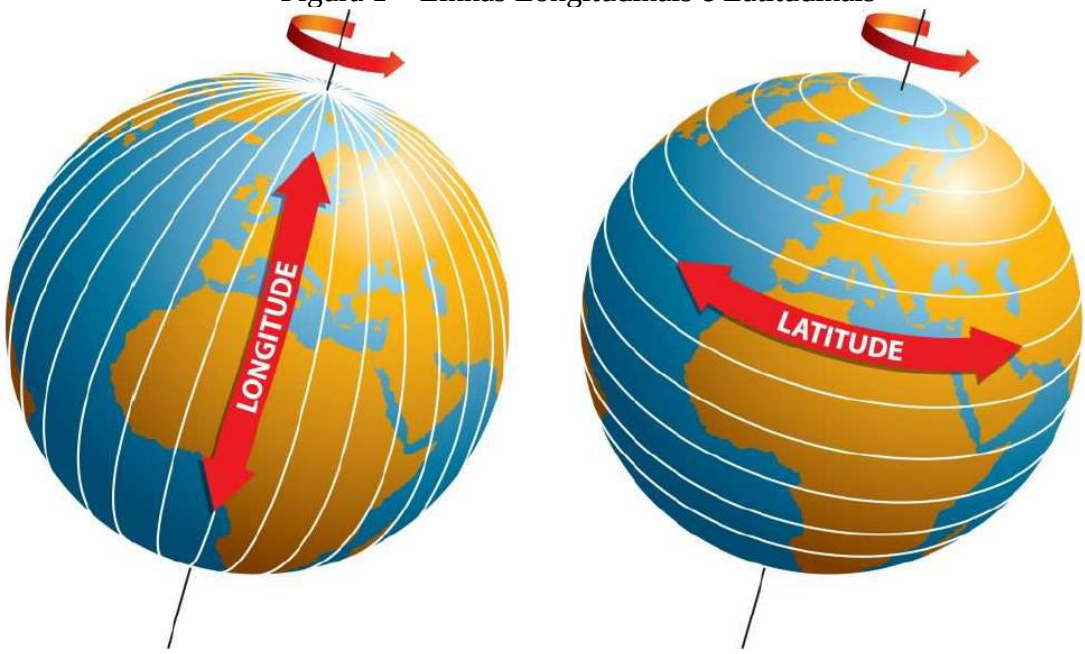
Neste capítulo será apresentada a fundamentação teórica utilizada para a concepção deste trabalho. Tendo como finalidade a orientação e informação a respeito dos conceitos utilizados no desenvolvimento do aplicativo Android, o qual compõe o objetivo central do trabalho.

2.1 Coordenadas Geográficas

O sistema de coordenadas geográficas foi desenvolvido com o intuito de identificar cada ponto existente na superfície terrestre utilizando de linhas imaginárias, paralelos e meridianos, que se interceptam no globo terrestre possibilitando dizer com certa exatidão a localização de um ponto em qualquer lugar do planeta Terra. (IBGE, 2021)

As linhas que são chamadas de paralelos, cortam o globo terrestre no sentido Leste-Oeste. As linhas de meridiano cortam o globo terrestre no sentido Norte-Sul, os meridianos ligam os dois polos. Respectivamente paralelos e meridianos são responsáveis pelas latitudes e a longitudes, como apresentado na Figura 1 (IBGE, 2021).

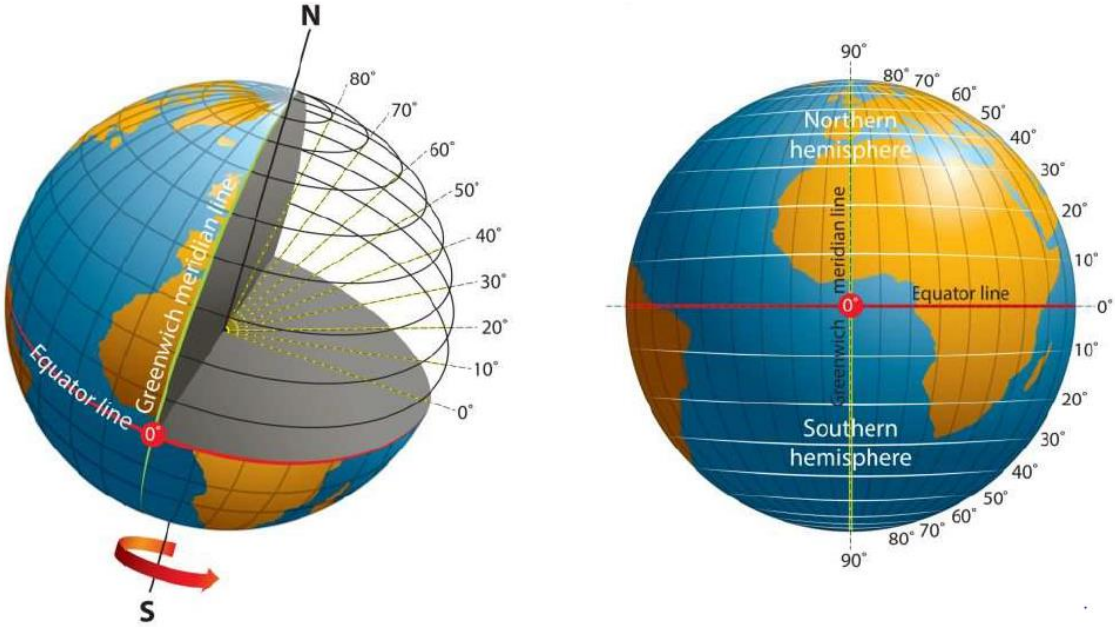
Figura 1 – Linhas Longitudinais e Latitudinais



Fonte: (Gomes, 2017)

A latitude representa a distância latitudinal, dada em graus, de qualquer ponto do globo terrestre em relação a linha do equador. A distância latitudinal e representada de modo que se inicia na linha do equador com 0° indo no máximo até 90° para o Norte e 90° para o Sul, onde Norte e Sul representam respetivamente o Polo Norte ou Ártico e Polo Sul ou Antártico, como apresentado na Figura 2 (GOMES,2017).

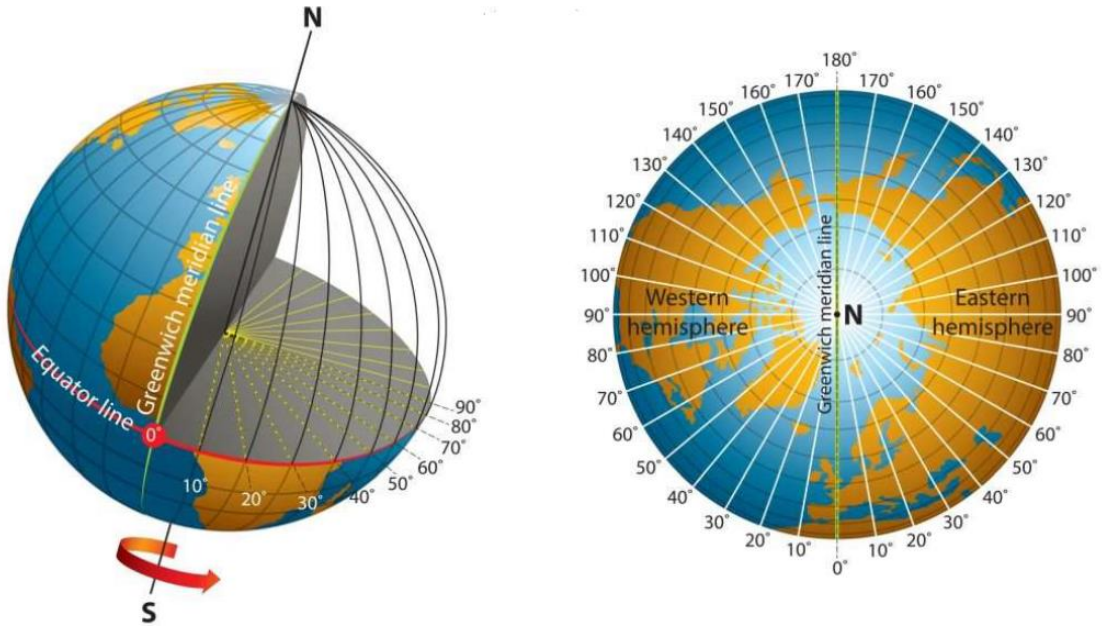
Figura 2 – Linhas Latitudinais



Fonte: (Gomes, 2017)

De forma semelhante à latitude, a longitude também representa a distância em graus de qualquer ponto do globo terrestre, porém de forma longitudinal dividindo o planeta verticalmente em relação ao Meridiano de *Greenwich*, marco referencial, onde se dá o hemisfério leste e oeste. A distância longitudinal é representada de modo que se inicia no Meridiano de *Greenwich* com 0° indo no máximo até 180° para Leste ou oriental (em inglês, *Eastern*) e 180° para oeste ou ocidental (em inglês, *Western*), como apresentado na Figura 3 (GOMES, 2017).

Figura 3 – Linhas Longitudinais



Fonte: Gomes, 2017

2.2 Sistema De Posicionamento Global (GPS)

Desde os primórdios o homem tem utilizado de novas tecnologias para se guiar de forma segura, inicialmente esse fato se dava por meio da busca de referência nos astros e estrelas. Posteriormente no século XV e XVI com o auge das grandes navegações, iniciou-se o processo de criação e desenvolvimento de sistemas de localização, com intuito de transportar carga e pessoas por longas distâncias intercontinentais. Com o passar dos tempos o sistema de localização evoluiu-se, culminando no ano de 1970 quando os militares americanos desenvolveram o *Global Positioning System* (GPS) (DUARTE E COELHO, 2019).

Inicialmente o sistema GPS era de uso exclusivo dos militares americanos. Posteriormente foi liberado para uso civil, o sinal era distorcido causando baixíssima precisão. Tal distorção só fora retirada no ano 2000, fazendo com que qualquer pessoa com um receptor próprio, pudesse em tempo real, ter acesso a sua longitude, latitude e altitude (PEREIRA, 2014).

O *Global Positioning System* se faz através de vinte e quatro satélites próprios para este sistema, que estão a ao redor do planeta Terra, à uma altitude aproximada de 20200 km acima do nível do mar. Os satélites são dispostos de maneira que, em qualquer lugar do mundo e a

qualquer momento, exista pelo menos quatro satélites que possam, em qualquer ponto do planeta, oferecer informação sobre a localização precisa (NETO, 2013).

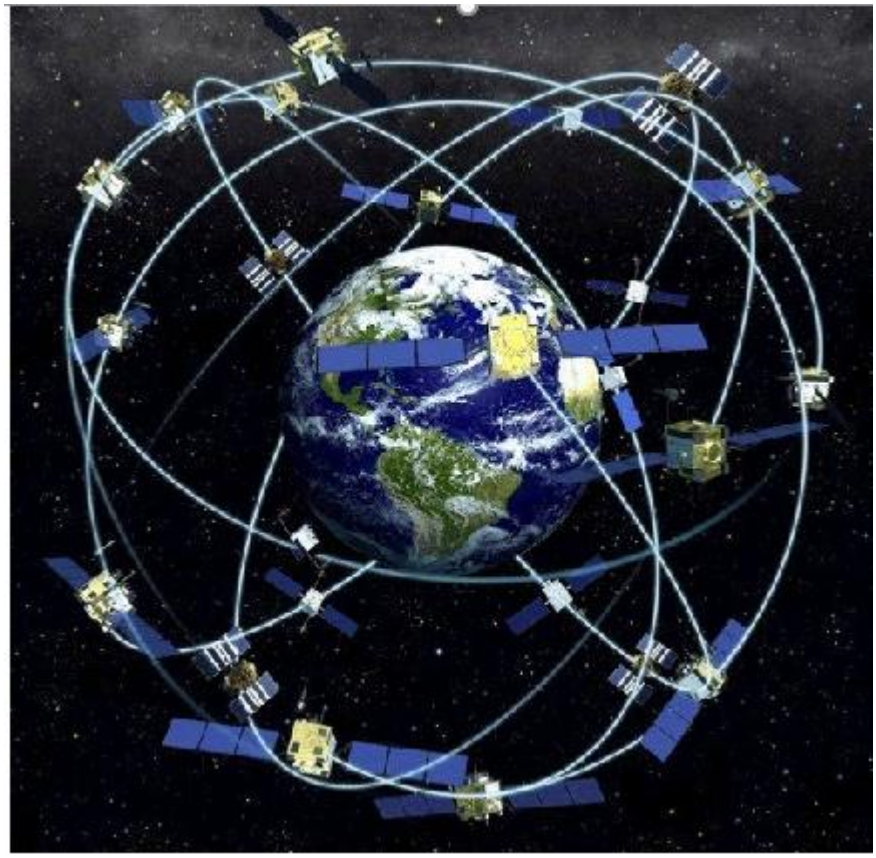
A localização se faz quando a informação de longitude, latitude e altitude, enviada pelo GPS, é aplicada a um mapa global preparado. Atualmente o GPS juntamente com sistema de localização, e usado em larga escala nos aparelhos *smartphones*, o que permite a criação de diversas aplicações envolvendo tal dispositivo (NETO, 2013).

2.2.1 GPS e seus segmentos

O sistema GPS se faz a partir de três segmentos onde o segmento espacial é composto pelos satélites artificiais que orbitam o planeta Terra, o segmento de controle são as estações terrestres que controlam os satélites e o segmento de usuário que recebe o sinal enviado pelo satélite determinando sua posição (PAZ E CUGNASCA, 1997).

Como apresentado na Figura 4 o segmento espacial é um conjunto de vinte e quatro satélites, equipados com um relógio de altíssima precisão, distribuídos em seis órbitas circulares a uma altitude aproximada de 20200 Km do planeta Terra. As órbitas se encontram a 55° de inclinação em relação a linha do Equador, garantindo que no mínimo quatro satélites estejam sendo visíveis em qualquer lugar do planeta (DUARTE E COELHO, 2019).

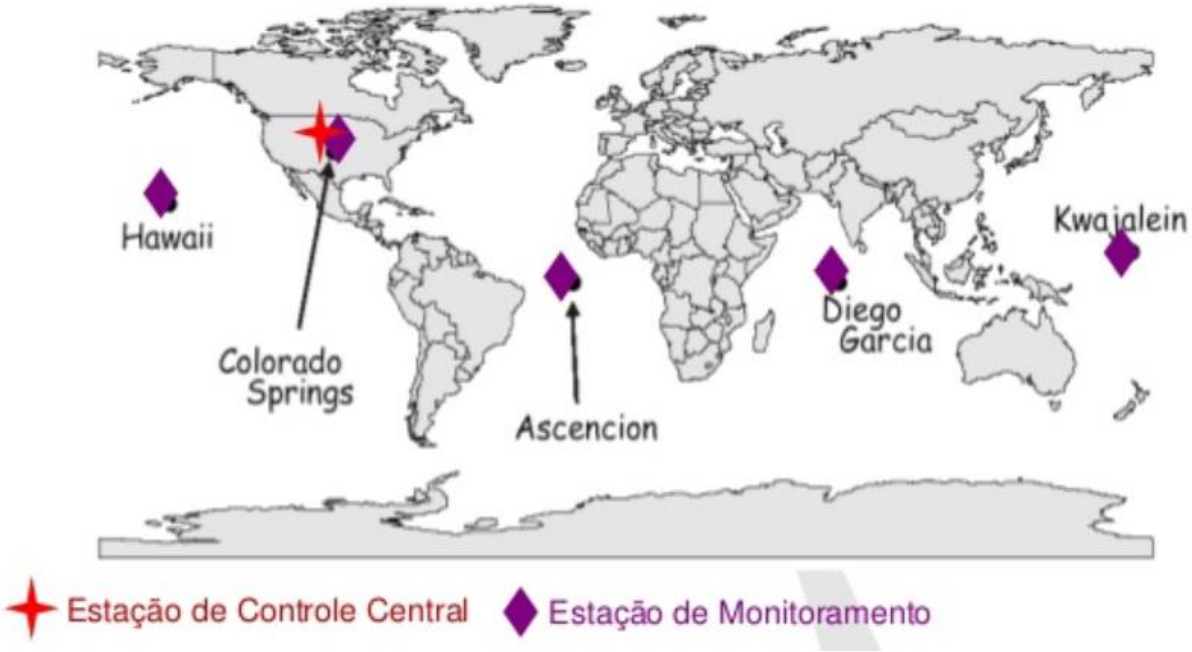
Figura 4 – Distribuição das 6 órbitas com os 24 satélites



Fonte: RIBEIRO, 2008

O segmento de controle é composto por cinco estações, distribuídas ao longo da superfície terrestre, com o intuito de monitorar os satélites efetuando correções em suas órbitas e em seus relógios. Como mostra a Figura 5 as estações são: Colorado Springs, no oeste dos Estados Unidos; *Hawaii*, no Oceano Pacífico nos Estados Unidos; *Kwajalein*, no Oceano Pacífico nas ilhas das Carolinas no norte dos Estados Unidos; ilha de *Ascencion*, de posse britânica no Atlântico Sul; e ilha de Diogo Garcia, posse britânica no Oceano Índico. A estação mestre onde abriga a central de operações é a estação de *Colorado Springs* (PAZ E CUGNASCA, 1997).

Figura 5 – Estações de Controle do GPS.



Fonte: RAILANO, 2014

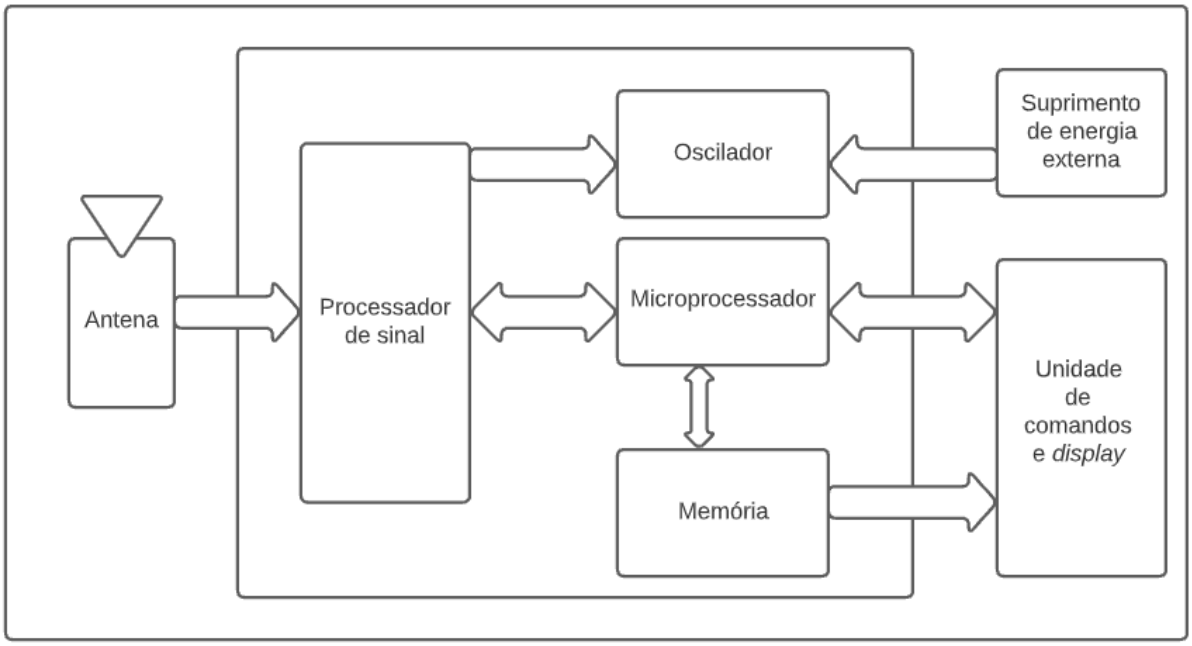
Por fim, o segmento de usuário consiste em receptores espalhados pelo mundo, como os exemplos de receptores apresentado na Figura 6. Sua função é de captar os sinais de satélites que estejam sendo visíveis pelo receptor em questão e desta forma, convertê-los em informações de posicionamento que são latitude, longitude e altitude (NETO,2013).

Figura 6 – Receptores de Usuário do GPS

Fonte: NETO, 2013

Os receptores GPS são constituídos de vários componentes, dentre eles pode-se citar alguns representados por blocos na Figura 7, que serão apresentados ao longo deste tópico.

Figura 7 – Diagrama de blocos do receptor GPS



Fonte: Elaborada pelo autor

A “Antena” é responsável por detectar as ondas eletromagnéticas emitidas pelos satélites, convertendo-as em sinais elétricos enviadas para a parte eletrônica do receptor. As antenas devem dispor de boa sensibilidade para que possam captar sinal em qualquer tipo de terreno ou elevações (MONICO, 2000).

“Processamento de Sinal” é um bloco utilizado para a seleção de rádio frequência (RF), onde o sinal e selecionado com ajuda dos “Osciladores” em uma frequência intermediaria (FI), facilitando a interpretação destes sinais pelos componentes eletrônicos do circuito (MONICO, 2000).

“Osciladores” geralmente fabricados a partir de componentes de cristal de quartzo, tem a responsabilidade de produzir a frequência de batimentos da onda portadora que irá demodular o sinal com ajuda do estágio de FI (MONICO, 2000).

“Microprocessador” é responsável por realizar as operações dos dados digitais obtidos, isso inclui obter e processar o sinal, codificar as informações, controlar o fluxo de dados, calcular posição, calcular velocidade do receptor entre outras funções (MONICO, 2000).

A “Unidade de Comando e *Display*” é a interface com o usuário, essa é integrada por meio de um *display* e por comandos inseridos por meio de teclas possibilitando selecionar e

visualizar diversas informações como coleta de dados, mostrar coordenadas, monitoramento do usuário dentre outros mais (MONICO, 2000).

As “Memórias” são utilizadas com intuito de armazenar dados das informações coletadas dos satélites como posição, distância, velocidade dentre outros que podem ser usados em alguma aplicação futura (BERNARDI E LANDIM, 2002).

O “Suprimento de energia” é essencial em relação a aplicabilidade do GPS pois seu uso, na maior parte das vezes, se faz em locais onde não existem fontes de energia externa o que na primeira geração de GPS foi considerado um fator crítico, devido ao consumo excessivo de energia, comparados com dispositivos GPS dos dias atuais (BERNARDI E LANDIM, 2002).

Os canais são a forma de se enviar dados no presente caso de informações de localização para o receptor, podendo ser de três tipos:

- Multicanais: cada canal e responsável por apenas um satélite visível naquele ponto geográfico, sendo necessária no mínimo quatro canais para se obter a posição (BERNARDI E LANDIM, 2002).
- Sequenciais: os canais são alternados em um intervalo de tempo entre os satélites visíveis naquela localização, utilizando vários canais até que a mensagem do satélite seja entregue (BERNARDI E LANDIM, 2002).
- Multiplexados: são efetuadas comutações muito rápidas em sequência, entre os canais, permitindo que a mensagem chegue quase simultaneamente ao receptor (BERNARDI E LANDIM, 2002).

Assim funcionavam os equipamentos para GPS exclusivamente, atualmente todas estas funcionalidades são obtidas em um circuito integrado (CI) embutido dentro dos *smartphones* atuais.

2.2.2 Funcionamento do GPS

O Sistema de Posicionamento Global consiste em vinte e quatro satélites programados para transmitir, via sinal de rádio aos receptores terrestres um padrão fixado que funciona com um cronômetro muito preciso, necessário para o cálculo da posição em que se encontra o receptor (NETO, 2013).

O receptor recebe o padrão de sinal do satélite para calcular, na ordem de um décimo de segundo, a diferença entre o tempo que foi recebido e o tempo em que foi emitido o sinal,

permitindo assim calcular a distância e o tempo entre o satélite e esse receptor, deste modo e possível localizar uma pessoa em um determinado local do espaço terrestre constituído de um raio, ou seja, um círculo em que o receptor esteja contido, e uma distância calculada, entre a pessoa e o satélite (NETO, 2013).

Deste modo obtém-se a distância entre uma pessoa e o satélite o que determinar a posição relativa entre ambos. No entanto para obter uma melhor precisão e acurácia dos dados são necessários interseções do receptor GPS com pelo menos três satélites simultaneamente (SANTANA et al, 2019).

2.2.3 Geolocalização em *smartphones*

O avanço da tecnologia possibilitou a integração do sistema GPS nos dispositivos *mobile*, fazendo com que diversos tipos de aplicativos móveis com acesso a localizações geográficas, pudessem ser criados. Este avanço permitiu que qualquer pessoa munida de um aparelho *smartphone* pudesse, em tempo real, ter acesso à localização de qualquer ponto no globo terrestre, imitando os receptores GPS convencionais. Dentre os diversos sistemas criados o mais comum é utilizado na navegação de trânsito, como ferramenta de *geomarketing* através dos aplicativos *Waze, Uber, iFood* dentre outros (SANTANA et al, 2019).

No entanto, existe atualmente diversos meios que pode fornecer a um dispositivo móvel informações sobre a localização geográfica além do sistema GPS convencional, pode-se citar também; o sistema *Assisted Global Positioning System* (AGPS) que é um aprimoramento do sistema GPS convencional onde as informações dos satélites são recebidas através de uma antena de telefonia celular; o sistema *Global System for Mobile Communications* (Sistema Global para comunicações Móveis, GSM) onde as informações de localização são obtidas por meio de ondas de rádios fornecidas pelas operadoras de telefonias moveis, necessitando da cobertura de sinal da operadora no aparelho para funcionar; e também através do *Wireless Fidelity* (Wi-Fi) onde também é possível se obter a localização através da internet (ALVES, 2018).

Nos aparelhos *smartphones* o sistema de localização empregado possui três modos de operação; o modo de Alta Precisão que mescla as informações provenientes do sistema GSM, com informações do sistema AGPS, para determinar a localização do usuário; já o modo de Economia de Bateria, para determinar a localização, utiliza apenas informações provenientes do sistema GSM; e por fim o modo Somente no Dispositivo que usa somente informações do receptor AGPS para obter a localização (ALVES, 2018).

A escolha do modo de operação e efetuada através do sistema operacional do aparelho móvel em questão, que escolhe o modo de operação com base na disponibilidade da rede de telefonia ou de acesso à internet, na requisição da aplicação utilizada no momento, no estado (ligado/desligado) do receptor AGPS entre outros (SANTANA et al, 2019).

2.3 Geofencing

Embora o *Geofencing* tenha ganhado mais espaço atualmente, seus princípios vêm sendo utilizados desde a criação da telefonia móvel. Para obter a localização de um usuário pegava como base a localização da torre de telefonia móvel onde ele se encontrava. Tal informações, de posse das operadoras, podem ser utilizadas como provas, onde o usuário poderia ser acusado ou inocentado do fato com base em sua localização, até como estratégias de *marketing* vendidas às empresas pelas operadoras, com intuito de disparar mensagens de texto SMS com ofertas para os usuários. Porém a precisão desta localização era restrita a área de cobertura da torre (STATLER,2016).

Nos dias atuais, esta tecnologia está sendo oferecida pelos aplicativos (APP) empregados nos novos aparelhos *smartphone*, que através do sistema GPS entrega uma localização muito mais rápida, fácil e precisa, possibilitando que inúmeros APPs, com diversas finalidades, possam ser criados e utilizados de maneira gratuita pelos usuários (WHITE,2017).

O *geofencing* atual trabalha de modo que um perímetro virtual, geoposicionado possa ser delimitado por meio de coordenadas geográficas emitidas pelo sistema GPS e transmitida ao aplicativo no *smartphone* do usuário de várias formas, já citadas anteriormente neste capítulo. O propósito é que seja tomada alguma ação específica, tendo como base o perímetro virtual (*geofencing*) estipulado e a função que o aplicativo deve exercer, dentre elas pode-se citar o disparo de mensagens de *marketing*, mensagem de alerta de segurança, uma tomada de decisão específica dentro do aplicativo dentre várias outras funções (WHITE,2017).

É importante salientar que o sistema de *geofencing* atualmente não se restringe apenas em APP *mobile*, sendo utilizado também em inúmeros outros dispositivos como *drones*, equipamentos de navegação, equipamentos de segurança entre muitos outros. Mas como o intuito deste trabalho é voltado ao desenvolvimento *mobile* (STATLER,2016).

Para se utilizar o sistema *geofencing* o aplicativo deve respeitar a regra dos três Ps do dispositivo móvel:

- Permissão: é necessário que o usuário baixe o aplicativo e posteriormente conceda as permissões de acesso ao serviço de localização do dispositivo ao aplicativo (STATLER,2016).
- Privacidade: não é permitido que o usuário seja rastreado em segundo plano fora da *geofencing*. O aplicativo só deverá operar mesmo que em segundo plano quando usuário estiver dentro da *geofencing* (STATLER,2016).
- Preferência: o usuário deve ter total controle das permissões concedidas ao sistema, podendo revogá-las a qualquer momento, além da opção de excluir totalmente a aplicação do sistema (STATLER,2016).

No presente trabalho o sistema *geofencing* será utilizado para delimitar as áreas de produção dentro da propriedade rural para que apenas com a aproximação do usuário o aplicativo seja capaz de identificar a área e retornar as informações alocas sobre aquele local.

2.4 Sistema Operacional *Android*

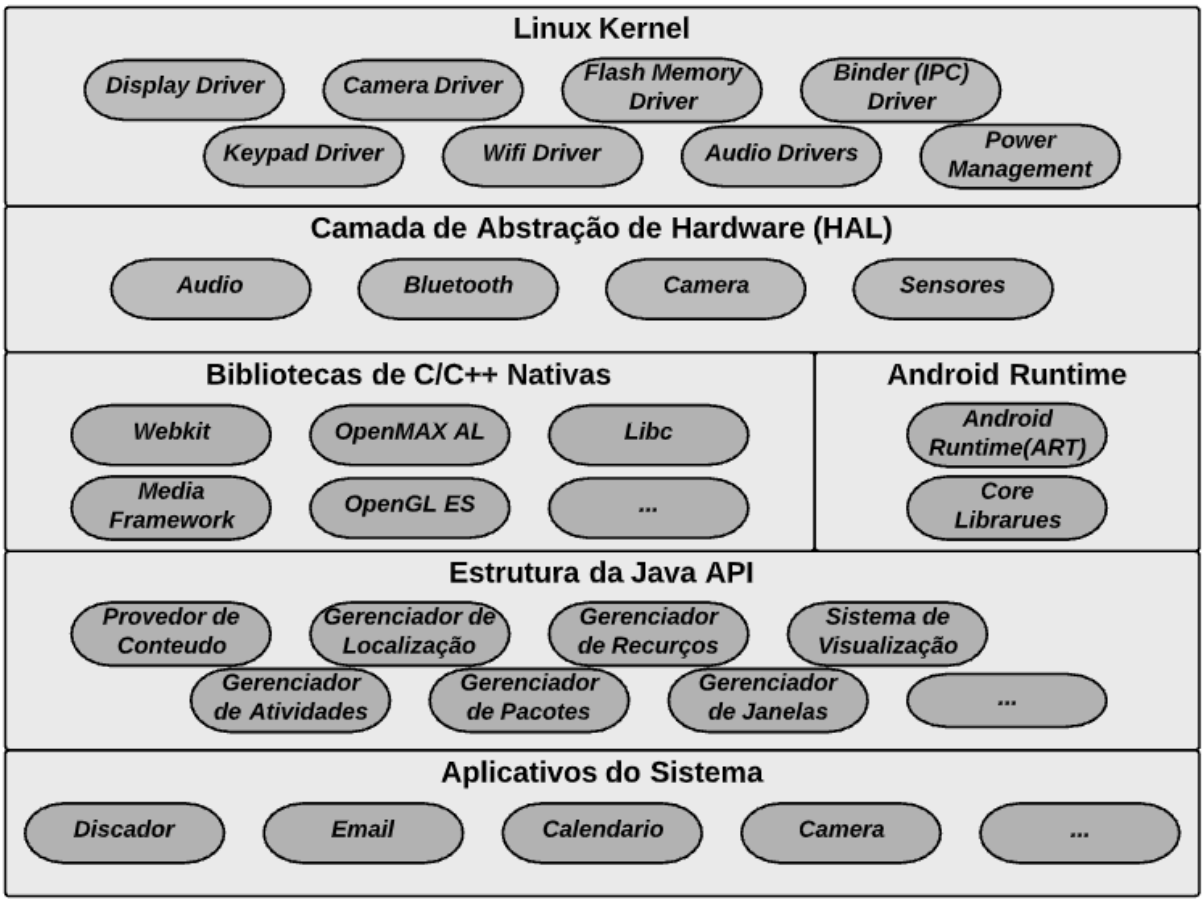
Os Sistemas Operacionais são primordiais para o funcionamento do aplicativo pois através dele o usuário consegue interagir com a máquina de forma a controlar as funções como teclado, mouse e os demais dispositivos além de garantir a execução dos programas (LIMA, 2021).

Da mesma forma que nos dispositivos computacionais os *smartphones* também requerem um sistema operacional para seu devido funcionamento. Atualmente são três os principais sistemas: o *IOS, Android* e o *Windows Phone*, sendo o mais utilizado e por isso o que será utilizado na concepção deste trabalho o sistema operacional *Android* (LIMA, 2021).

Esta plataforma teve início em 2003 baseada no sistema operacional *Linux*. Sendo melhorada com o passar do tempo trazendo cada vez mais funcionalidades, em 2005 foi comprada pela *Google* seguindo a ideologia de um sistema totalmente gratuito. Essa ideologia proporcionou que diversas companhias de *smartphones* que ainda não tinham um sistema operacional, optassem por utilizar o Android, fazendo dele o sistema operacional para *smartphones* mais utilizado do mundo (FAUSTINO, 2017).

Sobre a arquitetura do sistema operacional *android* pode-se analisar na Figura 8 os seguintes componentes da plataforma.

Figura 8 – Arquitetura do Sistema operacional Android.



Fonte: Elaborada pelo autor

O *Kernel* do *Linux*, de onde é fundada a plataforma *android*, é responsável por cobrir algumas funcionalidades e por gerenciar algumas ações como o encadeamento de memória de baixo nível (DEVELOPERS,2020 A).

A Camada de Abstração de *Hardware* tem a função de produzir interfaces através de módulos e bibliotecas, destinado a expor a capacidade de *hardware* do dispositivo como o modulo de gps, câmera, sensores, microfone, alto falantes dentre outros (DEVELOPER,2020).

O *Android Runtime* (ART) trabalha de modo que seja possível a execução de várias máquinas virtuais nos sistemas *Androids*, o qual tem como característica a baixa disponibilidade de memória. O ART opera executando arquivos no formato de *bytecode* chamado DEX, que tem como característica a otimização para um consumo mínimo de memória (DEVELOPERS,2020 A).

Bibliotecas C/C++ Nativas são bibliotecas nativas desenvolvidas em C e C++ com o propósito de oferecer recursos às demais camadas. Para desenvolvedores que desejam ter acesso

a estas bibliotecas nativas recomenda-se o uso do *Native Development Kit* (NDK) (DEVELOPERS,2020).

Estrutura da Java API é o conjunto de todos os recursos, chamados de *Application Programming Interface* (API), usado pelo sistema operacional *Android* na criação dos aplicativos, de modo que seja possível a reutilização de componentes e serviços específicos de cada módulos do sistema (DEVELOPERS,2020).

Aplicativos do Sistema são aplicativos principais nativos do *Android*, como o telefone, e-mail, calendário, navegador de internet e muitos outros. Estes aplicativos vêm junto com o sistema, mas alguns podem ser substituídos por aplicativos de terceiros, como o navegador, já outros não podem ser substituídos, como a Configuração de Sistema. Estes desenvolvem funcionalidades principais que podem ser acessados por outros aplicativos, por exemplo se um aplicativo deseja enviar um e-mail, então é possível invocar o aplicativo de e-mail para esta função, abrindo mão de reimplantar a funcionalidade no aplicativo atual (DEVELOPERS,2020).

2.5 *Application Programming Interface* (API)

Em português “Interface de Programação de Aplicações”, permite que aplicações possam se comunicar sem que haja acesso à implementação uma da outra, auxiliando na integração de componentes da arquitetura preexistente promovendo economia de tempo e dinheiro (REDHAT, 2021).

Normalmente existe uma regra para como as partes irão se comunicar, ou seja, se um dos sistemas envia uma requisição remota a partir de uma estrutura específica, esta, irá ditar a forma com que a outra parte deverá responder (REDHAT, 2021).

Com intuito de exemplificar a aplicação da API, suponha desenvolver um sistema que toma como base a localização instantânea do usuário a fim de verificar se ele se encontra em uma localização pré-determinada. O quão mais simples seria, se em vez de implementar todo o processo de obtenção da localização, através de programação nativa, fosse possível obtê-lo apenas com um comando buscando essa localização em sistema de terceiros como o *Google Maps*. Portanto esse tipo de troca de informação só é possível devido à API implementada pela *Google Maps* que permite essa troca de informação (REDHAT, 2021).

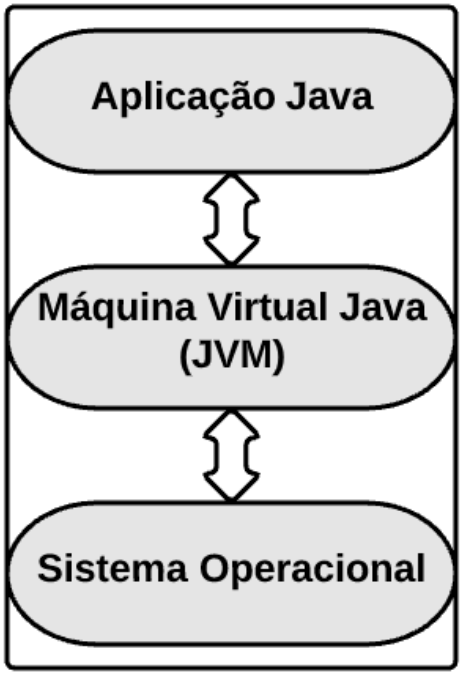
2.6 Linguagem Java

Sun Microsystems, posteriormente comprada pela *Oracle* em 2009, criou uma equipe com o propósito de desenvolver inovações tecnológicas. A equipe criou o Java, uma linguagem de programação multiplataforma que utiliza em sua estrutura um interpretador, proporcionando que aplicações rodem em todas as plataformas suportadas pela linguagem, sem a necessidade de reescrita de código (CAELUM, 2016).

Assim como em todas as linguagens de programação o Java também possui sintaxe e estrutura própria. A estrutura do Java foi baseada na linguagem de programação C, fazendo com que suas sintaxes sejam semelhantes, e o paradigma de programação também, com orientação à objeto (CAELUM, 2016).

As linguagens convencionais utilizam compilador nativo do sistema operacional, enquanto o Java converte o código em *bytecode* que posteriormente será executado por uma máquina virtual Java (JVM). Deste modo, como na Figura 9, todas as funções nativas da máquina virtual Java, possibilitam que os códigos em Java rodem em qualquer sistema operacional, desde que tenha suporte para a máquina virtual Java (SOUSA, 2017).

Figura 9 – Montagem operacional Java



Fonte: Elaborada pelo autor

Além da possibilidade de escrever código uma única vez para rodar em diversos sistemas operacionais, o Java também tem várias outras vantagens como permitir desalocar automaticamente espaço da memória não utilizada, possui construtores que permite uma alta produtividade superando até mesmo o C/C++, é uma linguagem gratuita e de código aberto, possui empregabilidade em diversas áreas indo desde telefones celulares até servidores possibilitando ao desenvolvedor um vasto leque de sistemas a serem desenvolvidos (SOUSA, 2017).

2.7 Linguagem *Kotlin*

A linguagem de programação *Kotlin* foi criada em 2010 pela empresa *JetBrains*, com o intuito de proporcionar uma melhor experiência de programação aos desenvolvedores *Android*. Seu nome foi dado em homenagem a ilha *Kotlin* em São Petersburgo, região onde a linguagem foi criada (OLIVEIRA, 2019).

O lançamento oficial da primeira versão da linguagem foi em fevereiro de 2016, licenciado pela Apache 2.0, totalmente *Open Source*, seu código fonte encontra-se disponível no *GitHub*. Em março de 2017 a *Google* anunciou oficialmente o suporte ao *Kotlin* nas plataformas *Android*. Apenas 2 anos após seu lançamento em 2018, a linguagem apareceu no *ranking* das mais usadas, sendo adotadas por grandes empresas como *Evernote*, *Uber*, *Pinterest* dentre outras, que já utilizavam o *Kotlin* não apenas para aplicativos *Android*, mas também como ferramentas internas, aplicativos *web*, *desktop* e muito mais (OLIVEIRA, 2019).

A linguagem *Kotlin* possui dois paradigmas de programação, orientada a objeto (OO) e programação funcional (FP), que podem ser utilizados juntos ou separados. Assim como o Java o *Kotlin* também compila *bytecode* que pode ser interpretado pela máquina virtual Java (JVM). Deste modo a linguagem é 100% compatível com a linguagem de programação Java, possibilitando total interação de sua base de código, juntamente com todas as suas bibliotecas e estruturas. Com isso é possível utilizar facilmente código Java em ambiente *Kotlin*. Devido a sua estrutura mais concisa, estima-se que haja uma redução de 40% no número de linhas de código escritas em *Kotlin* em comparação com o Java (KOTLIN, 2021).

O *Kotlin* é a linguagem proposta para a construção do aplicativo, mais detalhes são apresentados no próximo capítulo.

2.8 Ambiente de Desenvolvimento Integrado (IDE)

O *Android Studio* é um *software* destinado a auxiliar desenvolvedores na criação de aplicativos para sistemas operacionais *Android*. Esta categoria de *software* é conhecida como ambiente de desenvolvimento integrado (IDE) (DEVELOPERS, 2021).

Além do *Android Studio*, com o propósito de desenvolver *softwares* para *Android*, existem diversas outras IDEs com vários outros propósitos como: *Visual Studio* dedicado ao .net nas linguagens *Visual Basic*, C, C++, C#.; Eclipse uma IDE multiplataforma proporciona a inserção de *plugins* que otimiza o desempenho, além de poder programar em diversas linguagens como C, C++, *Python*, *Java*, *Perl*, *PHP* e muitas outras; *NotePad++* destinada a trabalhos mais simples geralmente voltado para desenvolvimento *Web* (COSTA, 2018).

2.9 Banco de Dados

Um banco de dados é um conjunto de dados agrupados de modo a trazer informação sobre um determinado assunto. Existe uma grande diferença entre dados e informações, o dado sozinho não garante uma informação, pois depende que outro intérprete sua função, por exemplo o número 150, sozinho não traz nenhuma informação, mas ao relacionar este dado com ‘quantidade de cabeças de gado no rebanho: 150’ observar que agora tem-se uma informação. Para uma boa interpretação de um dado é necessário o relacionamento com meta dados, que são informações pertinentes àquele dado (SILVA, 2015).

Nos dias atuais existem Sistemas Gerenciadores de Banco de Dados (SGBD) que são ferramentas tecnológicas que auxiliam no registro e gerenciamento dos dados, relacionando-os entre si e com seus respectivos metas dados. Deste modo o sistema facilita a manipulação dos dados, auxiliando no processo de buscar informações mais rapidamente, de forma precisa e segura (SILVA, 2015).

Dentre as diversas vantagens em se utilizar um SGBD pode-se citar:

- Aumento da Produtividade: garante um aumento de produtividade devido a facilidade e à rapidez em acessar os dados (ROVEDA, 2021).
- Aumento de Segurança: reduz o risco de perda de informação por má organização dos dados (ROVEDA, 2021).
- Melhor Relacionamento: tem um melhor controle sobe a demanda de várias requisições simultâneas aos dados (ROVEDA, 2021).

- Melhora o Planejamento de Decisões: o SGBD permite uma organização dos dados de forma que facilita a visualização de informações importantes para tomadas de decisões na aplicação (ROVEDA, 2021).
- Reduz o Risco: os sistemas SGBD carrega um sistemas de segurança que restringem o acesso aos dados de pessoas não autorizas, reduzindo os riscos de invasão (ROVEDA, 2021).

Atualmente os sistemas SGBD se dividem em dois diferentes tipos, o primeiro é o sistema de gerenciamento do banco de dados relacional, conhecido também como banco de dados *Structured Query Language* (SQL), o segundo é o SGBD não relacional conhecido também como *Not Only Structured Query Language* (NoSQL) (ROVEDA, 2021)..

A fim de suprir a necessidade em armazenar dados relacionando estruturas do mundo real, foi criado o banco de dados relacional, que armazena as informações em forma de tabelas (relações), separando os dados distintos em tabelas distintas. Sua principal característica se dá ao fato de garantir a consistência dos dados por meio de restrições de integridade. Além de integrar o processo de normalização constituídos em regras específicas para construção adequada das tabelas. As associações entre tabelas são feitas através de chaves de identificação, proporcionando uma maior consistência no armazenamento dos dados, maior eficiência em acessar esses dados, menor redundância e diminuindo a inconsistência dos dados. A linguagem adotada para manipulação de dados em banco de dados relacional e chamada *Structured Query Language* (SQL) (MARILIA, 2012).

Com o propósito de suprir demandas como escalabilidade, performance e disponibilidade, não suportada pelo banco de dados relacional, foi criado o banco de dados não relacional. Tendo como base o modelo relacional com a característica de eliminar certas regras estruturadas, o modelo não relacional propicia ganho de performance, gerando alto armazenamento e grande disponibilidade dos dados. Como sua construção, sua linguagem de manipulação dos dados também deriva do modelo relacional, passando a se chamar *Not Only Structured Query Language* (NoSQL) (MARILIA, 2012)..

Entre as diversas características do banco de dados não relacional pode-se citar suporte a replicações, simplicidade no acesso ao banco de dados, ordenação de ocorrência de eventos, escalabilidade horizontal, suporte a transações paralelas, manipulação de grandes volumes de dados ao longo dos nós de uma rede de computadores, ausência de esquemas que proporciona alta escalabilidade e consistência eventual dentre outras (MARILIA, 2012).

3 PROPOSTA DE SOLUÇÃO

Este capítulo apresenta a proposta de solução para a implementação a ser realizada no TCC2 (Trabalho de Conclusão de Curso para o segundo semestre). De forma a expor as ferramentas de *hardware* e *software*, além de conceito sugestionados em capítulos anteriores para a construção do aplicativo que se pretende implementar.

3.1 *Kotlin*

As vantagens que levaram a adoção da linguagem *Kotlin* para o desenvolvimento do aplicativo foram:

- Menor gasto de tempo com escrita de código e com interpretação de código de terceiros (KOTLIN, 2021).
- Uma linguagem madura, utilizada por diversas empresas que desenvolve para *Android* (KOTLIN, 2021).
- Interopera com o Java, sem a necessidade de se migrar o código para *Kotlin*.
- Suporta multiplataforma, possibilitando compartilhar código também em *IOS*, *back-end* e *Web* (KOTLIN, 2021).
- Possui código mais seguro devido a maior legibilidade acarretando menos erros (KOTLIN, 2021).
- Facilidade em aprender a linguagem, especialmente para quem já desenvolve em Java (KOTLIN, 2021).

Para criar aplicações para *Android* na linguagem *Kotlin* recomenda-se a utilização do ambiente de desenvolvimento (IDE) *Android Studio*, criada pela *JetBrains*, a mesma empresa responsável pela criação do *Kotlin*. Recomenda-se também o *Software Development Kit* (SDK) que contêm todas as ferramentas e emuladores necessário para o desenvolvimento *Android* (LECHETA, 2017).

Diante de todas os pontos apresentados, em relação ao desenvolvimento de aplicações *Android* por meio da linguagem de programação *Kotlin*, será adotada como a linguagem do código fonte da aplicação proposta a ser desenvolvida pelo presente trabalho.

3.2 *Android* Studio

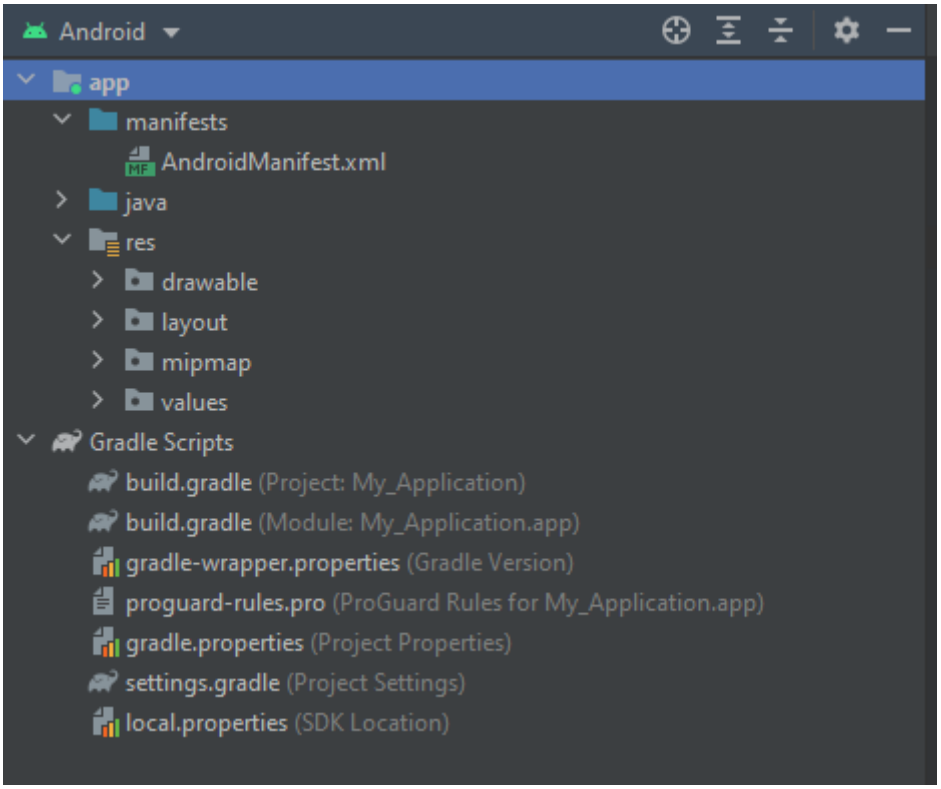
Por ser o ambiente de desenvolvimento integrado oficial para desenvolver aplicativos em dispositivos *Android*, será utilizado neste trabalho o *Android Studio*, que oferece recursos como:

- Sistema de compilação flexível (DEVELOPERS, 2021 A).
- Emulador rápido com vários recursos (DEVELOPERS, 2021 A).
- Possui um ambiente unificado proporcionando desenvolver para todos os dispositivos *Android* (DEVELOPERS, 2021 A).
- Ferramentas que proporcionam mudanças no código fonte do sistema em execução sem a necessidade de reinicialização do aplicativo (DEVELOPERS, 2021 A).
- Possui total integração com o *GitHub* auxiliando a importar e exportar códigos (DEVELOPERS, 2021 A).
- Diversas opções de ferramentas e *Frameworks* (DEVELOPERS, 2021 A).
- Ferramentas que auxiliam na detecção de problemas como desempenho, usabilidade, compatibilidade entre outros (DEVELOPERS, 2021 A).
- Possui suporte à linguagem *Kotlin* (DEVELOPERS, 2021 A).

Estes foram apenas alguns dentre tantos outros motivos que levaram a escolha do *Android Studio* como o ambiente de desenvolvimento integrado na concepção do aplicativo para o presente trabalho.

Ao criar um projeto, no *Android Studio* cria-se uma estrutura de pastas em que os arquivos da aplicação são organizados por meio de módulos. Os módulos permitem que os arquivos sejam agrupados de maneiras distintas de acordo com cada funcionalidade, de modo à facilitar o acesso aos arquivos do projeto. A Figura 10 mostra um exemplo de estrutura de pasta de um novo projeto do *Android Studio* (Peixoto, 2016).

Figura 10 – Estrutura de Pasta *Android Studio*



Fonte: Elaborada pelo autor

- Pasta *app* se trata de um módulo principal, criada automaticamente pelo *Android Studio*, outros módulos podem ser criados dependendo da necessidade da aplicação em questão. Esta estrutura contém códigos fontes e arquivos referentes especificamente a este modulo como (Peixoto, 2016):
    - Pasta *manifests* contém o arquivo *AndroidManifest.xml* responsável por declarar todos os componentes de configuração do modulo, como permissões do usuário concedidas ao sistema, tipo de exibição dos ícones, modo de reprodução de tela entre outros (Developers, 2016).
    - Pasta Java que apesar do sistema ser desenvolvido em *Kotlin*, existe alguns arquivos que são dependentes da linguagem de programação Java. A pasta Java contém os códigos fontes responsáveis pelas classes, arquivos responsáveis pelos testes automatizados e arquivos responsáveis pelos testes unitários do sistema (Developers, 2021).
    - Pasta *res* contêm todos os recursos utilizados no sistema como imagens, textos dentre outros (Peixoto, 2016):

- Pasta *drawable* e responsável por armazenar todos os ícones da aplicação (Peixoto, 2016).
- Pasta *layout* e responsável por armazenar os arquivos xml dos *layouts* da aplicação, sendo que cada *layout* contido na aplicação deve possuir um arquivo xml que o representa (Peixoto, 2016).
- Pasta *mipmap* responsável por armazenar imagens, como os ícones de *launcher*, para diferentes configurações de tela, sendo que, a escolha dos dois ícones em relação ao tipo de tela é feita automaticamente pelo sistema dependendo do dispositivo em que a aplicação se encontra (Peixoto, 2016).
- Pasta *values* basicamente contém todos os recursos da aplicação que não podem fazer parte das pastas anteriores como configurações de cores, configuração de *strings*, tipos de temas etc (Peixoto, 2016).

- *Gradle Script* se trata de um grupo, que reúne todos os arquivos de configuração e de *build* da aplicação (Peixoto, 2016), são eles:
  - *build.gradle(Project)* é responsável pela compilação e configurações referente a todos os módulos criados no projeto (Peixoto, 2016).
  - *build.gradle(Module)* é responsável pela compilação e configurações de um modulo específico, sendo que, cada modulo deve conter seu próprio *build.gradle(Module)* diferenciando pelo *build.gradle(Module: My_Application.*nomeDoModulo*)* (Peixoto, 2016).

3.3 *Android SDK*

O Kit de Desenvolvimento de *Software* para *Android* (*Android SDK*) auxilia os desenvolvedores a produzirem *software* para o sistema operacional *Android*. Dentro deste Kit de desenvolvimento encontra-se emuladores para simular celulares, exemplos de código-fonte, bibliotecas e ferramentas de desenvolvimento (CORDEIRO, 2018).

O SDK utiliza na concepção dos aplicativos a linguagem de programação Java e recentemente a linguagem de programação *Kotlin*, ambas são executada em um ambiente virtual específico que posteriormente irá rodar em um dispositivo *Android* (CORDEIRO, 2018).

A utilização do SDK comumente se dá através da *Integrated Developmente Encironment* (IDE) vinculada por meio de plug-in. Esta integração possibilita que o emulador

comunique com a plataforma de desenvolvimento tornando possível a depuração do código-fonte (LECHETA, 2017).

3.4 *Google* Mapas API

De modo geral, as APIs de localização auxiliam os desenvolvedores, dispondo de um conjunto de rotinas e padrões específicos que, por meio de scripts, possibilitam acessar determinadas funções referente ao serviço de localização (JUNIOR, 2018).

Na construção do aplicativo em questão, será utilizada a API SDK do *Maps* para *Android*. Trata-se de uma API de localização disponibilizada pala *Google* que traz uma série de recursos de mapeamento para aplicações *mobile*. Dentre os principais recurso estão adicionar mapas ao aplicativo, personalizar mapa, controlar a visualização do usuário dentre outros (DEVELOPERS, 2020 B).

Entres as muitas vantagens em utilizar o SDK do *Maps* se dá a sua total integração com a linguagem *Kotlin* e com *Android SDK*, também utilizados na construção deste aplicativo (DEVELOPERS, 2020 B).

3.5 *Google Firebase*

O *Firebase* consiste em uma plataforma de desenvolvimento da *Google* com o propósito de fornecer ferramentas e serviços focados em automatizar o processo *Backend* de aplicação *Web* e *Mobile*, renunciando a preocupações como segurança na comunicação, transferência de dados, limitações de infraestrutura, autenticação, compatibilidade entre dispositivos dentre outros fatores encarregados à plataforma *Firebase*. Deste modo, proporciona-se aos desenvolvedores mais tempo na implementação do *Frontend*, gerando ao final do processo otimização de tempo, qualidade e recursos (SILVA,2021).

A plataforma *Firebase* dispõe-se de dois planos para o uso de seus serviços. O plano *Spark* é gratuito para qualquer desenvolvedor, contendo um limite específico de autenticações, armazenamentos, invocações, hospedagem entre outros. Já o plano *Blaze* somente será cobrado taxas referente aos serviços com limite extrapolado em relação ao plano *Spark*, supondo que o limite para armazenamento no plano *Spark* seja 5GB, ao extrapolar este valor será cobrado uma taxa referente ao excedente utilizado. Como não será possível calcular neste momento, o

consumo de recursos desta aplicação, espera-se que seja suficiente o plano *Spark* (SILVA,2021).

Dentre os diversos serviço do *Firebase*, será utilizado nesta aplicação o *Firebase Realtime Database*, um banco de dados *Not Only Standard Query Language* (NoSQL) com a função de armazenar persistentemente os dados da aplicação. Mais informações sobre a estrutura de banco de dados NoSQL são apresentadas no tópico Banco de Dados do capítulo anterior (FIREBASE, 2019).

O *Firebase Realtime Database* salva os arquivos no formato JSON e os armazenam em nuvem fazendo atualizações em tempo real. No entanto, a API só permite execução de operações que podem ser executadas com rapidez, otimizando o processo em tempo real, o que não compromete a capacidade de resposta (FIREBASE, 2019).

Mesmo fazendo armazenamento em nuvem, o *Realtime Database* mantém os dados armazenados localmente no dispositivo, permitindo que a aplicação opere normalmente mesmo estando *off-line*. Ao recuperar a conexão com a internet, o serviço atualiza automaticamente com os dados locais, incluindo as alterações feitas no momento que a aplicação remota se encontrava *off-line* (FIREBASE, 2019).

Com base no que foi apresentado, em relação proposta da ferramenta *Firebase*, pode-se concluir que, a utilização desta ferramenta na concepção do aplicativo *mobile*, sugerido nesse trabalho, se viabilizou e possibilitará suporte à estrutura *Backend* que será utilizada na aplicação (FIREBASE, 2019).

3.6 Linguagem XML

A linguagem XML (*eXtensible Markup Language*) foi criada com o propósito de padronizar a codificação de documentos em uma linguagem legível para máquinas. Atualmente o XML também é utilizado como linguagem de estilização das UIs (Interface de Usuário) dos sistemas operacionais *Android*. Sua principal função e de definir padrões e formatos de exibição dos elementos gráficos que compõem a UI da aplicação. Deste modo a linguagem XML define como os dados serão expostos na tela do usuário incluído sua forma de visualização (MAGALHAES, 2020).

3.7 Desenvolvimento da Aplicação

Esta seção descreve o processo de criação da aplicação, expondo os métodos presentes no código-fonte. O processo de desenvolvimento é apresentado em tópicos representando suas respectivas funcionalidades.

3.7.1 Implementação do Mapa

Para a implementação do Mapa ao sistema é utilizado o *Maps Fragement* que trata-se de um *plugin* fornecido pelo *Google* que disponibiliza todos os recursos essenciais na implantação do *google* mapas. Para que seja possível implementar esta estrutura é necessário que se crie um certificado digital na *Google* API Serviços que irá gerar uma chave de API utilizada para liberar o acessos aos recursos dentro da aplicação. Também é necessário que o usuário conceda algumas permissões ao sistema, que devem ser implementadas, antes da utilização dos serviços, no arquivo *AndroidManifest.xml* (*GOOGLE* DEVELOPERS, 2020).

Após todo o processo necessário para utilizar o *plugin*, o *Google Maps Fragment* e incorporado ao projeto, gerando três arquivos principais. O *google_maps_api.xml(debug)* é utilizado para armazenar a chave de API gerada no certificado digital e inserida no arquivo pelo desenvolvedor. A *activity_maps.xml* infla na UI um único fragmento responsável por mostrar o mapa na tela através da classe *SupportMapFragment*, que pode ser chamada através da *tag* *<fragmente>* em qualquer *ViewGoup* da aplicação. Por fim o *MapsActivity.java* responsável por instanciar o *SuportMapFragment* no método *onCreate()* o qual irá inicializar automaticamente a visualização do mapa (GOOGLE DEVELOPERS, 2020).

Após todos estes processos iniciais o *google* mapas podem ser carregados na aplicação e sua visualização é conforme a figura 11. Observa-se que até este ponto não foi adicionada nenhuma funcionalidade à aplicação, possibilitando que o usuário apenas visualize o mapa. Todo o processo para a apresentação do *google* mapa é implementado na classe *MapsFragment*.

Figura 11 – Interface inicial do *Google* Mapas

Fonte: Elaborada pelo autor

O *Google Maps Fragment* oferece vários tipos de visualizações de mapas como Mapa Normal apresentado na figura 11, Mapa de Satélite, Mapa de Relevo, Mapa Híbrido e muitos outros. Cada mapa tem sua aplicação, como por exemplo o Mapa Normal indicado para navegação com carros, por possuir informações importantes para este fim, como nome de ruas. Para esta aplicação será utilizado o Mapa de Satélite por ser possível visualizar os detalhes presentes nas estruturas externas das propriedades rurais (*GOOGLE* DEVELOPERS, 2020).

Para alterar o modo de visualização de Mapa Normal que vem como padrão ao carregar a estrutura *Maps Fragment* e preciso alterar o valor da variável *mapType* presente no objeto mapa, este procedimento utiliza o comando

map.mapType=GoogleMap.MAP_TYPE_SATELLITE para alterar a visualização do mapa para modo Satélite como apresentado na Figura 12.

Figura 12 – Interface *Google* Mapa de Satélite

Fonte: Elaborada pelo autor

A figura 12 deve apresentar o comando para alterar a visualização do mapa, isso é o que foi implementado no método *onMapReady()* desenvolvido. Este método é chamado logo quando o mapa é instanciado e está pronto para uso. É a partir dele que todos os outros métodos relacionados ao mapa podem ser chamados e executados.

As localizações presentes no mapa se dão através de pontos onde cada ponto do mapa e representado por uma coordenada latitude e longitude. Para exemplificar foi instanciado uma

variável chamada local do tipo *LatLng* que e responsável por armazenar uma coordenada geográfica pegando como entrada a latitude o longitude. Este exemplo será passado para a variável local as coordenadas da PUC-GO área 3. Em seguida o método *addMarcker*() irá receber como parâmetro a variável local para marcar este local com um marcador (*GOOGLE* DEVELOPERS, 2020).

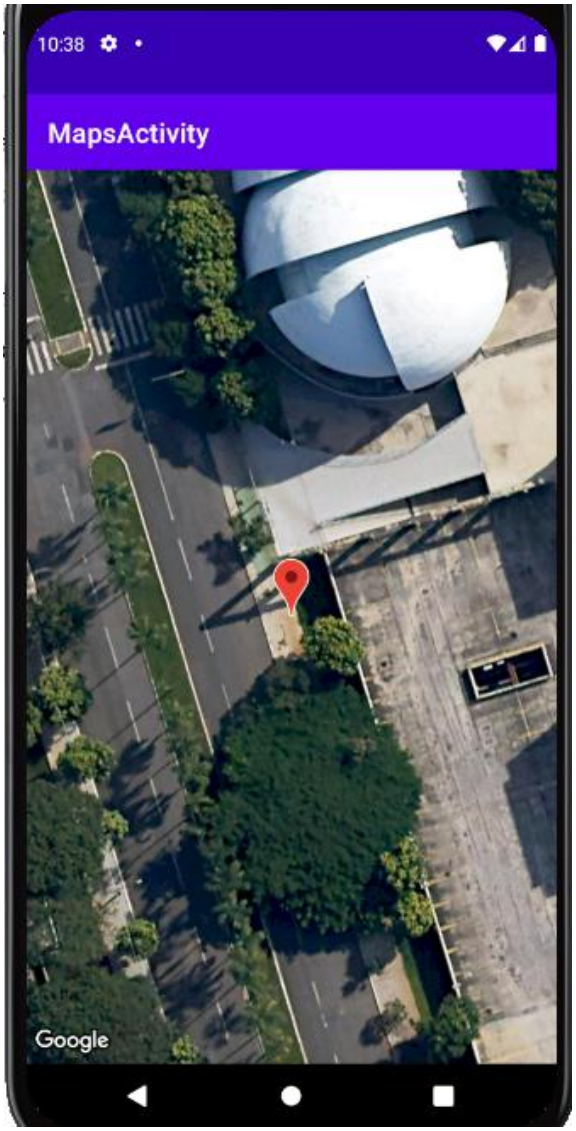
Em seguida é chamado o método *moveCamera*() que recebe como parâmetro de entrada a variável local e o parâmetro 20f. O presente método irá mover o mapa até a variável local e aplicar um *zoom* de 20f que pode ser 1f, 5f, 10f, 15f e 20f que corresponde respectivamente aos níveis mundo, continentes, cidades, ruas e construções. Deste modo o método *onMapReady()* ficara como na figura 13 e a visualização da aplicação e apresentada na figura 14.

Figura 13 –Exemplo do *onMapReady()*.

```
override fun onMapReady(googleMap: GoogleMap) {
    map = googleMap
    map.mapType = GoogleMap.MAP_TYPE_SATELLITE
    var local = LatLng( latitude: -16.6785952, longitude: -49.2443316)
    map.addMarker(MarkerOptions().position(local))
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(local,  zoom: 20f))
}
```

Fonte: Elaborada pelo autor

Figura 14 –Marcador do *Maps* na PUC-GO



Fonte: Elaborada pelo autor

3.7.2 Implementação das Permissões

Para ter acesso a localização do dispositivo, o usuário precisa conceder as permissões de localização ao sistema. Estas são divididas em duas categorias que são utilizadas em situações diferentes:

- Localização em Primeiro Plano fornece a localização do usuário apenas durante o uso do app em primeiro plano em um espaço curto de tempo. A declaração para acessar a localização em primeiro plano é ACCESS_COARSE_LOCATION que fornece a localização com uma

estimativa de 1,6 Km e ACCESS_FINE_LOCATION que fornece a localização com uma estimativa de 50 até menos de 3 metros (DEVELOPERS, 2021 B).

- Localização em Segundo Plano é necessária caso a aplicação precise acessar a localização do dispositivo em segundo plano ou precise compartilhar esta localização em tempo real com algum outro recurso. A declaração utilizada para conceder este tipo de permissão, se dá pelo ACCESS_BACKGROUND_LOCATION, o mesmo só poderá ser concedido caso já tenha tido a permissão concedida em primeiro plano, sendo que, a precisão será ditada pelo processo de permissão concedida em primeiro plano (DEVELOPERS, 2021 B).

Todos os processos de permissões são concedidos no arquivo *manifests* com a utilização da *tag uses-permission* passando o caminho e a declaração da permissão como mostra a figura 15 (DEVELOPERS, 2021 B).

Figura 15 –Declaração de Permissões

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

Fonte: Elaborada pelo autor

A presente aplicação requer acesso a localização em segundo plano para o monitoramento da *geofencing*, visto que este processo requer o monitoramento constantemente da localização do dispositivo em busca de identificar o acesso a uma cerca virtual. Deste modo, pode-se observar que na Figura 15, que apresenta a implementação destas permissões, foi declarado como permissão em primeiro plano o ACCESS_FINE_LOCATTION visando uma melhor precisão no sistema.

O modo mais utilizado para se fazer com que o usuário conceda as permissões necessárias seria requerendo-as em tempo de execução, de tal modo que impeça o usuário de prosseguir com o app caso elas não sejam concedidas (DEVELOPERS, 2021 B).

Para requere as permissões em tempo de execução, primeiramente é necessário averiguar anteriormente, se foi concedido permissão para essa execução, este processo é realizado logo após o processo de *login* através do método *ContextCompat.checkSelfPermission()* que retornará PERMISSION_GRANTED ou *PERMISSION_DENIED* respectivamente se a permissão foi ou não concedida. Caso retorne

PERMISSION_GRANTED então a permissão foi concedida e a aplicação segue normalmente dirigindo-se a tela *MapsFragment*. Caso retorne *PERMISSION_DENIED* então a permissão não foi concedida e deste modo abre-se a página de permissão *PermissionFragment* como apresentado na Figura 16. A página mostrará uma interface educacional para o usuário explicando o motivo da necessidade de concessão da permissão para o app (DEVELOPERS, 2021 C).

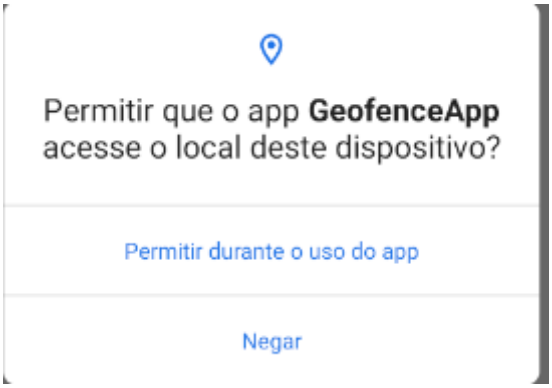
Figura 16 – Página Educacional de Permissão

Fonte: Elaborada pelo autor

Ao clicar em CONTINUAR é executado o método *shouldShowRequestPermissionRationale()* esse mostrará na tela do *smartphone* uma caixa de diálogo, em que o usuário, poderá escolher por conceder ou não a permissão requerida pelo

sistema como apresenta a Figura 17. Caso a requisição seja negar a aplicação permanece na página de *PermissionFragment*. Caso seja permitida a aplicação é direcionada a página principal *MapsFragment* (DEVELOPERS, 2021 C).

Figura 17 – Permissão de Localização em Primeiro Plano



Fonte: Elaborada pelo autor

Deste modo e requerida a permissão de localização em primeiro plano, essencial para obter a localização do usuário em tempo real durante o uso do app. Neste aplicação em questão, este processo, será suficiente para acessar ao mapa e a localização do usuário demonstrado no tópico anterior “3.4.1 Implementação do Mapa”. A permissão em segundo plano só será requerida ao usuário quando for adicionar uma *geofencing* ao mapa, visto que a localização em segundo plano só e necessária para o monitoramento da *geofencing*. O processo para requerer a permissão segue o mesmo modelo de requerer a permissão em primeiro plano exceto pelo fato que a partir do *Android* 10 (API 29) a função *shouldShowRequestPermissionRationale()* retornará uma caixa de diálogo diferente como mostra a Figura 18 (DEVELOPERS, 2021 B).

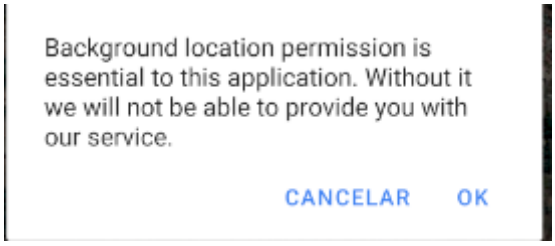
Figura 18 – Permissão de Localização em Segundo Plano

Fonte: Elaborada pelo autor

Quando o usuário seleciona a opção ‘Permitir o tempo todo’ a localização em segundo plano e ativada e a aplicação segue normalmente (DEVELOPERS, 2021 B).

A opção ‘Manter acesso durante o uso’ não ativa a localização em segundo plano. Nesta aplicação, esta decisão resulta na abertura de uma caixa de diálogo educacional, como na Figura 19, que instrui o usuário sobre a necessidade de conceder a permissão. Caso ele aperte em ok retorna a caixa de diálogo para requer a permissão, caso contrário a aplicação não permite que o usuário prossiga com a criação da *geofencing* (DEVELOPERS, 2021 B).

Figura 19 –Requerer Permissão Novamente

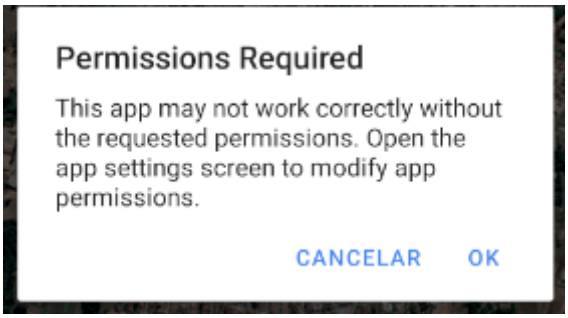


Fonte: Elaborada pelo autor

Por último a opção ‘Manter e não perguntar novamente’ que também não ativa a localização em segundo plano e não permite que a caixa de diálogo que requerer as permissões apareça novamente. Nesta aplicação, esta ação resulta no retorno à ação anterior, ao tentar criar

a *geofencing*, ao tentar criar novamente a *geofencing* o usuário se depara com uma caixa de diálogo, conforme a Figura 20, pedindo para entrar nas informações do app e conceder a permissão (DEVELOPERS, 2021 B).

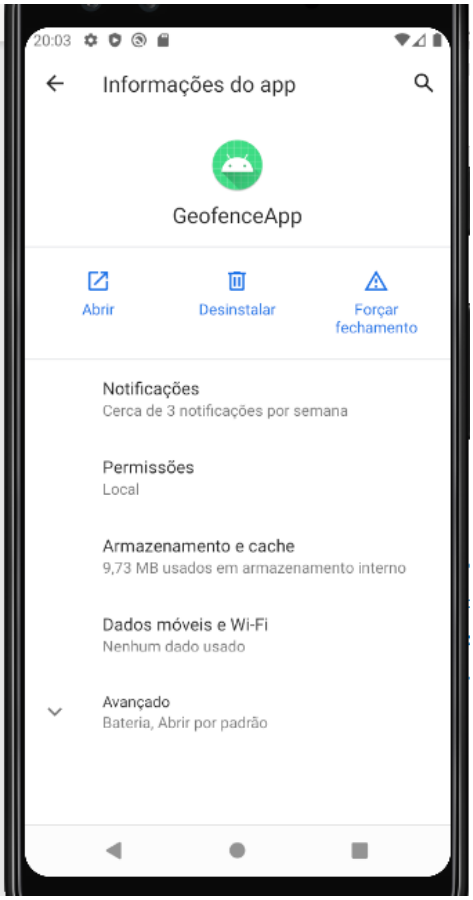
Figura 20 – Alterar Perdições em Informações do App



Fonte: Elaborada pelo autor

Ao pressionar OK e aberto a aba de informações, como na Figura 21, para que se conceda a permissão, e ao pressionar CANCELAR o app retorna ao estado anterior e não permite criar uma *geofencing* (DEVELOPERS, 2021 B).

Figura 21 – Aba Informações



Fonte: Elaborada pelo autor

Nesta aplicação todo este processo de permissões apresentado é implementado na classe *PermissionFragment()*, criada para manter todos os métodos unidos e organizados em um local especifico no código fonte.

3.7.3 Implementação das *Geofencing*

Para demostrar o processo de implementação da *geofencing* ao sistema, será detalhado em passo a passo para se adicionar uma nova *geofencing*, descrevendo os métodos utilizados no processo.

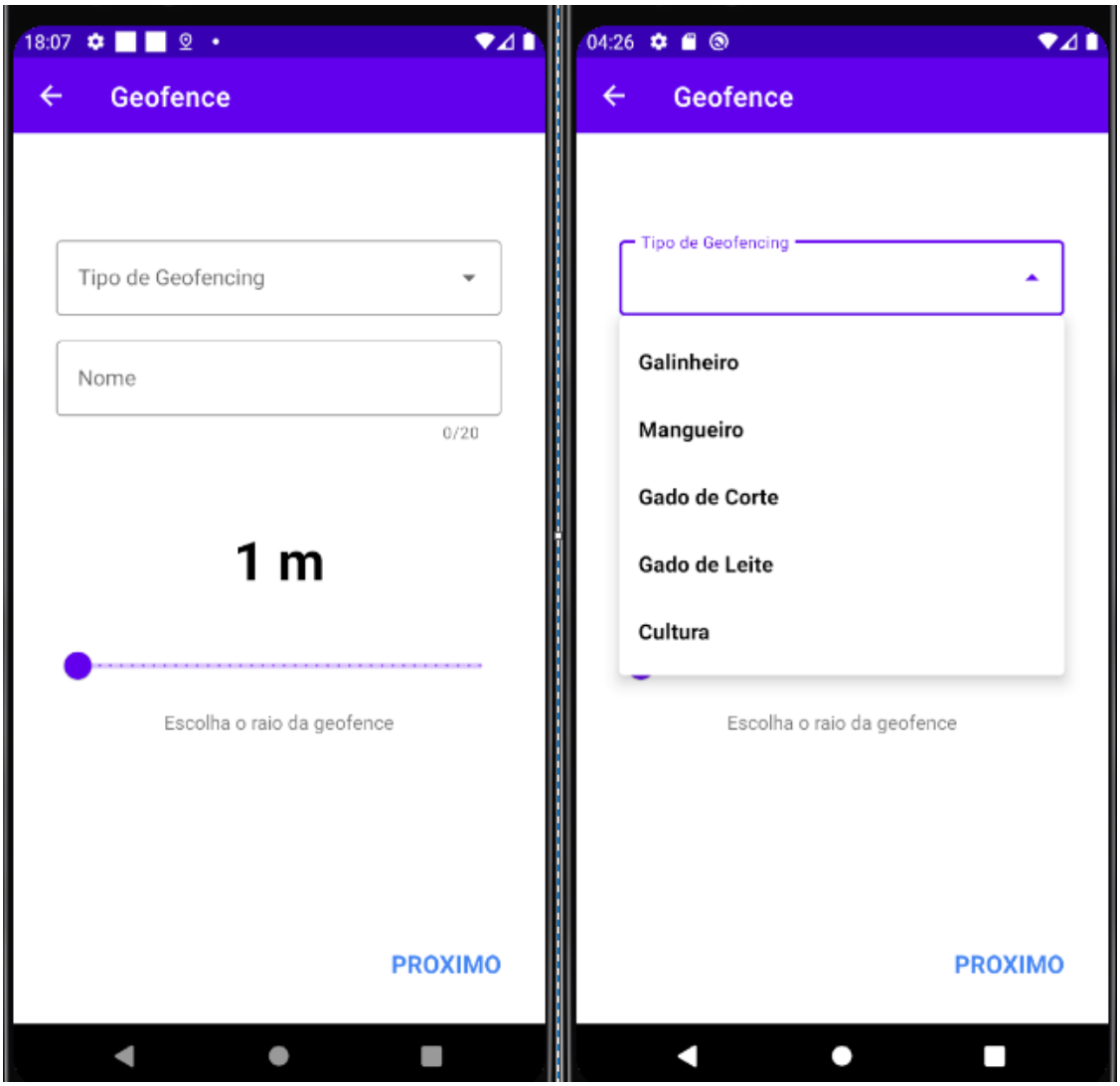
Na página do fragmento *MapsFragment* (visto anteriormente no tópico 3.7.1 Implementação do Mapa), é adicionado um botão circular de ação flutuante com um sinal de mais (+) como na Figura 22, responsável por adicionar uma nova *geofencing*. O método encarregado de monitorar e executar esta ação e o *setOnClickListener*() localizado dentro do método *onCreateView*(). O método *setOnClickListener*() monitora os eventos de *click* e quando o botão em questão é acionado e chamado o método *findNavController().navigate(R.id.action_mapsFragment_to_add_geofence_graph)* faz a navegação para a página *AddGeofenceFragment* onde inicia-se o processo de criação de uma nova *geofencing*.

Figura 22 – Botão Adicionar *Geofencing*

Fonte: Elaborada pelo autor

Na página *AddGeofenceFragment* o usuário define o tipo de *geofence*, nomeia e determina o tamanho do raio. Para este processo uma janela de entrada de dados *ExposedDropdownMenu* realiza a importação do tipo de *geofence* que pode ser escolhida, entre: Galinheiro, Mangueiro, Gado de Corte, Gado de Leite e Cultura. Em seguida o *TextInputLayout* importa o nome da *geofencing* e uma barra de progressão que varia de 1 a 1000 metros importa o raio como mostra a Figura 23. Ao pressionar em PROXIMO a aplicação é direcionada para próxima página que pode variar de acordo com tipo de *geofencing* escolhido, sendo que cada *geofencing* possui propriedades diferentes.

Figura 23 – Tipo, Nome e Raio da *Geofencing*



Fonte: Elaborada pelo autor

Ao clicar em PROXIMO com o tipo de *geofencing* escolhida como Galinheiro a aplicação e direcionada para a página EditGalinheiroFragment para que sejam os dados

importados do usuário referente a *geofencing* do tipo Galinheiro como apresenta a Figura 24. Os dados a serem importados são quantidade de animais, tipo de alimento ministrado, quantidade deste alimento que é consumida diariamente e a finalidade de produção que pode variar entre Postura de Ovos, Abate, Recria e Todos.

Figura 24 – Dados Galinheiro

Fonte: Elaborada pelo autor

Caso o tipo de *geofencing* selecionado seja Mangueiro é carregado a página EditMangueiroFragment que apresenta o mesmo padrão de importação de dados do galinheiro

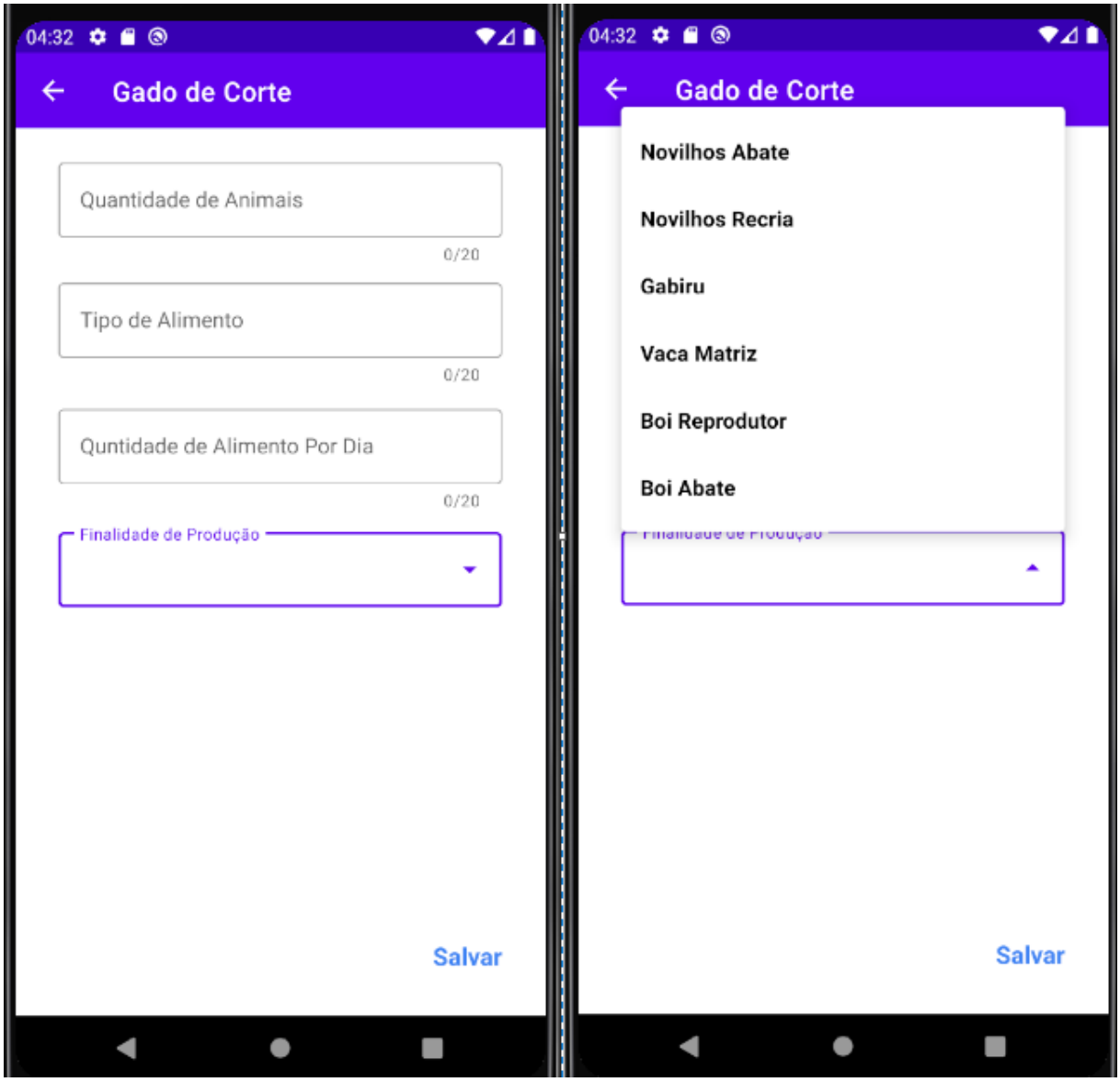
exceto pela finalidade de produção que mudará as opções para Abate, Recria e Abate Recria como apresentado na Figura 25.

Figura 25 – Dados Mangueiro

Fonte: Elaborada pelo autor

Para o tipo de *geofencing* Gado de Corte, a página recarregada é EditGadoDeCorteFragment e os dados a serem importados são quantidade de animais, tipo de alimento ministrado ao rebanho, quantidade deste alimento que é consumida diariamente e a finalidade do rebanho que pode variar entre Novilhos Abate, Novilhos Recria, Gabiru, Vaca Matriz, Boi Reprodutor e Boi Abate como apresentado na Figura 26.

Figura 26 –Dados Gado de Corte



Fonte: Elaborada pelo autor

Caso o tipo de *geofencing* seja Gado de Leite a página de importação de dados que e carregada na aplicação é a página EditGadoDeLeiteFragment e o padrão de importação dos dados seque o mesmo do gado de corte, exceto pela finalidade de produção que e substituída pela quantidade de litros de leite por dia, como apresenta a Figura 27.

Figura 27 –Dados Gado de Leite

Fonte: Elaborada pelo autor

Por fim o tipo de *geofencing* Cultura responsável pelas plantações da propriedade. Os dados são importados por meio da página EditCulturaFragment e os dados são: Tipo de Cultura, Área Plantada, Quantidade de Sementes e Data Prevista Para Colheita. Como apresenta a Figura 28.

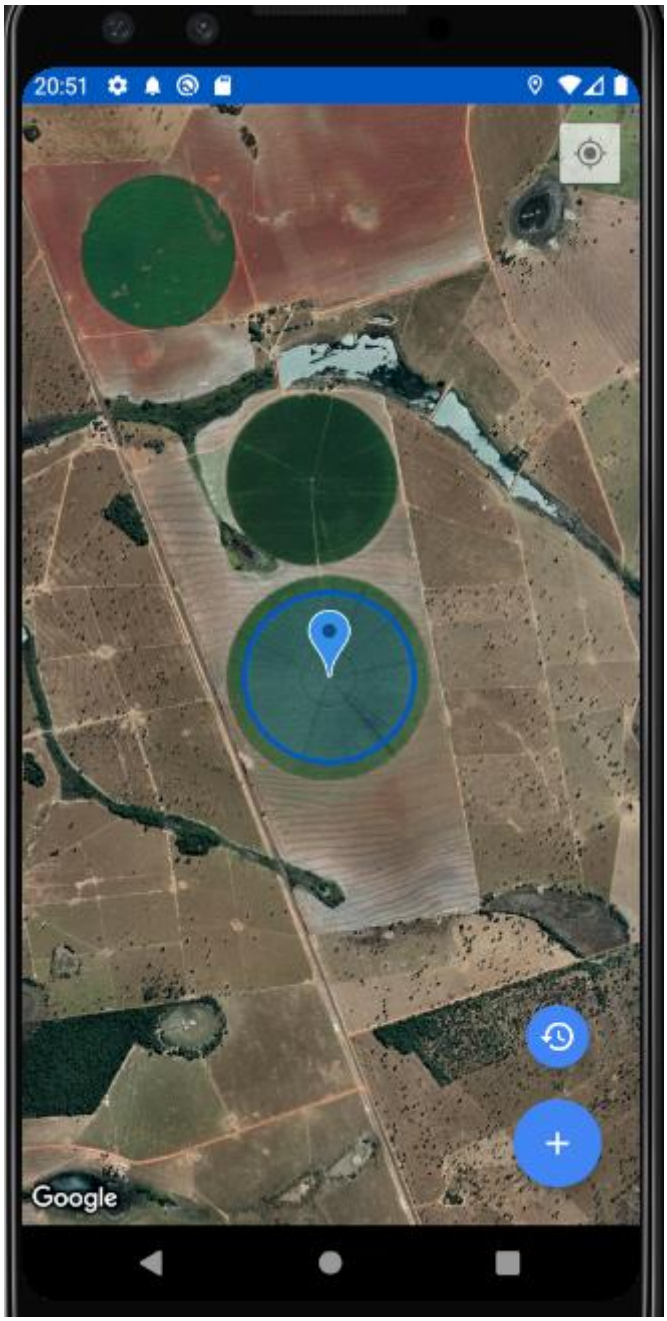
Figura 28 –Dados Cultura

Fonte: Elaborada pelo autor

Ao pressionar o botão SALVAR a aplicação retorna ao *MapsFragment* e pede ao usuário que de um longo *click* na região do mapa onde deseja adicionar a nova *geofencing*. A ação de longo *click* e monitorada pela API do *Google* Mapas através do método *onMapLongClick* que retorna as coordenados clicadas pelo usuário. As coordenados retornadas pelo método são enviadas ao método *setupGeofence* juntamente com todas as demais informações da *geofencing* importadas no processo anterior. No método *setupGeofence* são

salvas as coordenadas geográficas no banco de dados do *Firebase* e adicionado ao mapa um marcador e um círculo com base no ponto e no raio como apresenta a Figura 29.

Figura 29 – *Geofencing* Criada



Fonte: Elaborada pelo autor

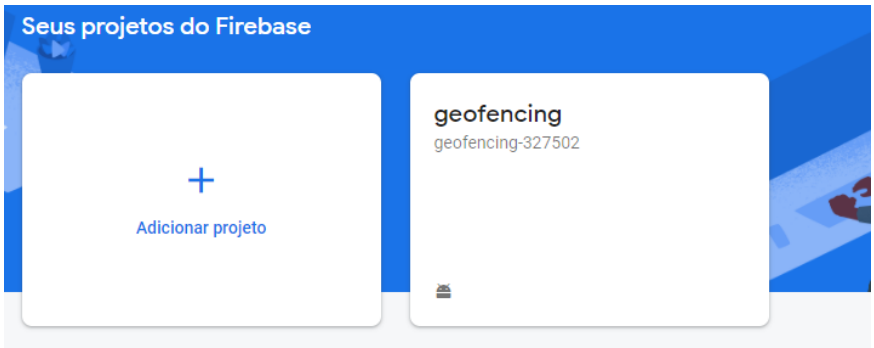
3.7.4 Implementação do *Login* do Usuário

O *login* é a primeira interface que o usuário terá acesso na aplicação. Esta etapa e de grande importância, pois através deste processo é possível vincular o usuário as suas *geofences* e dados relacionados a ela. Estes dados são armazenados no banco de dados *Realtime Database* e este processo será detalhado posteriormente no tópico de implementação do Banco de Dados.

A autenticação do usuário e realizada através do sistema de *Login* do *Google* vinculado a aplicação por meio do sistema *Firebase*. Para que seja efetuado o vínculo, necessita se que o projeto tenha no mínimo a versão de API 16 instalada, a aplicação deve rodar em sistemas *Android* com versões mínima de 4.1. É necessário a importação dos pacotes *com.android.tools.build:gradle* e *compileSdkVersion* no *build.gradle* do modulo e assim elencar as ferramentas necessárias no processo de construção feitas (DEVELOPERS, 2021 D).

Após a importação dos pacotes é necessário linkar a aplicação ao *Firebase*. Primeiramente deve ser criado uma conta de desenvolvedor no *Firebase* e através desta criar um projeto na plataforma. Após a criação do novo projeto (*geofencing*) o console do *Firebase* ficara como mostra a Figura 30 (DEVELOPERS, 2021D).

Figura 30 – Projeto *Firebase*

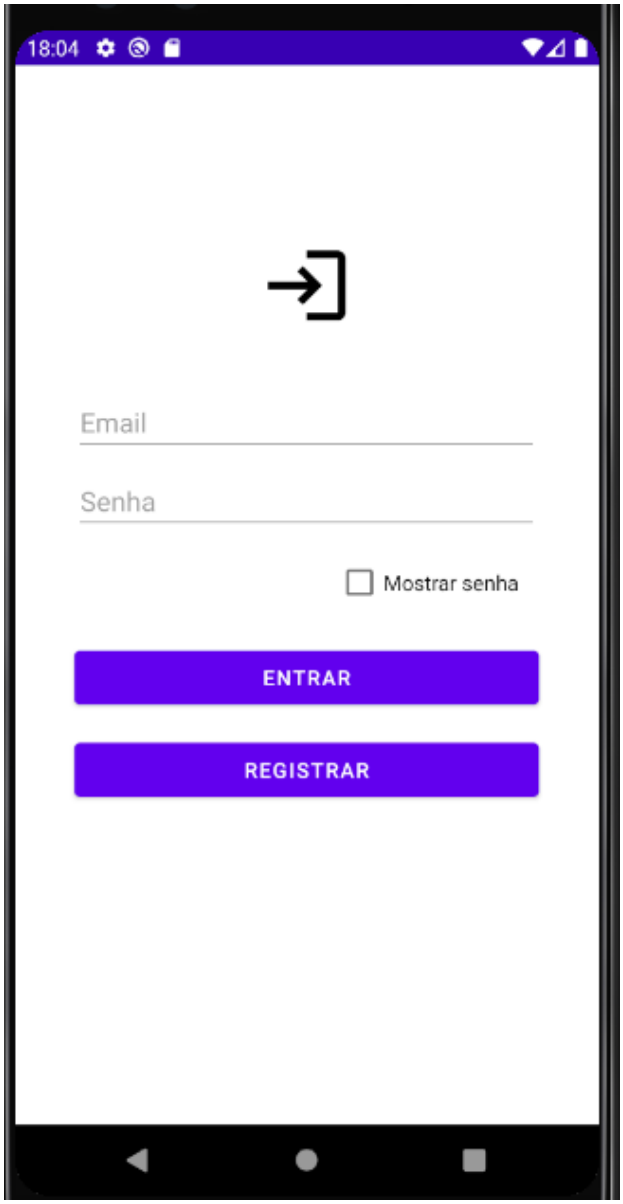


Fonte: Elaborada pelo autor

Com o projeto criado no *Firebase* o próximo passo é entrar no projeto e fazer o *Download* do arquivo de configuração *google-services.jason* que deve ser importado no modulo do aplicativo. Para ativar os componentes do *Firebase* é necessário adicionar os *plug-in com.google.GSM:google-services:4.3.10* no *build.gradle* do projeto e *com.google.GSM.google-services* no *build.gradle* do módulo. Ao término deste processo a vinculação do projeto do *Firebase* estará concluído, no entanto é necessário que se faça a declaração das dependências do *Firebase* utilizadas. No caso do *Login* é necessário a importação do *com.google.firebase:firebase-auth* no *build.gradle* do módulo para fazer a autenticação do usuário (DEVELOPERS, 2021 D).

Após concluída a etapa de vinculação da aplicação com *Firebase* é criada a página de *Login* do Usuário, mostrado na Figura 31, composta por dois editores de textos responsáveis pela aquisição do e-mail e senha além de dois botões ENTRAR e REGISTRAR responsáveis respectivamente por efetuar *login* e registrar usuário caso ainda não possua conta vinculada a aplicação.

Figura 31 – Página de *Login*



Fonte: Elaborada pelo autor

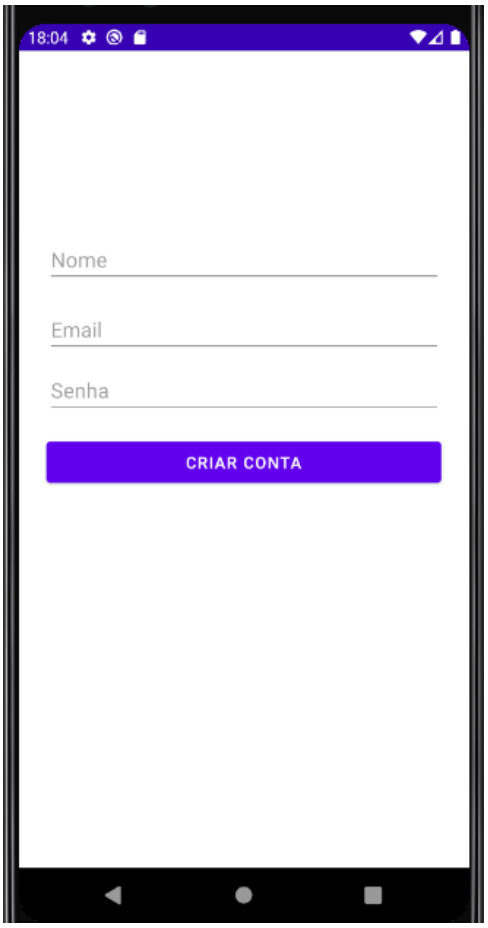
As funcionalidades do *Login* ficam no arquivo *LoginFragment.kt*. a função *onCreateView*() é responsável por inflar a interface na tela. Após este processo entra automaticamente a função

onViewCreated() que monitora os eventos de *click* nos botões ENTRAR e REGISTRAR através do método *setOnClickListener*().

Ao clicar no botão ENTRAR são capturados os dados dos campos da interface o e-mail e a senha. Posteriormente é chama a função *loginUser*(), responsável por validar os dados digitados e consultar no banco de dados *Firebase* através método *signInWithEmailAndPassword*() se o usuário possui cadastro e está autenticado para acessar a aplicação. Caso a autenticação seja bem sucedida é chamado o método *findNavController*() para navegar até a página de permissões vista no tópico 3.7.2 Implementação das Permissões. Caso o usuário não tenha sido autenticado é carregado na página uma mensagem de erro dizendo que não foi possível efetuar o *login*.

Ao clicar em REGISTRAR executa-se o método *findNavController*() para navegar até a página de criação de conta como mostra na Figura 32. A pagina é composta por três campos responsáveis respectivamente pelas aquisições do Nome, E-mail e Senha do usuário, além do botão CRIAR CONTA responsável por efetivar o cadastro.

Figura 32 – Página de Criação de Conta



Fonte: Elaborada pelo autor

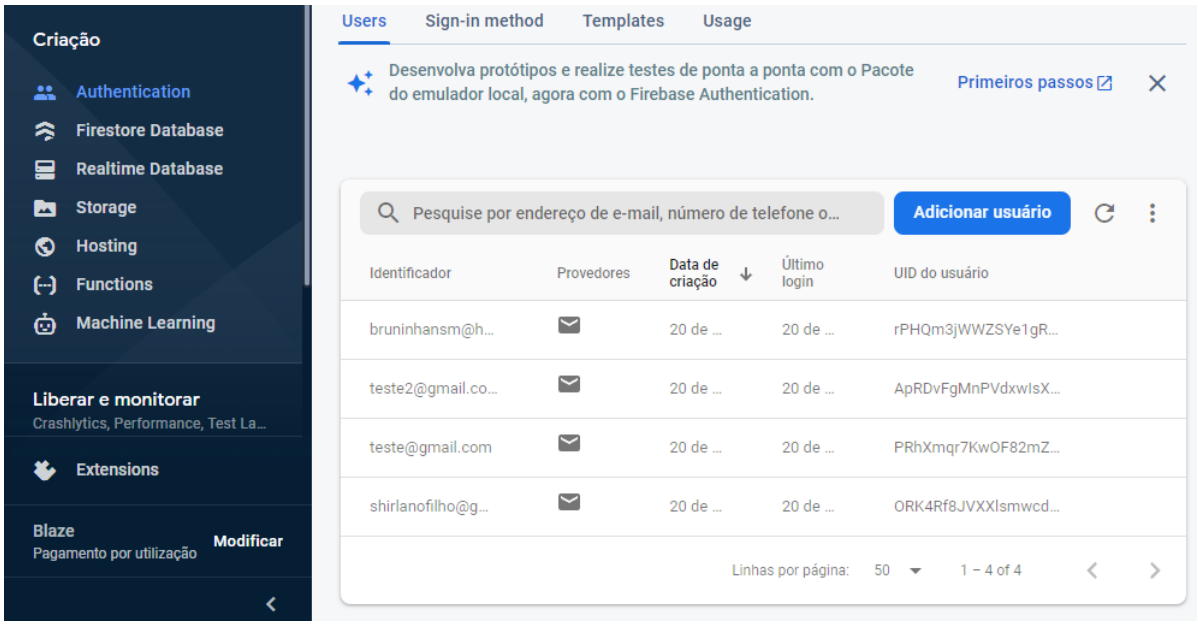
As funcionalidades do processo de criação de conta encontra-se no arquivo *CreateAccountFragment.kt*. Da mesmo forma do *login* a função *onCreateView*() e responsável por

inflar a interface na tela e a função *onViewCreated*() por monitorar o evento de *click* do botão CRIAR CONTA através do método *setOnClickListener*().

Ao clicar em CRIAR CONTA são capturados os dados dos campos da interface nome, e-mail e senha. Posteriormente é chama a função *createNewAccount*(), responsável por validar os dados digitados, criar e vincular uma nova conta no banco de dados *Firebase*. Caso não tenha sido possível criar uma conta é apresentado na interface do usuário um erro dizendo que a autenticação falhou. Caso o *Firebase* tenha conseguido efetuar o cadastro é chamada a função *findNavController*().navigateUp() para retornar a página de *login* onde o usuário pode logar na aplicação.

Todas as contas criadas são armazenadas no banco de dados do *Firebase* e podem ser acessadas pelo desenvolvedor através da página do *Firebase* na aba *Authentication* no campo *Users*. como mostra a Figura 33.

Figura 33 – Página *Firebase* de Usuários.



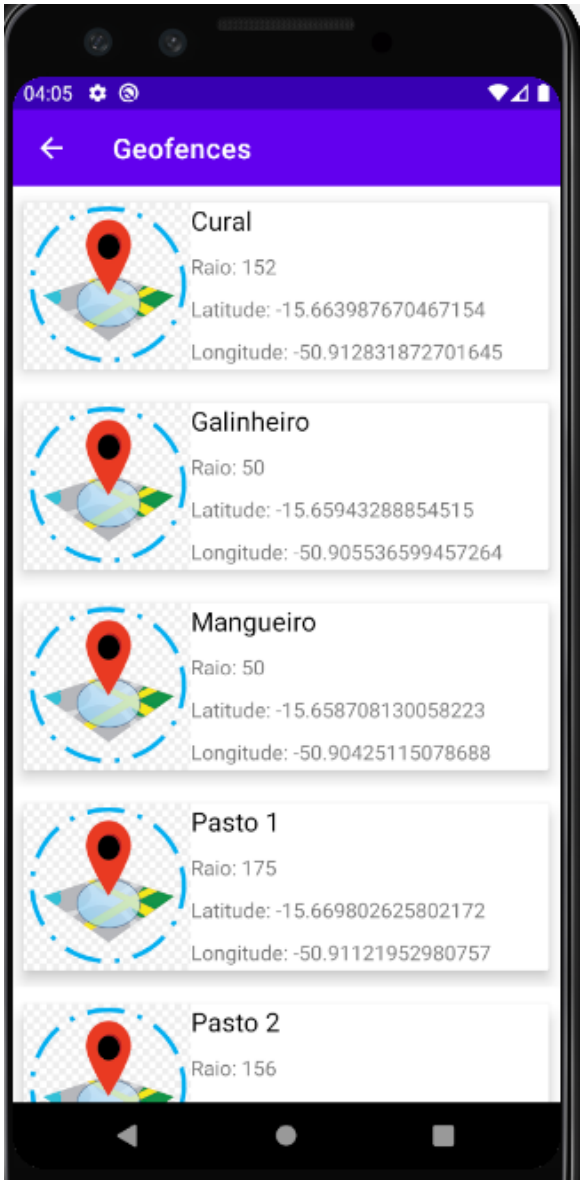
Fonte: Elaborada pelo autor

Os usuários, segundo a Figura 33, são armazenados em forma de tabela junto com seu Identificador email, o Provedor que tratasse do tipo de *login* utilizado pelo sistema que neste caso é email/senha, a Data de Criação da conta, a data do Último *Login* e o UID do usuário que é o identificador único dado pelo *Firebase* cada usuário para operações internas do sistema.

3.7.5 Implementação da Exibição dos Dados

A lista das *geofencing* do usuário pode ser acessada através do botão com ícone de histórico localizado logo acima do botão de adicionar *geofencing* na página *MapsFragment*. Ao selecionar o botão de histórico de *geofencing* a aplicação será direcionada a página *GeofencesFragment* que automaticamente acessará o banco de dados *Realtime Database* do *Firebase* para buscar todas as *geofencing* criadas a partir do usuário logado e listará todas elas através da API *RecyclerView* disponibilizada pelo *Android Studio*. A listagem se dá através dos nomes dado pelo usuário a cada *geofencing* sendo exibido juntamente ao raio e a localização que ela se encontra como mostra a Figura 34.

Figura 34 – Lista *de Geofences*



Fonte: Elaborada pelo autor

Ao clicar em cima de uma *geofencing* listada na aplicação, se é direcionada a página de dados que apresentará as informações contidas na *geofencing* como mostra a Figura 35.

Figura 35 – Dados da *Geofence*

Fonte: Elaborada pelo autor

3.7.6 Armazenamento da *Geofencing* no Banco de Dados

Através do processo de implementação da *geofence* descrito anteriormente no tópico 3.7.3 são obtidos todos os dados de uma *geofencing* que se pretende criar. Estes dados em um primeiro momento estarão localizados na página *MapsFragment* armazenado em uma lista global denominada "argumento". Ao receber estes dados o método *onViewStateRestored* verifica se realmente existe uma intenção de criar uma nova *geofencing*. Esta validação se dá por meio da variável "argumento.retorno" recebida no processo de criação da *geofence*. Caso não seja validado o processo de criação de *geofencing* por qualquer motivo, como por exemplo o usuário cancelou o processo, o processo segue normalmente na página principal *MapsFragment* sem qualquer alteração. Caso seja confirmado a criação entrará em ação o método *onMapLongClick*.

O método *onMapLongClick* trata-se de um *override* da API do *Google Maps*. Ou seja, esta classe está sendo reutilizada a partir da API do *Google Maps*. O método espera que o usuário de um longo clique no mapa e recebe como entrada a localização do ponto em que o usuário deu o *click*. Este ponto é o centro da *geofencing*. Após receber o ponto o método chama outra instância *setupGeofence* e passa como parâmetro o centro da *geofence*.

O método *setupGeofence* executa quatro outros métodos:

- *drawCircle*: recebe como paramento o ponto central da *geofence* e o raio, através da função *addCircle* disponibilizada pela API *Google Maps* cria no mapa um circula referente a *geofence*.
- *drawMarker*: recebe como parâmetro o ponto central e o nome da *geofence* contido na lista global "argumento". O método tem como função adicionar um marcador através da função *addMarker* disponibilizada pela API do *Google Maps* que coloca o marcador no centro da *geofence* juntamente com seu respectivo nome.
- *createGeoFence*: responsável por adicionar a *geofence* no servidor que monitora os movimentos de entrada e saída da *geofence*. Este processo será detalhado posteriormente no tópico 3.7.6 "Monitoramento da *Geofence*".
- *zoomToGeofence*: responsável por movimentar o *maps* de modo que seja possível visualizar toda a *geofence* criada independente do seu tamanho.

Após a execução dos quatro métodos são então inseridos os dados da *geofence* no banco de dados *Realtime Database* através da função *setValue* que persiste os dados no banco de dados.

Após inserido todos os dados no banco de dados, ainda no método *setupGeofence* e chamado o método *resetValues* para que a lista e as variáveis sejam reinicializadas para uma eventual nova *geofence*.

No banco de dados *Realtime Database* os dados são organizados como apresenta a Figura 36.

Figura 36 – Estrutura Final do *Realtime Database*

Fonte: Elaborada pelo autor

A estrutura de um banco de dados *Realtime Database* é baseada em uma árvore JSON onde todo elemento que aparece sem aspas como na Figura 35 é chamado do *child* ou nó da árvore e os que aparecem entre aspas são os dados. Os *childs* podem-se conter dados e ou outros *childs* filhos (Firebase, 2021).

O primeiro *child* "*Users*" tem como *child* filho o identificador do usuário que é dado pelo próprio *Firebase* no momento de criar conta na aplicação. Cada identificado referente e um usuário tem como *childs* filhos o *child* "name" que armazena o nome do usuário e o *child* "*Geofence*". O *child Geofence* possui *childs* filhos que são o nome das *geofences* criadas pelo respectivo usuário. Dentro da *child* que possui o nome da *geofence* criada têm-se os *childs* filhos "*lati*" que armazena a latitude da *geofence*, "*long*" que armazena a longitude, "nome" que armazena o nome da *geofence*, "raio" que armazena o raio da *geofence*, "tipo" que armazena o tipo de *geofence* e "Dados" que contém os *childs* que armazena os dados que variam de acordo com o tipo de cada *geofence*.

3.7.7 Carregamento das *Geofences* do Banco de Dados

O processo de busca das *geofences* salvas no banco de dados *Realtime Database* se inicia logo após a página principal *MapsFragment* ser carregada. No método *onMapReady* é feito o carregamento do mapa na página, como mostrado no tópico 3.7.1 "Implementação do Mapa". Logo após a execução de todas as rotinas de importação é chamado para execução o método *observeDatabase*. Este método faz uma busca no banco de dados por todas as *geofences* salvas na *child* do usuário logado e salva o nome de cada uma em uma lista de *geofence* 'geoLista'. Logo após todos os nomes de todas as *geofences* salvas na geoLista é chamado o método *getDatabase*().

O método *getDatabase*() faz uma busca no banco de dados atrás dos dados de cada uma das *geofences* salvas na geoLista e para cada *geofence* são chamados os métodos *drawMarker* e *drawCircle* vistos no tópico 3.7.6 "Armazenamento da *Geofence* no Banco de Dados" passando todos os parâmetros por eles requisitados. É chamado também para cada *geofence* o método *createGeoFence* atualizado no processo de monitoramento das *geofences*, porém este método será apresentado no tópico 3.7.8 "Monitoramento da *Geofence*". Ao término deste processo todas as *geofences* do usuário serão apresentadas no mapa.

3.7.8 Monitoramento da *Geofencing*

Para detectar as transições entre as *geofences* é preciso criar um *BroadcastReceiver* que para esta aplicação foi criado uma classe em *Kotlin* chamada de *GeofenceReceiver*. Esse elemento precisa ser especificado no arquivo *AndroidManifest* dentro de *application* como apresenta a Figura 37 (Developers, 2021 E).

Figura 37 – *Broadcast Geofence Receiver*

```
        </intent-filter>
    </activity>
    <receiver android:name=".GeofenceReceiver" />
</application>
```

Fonte: Elaborada pelo autor

A classe *GeofenceReceiver* é executada sempre que for detectada uma transição pré-determinada em uma das *geofences*. Para uma melhor compreensão do funcionamento do *Broadcast* é preciso voltar nos tópicos 3.7.6 “Armazenamento da *Geofence* no Banco de Dados” ou 3.4.6 “Carregamento das *Geofences* do Banco de Dados” para entender como os dados da *geofence* chegam até o método *createGeoFence* (Developers, 2021 E).

O método *createGeoFence* é responsável por criar uma variável do tipo *GeofencingClient* disponibilizado pela API do *Google Maps* e tem a função de monitorar a *geofence* para diversos tipos de ação. Para adicionar as *geofences* no *GeofencingClient* é preciso primeiramente criar um *Geofence.Builder* e passar para ele a chave da *geofence* que no caso deste aplicativo, é o nome dado pelo usuário à *geofence*, as coordenadas latitude e longitude da *geofence*, o raio, o tempo que as *geofences* ficarão armazenada no *GeofencigClient* que no caso desta aplicação é 24 horas que seria o tempo que o aplicativo ficaria ligado sem reinicializar e por fim o tempo de *delay* antes que uma ação seja tomada (Developers, 2021 E).

Posteriormente é criado outra variável do tipo *GeofencingRequest.Builder* que recebe o tipo de transição que pretende-se monitorar podendo variar em *GEOFENCE_TRANSITION_ENTER* para quando detectar uma entrada na *geofence*, *GEOFENCE_TRANSITION_EXIT* para quando detectar uma saída da *geofence*, *INITIAL_TRIGGER_DWELL* para quando detectar uma permanência dentro da *geofence* por um período pré-definido e *INITIAL_TRIGGER_ENTER* que será o utilizado nesta aplicação

para detectar quando um dispositivo já está dentro da *geofence*. E passado também para a variável o *Geofence.Builder* criado anteriormente (Developers, 2021 E).

O próximo passo é a criação de uma *Intent* recebendo como chave o nome da *geofence* e passando também a referência da classe *GeofenceReceiver*, responsável pela ação quando a transição é detectada. Posteriormente é criado uma *PendingIntent.getBroadcast* que recebe como parâmetro a *Intent* com a classe (Developers, 2021 E).

Por fim é adicionado ao *GeofencingClient* através da função *addGeofences* o *GeofencingRequest.Builder* contendo os parâmetros da *geofence* e o PendingIntent.get*Broadcast* contendo a classe *GeofenceReceiver*. Deste modo é adicionada uma *geofence* ao *GeofencingClient* responsável por monitorar todas as *geofences* nele contido (Developers, 2021 E).

No momento em que o *GeofencingClient* detecta a transição de entrada em uma determinada *geofence* o método *onReceive* localizado na classe *GeofenceReceiver* é chamado. Nele é recebida a *Intent* que possui a chave com mesmo valor do nome da *geofence* que foi adentrada. Deste modo é possível acessar o banco de dados *Realtime Database* e buscar os dados referentes aquela *geofence*. Com os dados em mão, o próximo passo é apresentá-los para o usuário por meio de *Toast* na tela do aplicativo através da função *makeText* e por meio de notificação através da classe *notificationManager*.

4 CONSIDERAÇÕES FINAIS

O presente trabalho tem como objetivo o desenvolvimento de um aplicativo *mobile* para sistemas operacionais *Android*, baseado na tecnologia *geofencing* com o propósito de auxiliar os produtores rurais no monitoramento das diversas informações contidas nas áreas de produção, de uma propriedade rural típica. O aplicativo busca monitorar as áreas da propriedade, de tal modo, que ao adentrar uma das áreas o usuário será notificado com informações importantes de produção e produtividade relacionada àquele local específico da propriedade.

Durante o processo de desenvolvimento foi notório o auxílio da plataforma *Android Studio* principalmente no processo de executar o código fonte por meio do emulador *Android*. Sem a presença de um emulador para fim de teste uma alternativa para testar o *software* seria executando-o em um dispositivo real e deslocando-se fisicamente com o dispositivo, o que demandaria muito tempo e esforço que foram totalmente supridos com a plataforma.

A API do *Google Maps* mostrou-se bem completa e versátil na implementação do mapa e seus derivados podendo ser acessados por meio de rotinas simples e de fácil compreensão. Possui uma vasta documentação de fácil acesso na plataforma oficial *Google Cloud Platform*.

O sistema *Firebase* mostrou-se de grande importância no processo de desenvolvimento da aplicação principalmente por automatizar o *Back-End* da aplicação como autenticação do usuário. *Firebase* foi de suma importância na parte de persistência dos dados através do *Realtime Database* que faz todo o processo de manipulação através de simples rotinas acessadas diretamente no código fonte. O plano básico gratuito disponibilizado pelo *Firebase* para utilização dos serviço é suficiente para o presente trabalho.

É notória à facilidade em se manipular a linguagem de programação *Kotlin*, ela possui uma forma de escrita muito mais clara com muito menos linhas de código em relação a linguagem Java. Além do que a plataforma *Android Studio* possui total suporte a linguagem de modo a converter automaticamente qualquer linha de código que eventualmente possa estar em Java.

O projeto mostrou-se de utilidade para produtores rurais que possuem diversas áreas de produção em sua propriedade. A forma facial e rápida com que o proprietário tem acesso as informações de cada área de produção agilizam o tempo além de melhorar a visualização dos dados para uma possível tomada de decisão ou simplesmente para se manter informado sobre seus investimentos na propriedade.

Além de beneficiar os produtores rurais com os benefícios deste trabalho, foi desenvolvido também o vasto conhecimento adquirido em relação aos métodos para desenvolvimento de aplicativos utilizando-se da *geofencing*. Este será de grande contribuição para a ciência através de futuros trabalhos que possam ser continuados através dos conhecimento adquiridos neste projeto.

4.1 Trabalhos Futuros

Pretende-se em trabalhos futuros:

- Implementar a possibilidade para o usuário criar distintas formas de *geofences* diferentes de um círculo.
- Aumentar a diversidade de informações para as *geofences*.
- Implementar uma forma de possuir mais tipos de usuários como usuários funcionários da fazenda de modo que possam editar do seu próprio *smart* algumas informações específicas de cada *geofencing* que sejam determinadas pelo usuário proprietário.
- Atribuir perfis para uma determinada fazenda, de modo que um usuário proprietário possa ter várias propriedades, e essas propriedades possam tem várias *geofencing* distintas, de tal forma que, os funcionários de uma fazenda, poderiam ou não, ter acesso as *geofencing* de outras fazendas, sendo que esta função pode ser editada, atribuída ou permitida pelo usuário proprietário.

REFERÊNCIAS

ALVES, Paulo Victor Alexandre. **Verificação Automática Georreferenciada**. 2018. Monografia (Conclusão do Curso) – Pontifícia Universidade Católica de Goiás, Escola de Ciências Exatas e da Computação, Goiânia.

BERNARDI, José Vicente Elias; LANDIM, Paulo M. Barbosa. **Aplicação do Sistema de Posicionamento Global (GPS) na coleta de dados**. 2002. Projeto de Pesquisa - UNESP/Campus de Rio Claro.

CAELUM. **O que é JAVA?** 2016. Disponível em: https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java. Acesso em: 28 abr. 2021.

CARVALHO, Rosana de. **Cresce O Número De Pessoas Com Acesso À Internet No Campo**. Panorama. 2015. Disponível em: https://pn7.com.br/cresce-o-numero-de-pessoas-comacesso-a-internet-no-campo. Acesso em: 15 mar. 2021.

COLUSSI, Joana. **Smartphone é ferramenta de gestão para sete em cada dez produtores rurais gaúchos**: Percentual de aparelhos móveis com internet cresceu de 17% para 67% em quatro anos no Estado. Uso massivo da conexão móvel ajuda a introduzir agricultores no mundo digital. [*S. l.*], 2018. Disponível em: https://gauchazh.clicrbs.com.br/economia/campo-e-lavoura/noticia/2018/10/smartphone-e-ferramenta-de-gestao-para-sete-em-cada-dez-produtores-rurais-gauchos-cjng5nfhi06gu01rx73frt8ct.html. Acesso em: 15 mar. 2021.

CORDEIRO, Fillipe. **Android SDK: O que é? Para que serve? Como usar?** 2018. Disponível em: https://www.androidpro.com.br/blog/android-studio/android-sdk/
Acesso em: 15 abr. 2021.

COSTA, Wilker. **Qual a melhor IDE para se começar na programação?** 2018. Disponível em: https://www.cursos.wlconsultoria.net/blog/qual-melhor-ide/. Acesso em: 01 maio 2021.

DEVELOPERS. **Arquitetura da plataforma**. 2020 A. Disponível em: https://developer.android.com/guide/platform?hl=pt-br. Acesso em: 20 abr. 2021.

DEVELOPERS. **Adicionar Mapas**. 2020 B. Disponível em: https://developer.android.com/training/maps?hl=pt-br. Acesso em: 20 abr. 2021.

DEVELOPERS. **Conheça o Android Studio**. 2021 A. Disponível em: https://developer.android.com/studio/intro?hl=pt. Acesso em: 20 abr. 2021.

DEVELOPERS. **Solicitar permissões de localização**. 2021 B. Disponível em: https://developer.android.com/training/location/permissions?hl=pt-br . Acesso em: 03 set 2021.

DEVELOPERS. **Solicitar permissões do app**. 2021 C. Disponível em: https://developer.android.com/training/permissions/requesting?hl=pt-br . Acesso em: 03 set 2021.

DEVELOPERS. **Adicionar o Firebase ao projeto para Android**. 2021 D. Disponível em: https://firebase.google.com/docs/android/setup?hl=pt-br . Acesso em: 03 set 2021.

DEVELOPERS. **Criar e monitorar Fronteiras Geográficas virtuais**. 2021 E. Disponível em: https://developer.android.com/training/location/geofencing . Acesso em: 03 set 2021.

DEVELOPERS. **Visão geral de projetos.** 2016. Disponível em: https://developer.android.com/studio/projects?hl=pt. Acesso em: 16 ago 2021.

DUARTE, Alessandra de Jesus; COELHO, Lucinda Maria de Fátima Rodrigues. **Sistema de Posicionamento Global (GPS): Uma aplicação da geometria analítica**. 2019. Revista Eletrônica do Curso de Licenciatura em Matemática – Uni-FACEF Centro Universitário Municipal de Franca.

FAUSTINO, Gleicy Kellen dos Santos et al. **Android e a influência do Sistema Operacional Linux.** 2017. Disponível em: http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/viewFile/829/728. Acesso em: 01 maio 2021.

FIREBASE. **Firebase Realtime Database**. 2019. Disponível em: https://firebase.google.com/docs/database?hl=pt-br. Acesso em: 15 maio 2021.

FIREBASE. **Estruturar o Banco de Dados**. 2021. Disponível em: https://firebase.google.com/docs/database/web/structure-data?hl=pt-br . Acesso em: 01 set 2021.

GOMES, Rodrigo Rodrigues Freire**. Latitude e Longitude**. 2017. UFG – Universidade Federal de Goiás.

GOOGLE DEVELOPERS. **Advanced Android in Kotlin**. 2020. Disponível em: https://developer.android.com/codelabs/advanced-android-kotlin-training-maps#0 . Acesso em: 04 set 2021.

IBGE. **Conceitos gerais: o que é cartografia?** 2021. Disponível em: https://atlasescolar.ibge.gov.br/conceitos-gerais/o-que-e-cartografia. Acesso em: 10 abr. 2021.

JUNIOR, Gilmar Margoti de Medeiros. **Implementação de um protótipo de aplicativo móvel com a integração com a ferramenta GOOGLE MAPS e o componente GPS dos Smartphones voltado ao auxílio no transporte público da cidade de Criciúma**. TCC (Graduação). Pontifícia Universidade Católica de Goiás. 2018.

KOTLIN. **Comece com Kotlin.** 2021. Disponível em: https://kotlinlang.org/docs/getting-started.html. Acesso em: 02 maio 2021.

LECHETA, Ricardo R. **ANDROID essencial com Kotlin**. 1° Edição. NovaTec Editora, 2017.

LIMA, Lucas. **O que é Android? Entenda a diferença para o iOS, do iPhone**. 2021. Disponível em: https://tecnoblog.net/432022/o-que-e-o-android-entenda-a-diferenca-para-o-ios-do-iphone/ . Acesso em: 02 maio 2021.

MAGALHÃES, André Lourenti. **O que é e para que serve o arquivo XML**. 2020. Disponível em: https://canaltech.com.br/software/xml-o-que-e/ . Acesso em: 16 set 2021.

MARILIA. **Banco de dados NoSQL: um novo paradigma – revista SQL magazine 102**. 2012. Disponível em: https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918. Acesso em: 29 abr. 2021.

MONICO, João Francisco Galera. **Posicionamento pelo NAVSTAR0GPS. Descrição, fundamentos e aplicações**: O sistema de posicionamento global (GPS): conceitos preliminares. 1. ed. Presidente Prudente: Editora UNESP, 2000.

NETO, Pedro César Gomes. **O GPS como fator motivacional no processo de Aprendizagem**. 2013. Dissertação (Mestrado em Matemática) – UENF Universidade Estadual do Norte Fluminense Darcy Ribeiro, Campos dos Goytacazes-RJ.

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

PAZ, Sérgio M.; CUGNASCA, Carlos E. **O Sistema de Posicionamento Global (GPS) e suas aplicações**. São Paulo, SP, 1997. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo.

PEREIRA, Expedito Henrique Ulisses**. A matemática do GPS**. 2014. Dissertação (Mestrado em Matemática) – Universidade Federal do Piauí, Centro de Ciências da Natureza.

PEIXOTO, Rodolfo. **Conceitos básicos do Android – part 1 – Estrutura do projeto no Android Studio**. Disponível em: https://rodolfopeixoto.medium.com/conceitos-b%C3%A1sicos-do-android-part-1-estrutura-do-projeto-no-android-studio-c9e6cb98e9c0 . Acesso em: 24 ago 2021.

RAILANO. **GPS - GLOBAL POSITIONING SYSTEM**, 2014. Disponível em: https://pt.slideshare.net/railano/aula08-geo-pgsensremotogps. Acesso em: 10 abr. 2021.

REDHAT. **Interface de programação de aplicações**: **O que é API?** 2021. Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 01 maio 2021.

RIBEIRO, Danilo Chagas. **Como Funciona o sistema GPS**. 2008. Disponível em: https://popa.com.br/como-funciona-o-sistema-gps/. Acesso em: 15 abr. 2021.

ROVEDA, Ugo. **Banco de dados: o que é, para que serve, tipos e como criar**. 2021. Disponível em: https://kenzie.com.br/blog/banco-de-dados/. Acesso em: 29 abr. 2021.

SANTANA, John Kennedy Ribeiro et al. **Precisão de GPS de smartphones: uma ferramenta para pesquisas acadêmicas e trabalhos em campo**. 2019. Revista de Geografia. Programa de Pós-Graduação em Geografia UFJF.

SANTOS , Lorena. **Revisão Bibliográfica: um recurso para melhorar o seu negócio**: Descubra porque a revisão bibliográfica é a base da pesquisa científica. [*S. l.*], 2018. Disponível em: https://formulajr.com.br/revisao-bibliografica/#:~:text=A%20import%C3%A2ncia%20da%20revis%C3%A3o%20bibliogr%C3%A1fica,de%20partida%20baseado%20em%20fatos.&text=Al%C3%A9m%20disso%2C%20a%20revis%C3%A3o%20bibliogr%C3%A1fica,diminuindo%20o%20ac%C3%BAmulo%20de%20erros. Acesso em: 15 mar. 2021.

SILVA, Débora. **Banco de dados**. 2015. Disponível em: https://www.estudopratico.com.br/banco-de-dados/. Acesso em: 29 abr. 2021.

SILVA, Eduardo. **Firebase: o que é e quando usar no desenvolvimento mobile?**.2021. Disponível EM: https://blog.geekhunter.com.br/firebase-o-que-e-e-quando-usar-no-desenvolvimento-mobile/#O_que_e_Firebase. Acesso em: 15 maio 2021.

SOUSA, Elisaudo. **Descubra porque a linguagem de programação Java é a número 1 no mundo.** 2017. Disponível em: http://www.3way.com.br/descubra-porque-linguagem-de-programacao-java-e-numero-1-no-mundo/. Acesso em: 28 abr. 2021.

SPROVIERO, Fernando. **Kotlin Android Extensions**. 2018. Disponível em: https://www.raywenderlich.com/84-kotlin-android-extensions . Acesso em: 13 set 2021.

STATLER, Stephen. **Beacon Technologies: The Hitchhiker's Guide to the Beacosystem.** 2016. San Diego California, USA.

VECCHIO, José Humberto. **Os 4 maiores desafios da gestão de pessoas no meio rural**. 2018. Disponível em: http://www.pioneersementes.com.br/blog/182/os-4-maiores-desafios-da-gestao-de-pessoas-no-meio-rural. Acesso em: 15 mar. 2021.

WHITE, Sarah K. ***What is geofencing? Putting location to work***. CIO Magazine, IDG Communications, Inc. 01 nov. 2017. Disponível em: https://www.cio.com/article/2383123/geofencing-explained.html. Acesso em: 15 mar. 2021.

APÊNDICES

APÊNDICE A - Código build.gradle(Project)

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        def nav_version = "2.3.5"
        classpath("androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version")
        classpath "com.android.tools.build:gradle:7.0.0"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.21"
        classpath 'com.google.GSM:google-services:4.3.10'



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

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

APÊNDICE B - Código build.gradle(Module)

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
```

```
    id 'kotlin-android-extensions'
    id 'com.google.GSM.google-services'
    id("androidx.navigation.safeargs")

}

android {
    compileSdk 31

    defaultConfig {
        applicationId "com.example.myapplication"
        minSdk 29
        targetSdk 31
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    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 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'com.google.android.GSM:play-services-maps:17.0.1'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'androidx.navigation:navigation-fragment-ktx:2.3.5'
    implementation 'androidx.navigation:navigation-ui-ktx:2.3.5'
    implementation 'com.google.android.GSM:play-services-maps:17.0.0'
    implementation 'com.vmadalin:easypermissions-ktx:0.1.0'
    implementation 'com.google.firebase:firebase-auth-ktx:21.0.1'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    implementation 'com.google.firebase:firebase-database-ktx:20.0.2'
    implementation 'com.google.android.GSM:play-services-location:18.0.0'
    implementation 'com.google.firebase:firebase-firestore:23.0.4'
    implementation 'com.google.android.GSM:play-services-fido:18.1.0'

    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
    implementation "androidx.room:room-runtime:2.2.6"

    implementation "com.google.dagger:hilt-android:2.31.2-alpha"

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

    implementation 'com.google.maps.android:android-maps-utils:0.5'

    implementation "androidx.cardview:cardview:1.0.0"
```

```
    implementation "androidx.recyclerview:recyclerview:1.2.0"
    def glide_version = "4.12.0"
    implementation "com.github.bumptech.glide:glide:$glide_version"
    annotationProcessor "com.github.bumptech.glide:compiler:$glide_version"
}
```

APÊNDICE C – Código AddGeofenceFragment.kt

```
package com.example.myapplication

import android.app.Activity
import android.os.Bundle
import android.text.TextUtils
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import android.widget.Toast
import androidx.navigation.fragment.findNavController
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import kotlinx.android.synthetic.main.fragment_create_account.*


class CreateAccountFragment : Fragment() {

    //Elemento de interface
    private var etName: EditText? = null
    private var etEmail: EditText? = null
    private var etPassword: EditText? = null
```

```
private var btnCreateAccount: EditText? = null

//Referencias ao banco
private var mDatabaseReference: DatabaseReference? = null
private var mDatabase: FirebaseDatabase? = null
private var mAuth: FirebaseAuth? = null
private val TAG = "CreateAccountFragment"

//Variaveis Globais
private var name: String? = null
private var email: String? = null
private var password: String? = null

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    return inflater.inflate(R.layout.fragment_create_account, container, false)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    mDatabase = FirebaseDatabase.getInstance()
    mDatabaseReference = mDatabase!!.reference!!.child("Users")
    mAuth = FirebaseAuth.getInstance()

    createlogin.setOnClickListener {
        etName = createname as EditText
        etEmail = createemail as EditText
        etPassword = createpassword as EditText
        createNewAccount()
    }
}
```

```
    private fun createNewAccount(){
        name = etName?.text.toString()
        email = etEmail?.text.toString()
        password = etPassword?.text.toString()

        if (!TextUtils.isEmpty(name) && !TextUtils.isEmpty(email) &&
!TextUtils.isEmpty(password)){
            Toast.makeText(context, "Informaçoes Prenchidas Corretamente",
Toast.LENGTH_SHORT).show()
        }else{
            Toast.makeText(context, "Entrar Com Mais Detalhes",
Toast.LENGTH_SHORT).show()
        }
        mAuth!!
            .createUserWithEmailAndPassword(email!!, password!!).addOnCompleteListener{
task->
                if (task.isSuccessful){
                    Log.d(TAG,"CreateUserWithEmail:Sucess")
                    val userdId = mAuth!!.currentUser!!.uid
                    val currentUserDb = mDatabaseReference!!.child(userdId)
                    currentUserDb.child("name").setValue(name)
                    updateUserInfoandUi()
                }else{
                    Log.w(TAG,"CreateUserWithEmail:Failure",task.exception)
                    Toast.makeText(context,"A autenticação falhou",
Toast.LENGTH_SHORT).show()
                }
            }
    }
    private fun updateUserInfoandUi(){
        findNavController().navigateUp()
    }
}
```

APÊNDICE D – Código CreateAccountFragment.kt

```
package com.example.myapplication

import android.app.Activity
import android.os.Bundle
import android.text.TextUtils
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import android.widget.Toast
import androidx.navigation.fragment.findNavController
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import kotlinx.android.synthetic.main.fragment_create_account.*


class CreateAccountFragment : Fragment() {
    //Elemento de interface
    private var etName: EditText? = null
    private var etEmail: EditText? = null
    private var etPassword: EditText? = null
    private var btnCreateAccount: EditText? = null

    //Referencias ao banco
    private var mDatabaseReference: DatabaseReference? = null
    private var mDatabase: FirebaseDatabase? = null
    private var mAuth: FirebaseAuth? = null
    private val TAG = "CreateAccountFragment"
```

```
    //Variaveis Globais
    private var name: String? = null
    private var email: String? = null
    private var password: String? = null

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_create_account, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        mDatabase = FirebaseDatabase.getInstance()
        mDatabaseReference = mDatabase!!.reference!!.child("Users")
        mAuth = FirebaseAuth.getInstance()

        createlogin.setOnClickListener {
            etName = createname as EditText
            etEmail = createemail as EditText
            etPassword = createpassword as EditText
            createNewAccount()
        }
    }

    private fun createNewAccount(){
        name = etName?.text.toString()
        email = etEmail?.text.toString()
        password = etPassword?.text.toString()

        if (!TextUtils.isEmpty(name) && !TextUtils.isEmpty(email) &&
!TextUtils.isEmpty(password)){
```

```
            Toast.makeText(context, "Informaçoes Prenchidas Corretamente",
Toast.LENGTH_SHORT).show()
        }else{
            Toast.makeText(context, "Entrar Com Mais Detalhes",
Toast.LENGTH_SHORT).show()
        }

        mAuth!!
            .createUserWithEmailAndPassword(email!!, password!!).addOnCompleteListener{
task->
                if (task.isSuccessful){
                    Log.d(TAG,"CreateUserWithEmail:Sucess")
                    val userdId = mAuth!!.currentUser!!.uid
                    val currentUserDb = mDatabaseReference!!.child(userdId)
                    currentUserDb.child("name").setValue(name)
                    updateUserInfoandUi()
                }else{
                    Log.w(TAG,"CreateUserWithEmail:Failure",task.exception)
                    Toast.makeText(context,"A autenticação falhou",
Toast.LENGTH_SHORT).show()
                }
            }
    }
    private fun updateUserInfoandUi(){
        findNavController().navigateUp()
    }

}
```

APÊNDICE E – Código EditCulturaFragment.kt

```
package com.example.myapplication
```

```
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.FragmentManager
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import kotlinx.android.synthetic.main.fragment_add_geofence.*
import kotlinx.android.synthetic.main.fragment_edit_cultura.*
import kotlinx.android.synthetic.main.fragment_edit_cultura.toolbar
import kotlinx.android.synthetic.main.fragment_geofences.*
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

class EditCulturaFragment : Fragment() {
    val argumento by navArgs<EditCulturaFragmentArgs>()
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_edit_cultura, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        toolbar.setNavigationOnClickListener {
```

```
            requireActivity().onBackPressed()
        }
        Salvar.setOnClickListener {
            var action =
EditCulturaFragmentDirections.actionEditCulturaFragmentToMapsFragment()
            action.nomeGeofence = argumento.nome
            action.raioGeofence = argumento.raio.toFloat()
            action.tipoGeofence = "Cultura"
            action.retorno = "addGeofence"
            action.dado1 =  tipo_cultura_imput.text.toString()
            action.dado2 = area_plantada_imput.text.toString()
            action.dado3 = quantida_sementes_imput.text.toString()
            action.dado4 = data_prevista_imput.text.toString()
            findNavController().navigate(action)
            }
        }
}
```

APÊNDICE F – Código EditGadoDeCorteFragment.kt

```
package com.example.myapplication

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import kotlinx.android.synthetic.main.fragment_add_geofence.*
import kotlinx.android.synthetic.main.fragment_add_geofence.autoCompleteTextView3
import kotlinx.android.synthetic.main.fragment_edit_cultura.*
```

```
import kotlinx.android.synthetic.main.fragment_edit_cultura.Salvar
import kotlinx.android.synthetic.main.fragment_edit_cultura.toolbar
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.*
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.alimento_por_dia_imput
import
kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.quantidade_de_animais_imput
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.tipo_alimentaçao_imput
import kotlinx.android.synthetic.main.fragment_edit_mangueiro.*


class EditGadoDeCorteFragment : Fragment() {

    val argumento by navArgs<EditCulturaFragmentArgs>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_edit_gado_de_corte, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val languages = arrayOf("Novilhos Abate", "Novilhos Recria", "Gabiru", "Vaca Matriz",
"Boi Reprodutor", "Boi Abate")
        val arrayAdapter = ArrayAdapter(requireContext(),R.layout.dropdown_item, languages)
        autoCompleteTextView3.setAdapter(arrayAdapter)

        toolbar.setNavigationOnClickListener {
            requireActivity().onBackPressed()
        }
        Salvar.setOnClickListener {
```

```
            var action =
EditGadoDeCorteFragmentDirections.actionEditGadoDeCorteFragmentToMapsFragment()
            action.nomeGeofence = argumento.nome
            action.raioGeofence = argumento.raio.toFloat()
            action.tipoGeofence = "Gado de Corte"
            action.retorno = "addGeofence"
            action.dado1 =  quantidade_de_animais_imput.text.toString()
            action.dado2 = tipo_alimentaçao_imput.text.toString()
            action.dado3 = alimento_por_dia_imput.text.toString()
            action.dado4 = autoCompleteTextView3.text.toString()
            findNavController().navigate(action)
        }
    }
}
```

APÊNDICE F – Código EditGadoDeLeiteFragment.kt

```
package com.example.myapplication

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import kotlinx.android.synthetic.main.fragment_add_geofence.*
import kotlinx.android.synthetic.main.fragment_edit_cultura.*
import kotlinx.android.synthetic.main.fragment_edit_cultura.Salvar
import kotlinx.android.synthetic.main.fragment_edit_cultura.toolbar
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.*
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.alimento_por_dia_imput
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.autoCompleteTextView3
```

```
import
kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.quantidade_de_animais_imput
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.tipo_alimentaçao_imput
import kotlinx.android.synthetic.main.fragment_edit_gado_de_leite.*


class EditGadoDeLeiteFragment : Fragment() {

    val argumento by navArgs<EditCulturaFragmentArgs>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_edit_gado_de_leite, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        toolbar.setNavigationOnClickListener {
            requireActivity().onBackPressed()
        }
        Salvar.setOnClickListener {
            var action =
EditGadoDeLeiteFragmentDirections.actionEditGadoDeLeiteFragmentToMapsFragment()
            action.nomeGeofence = argumento.nome
            action.raioGeofence = argumento.raio.toFloat()
            action.tipoGeofence = "Gado de Leite"
            action.retorno = "addGeofence"
            action.dado1 =  quantidade_de_animais_imput.text.toString()
            action.dado2 = tipo_alimentaçao_imput.text.toString()
            action.dado3 = alimento_por_dia_imput.text.toString()
```

```
            action.dado4 = quantidade_de_leite_imput.text.toString()
            findNavController().navigate(action)
        }
    }
}
```

APÊNDICE G – Código EditGalinheiroFragment.kt

```
package com.example.myapplication

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import kotlinx.android.synthetic.main.fragment_add_geofence.*
import kotlinx.android.synthetic.main.fragment_add_geofence.autoCompleteTextView3
import kotlinx.android.synthetic.main.fragment_edit_cultura.*
import kotlinx.android.synthetic.main.fragment_edit_cultura.Salvar
import kotlinx.android.synthetic.main.fragment_edit_cultura.toolbar
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.*

class EditGalinheiroFragment : Fragment() {
    val argumento by navArgs<EditCulturaFragmentArgs>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
```

```
        return inflater.inflate(R.layout.fragment_edit_galinheiro, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val languages = arrayOf("Postura de Ovos", "Abate", "Recria", "Todos")
        val arrayAdapter = ArrayAdapter(requireContext(),R.layout.dropdown_item, languages)
        autoCompleteTextView3.setAdapter(arrayAdapter)

        toolbar.setNavigationOnClickListener {
            requireActivity().onBackPressed()
        }
        Salvar.setOnClickListener {
            var action =
EditGalinheiroFragmentDirections.actionEditGalinheiroFragmentToMapsFragment()
            action.nomeGeofence = argumento.nome
            action.raioGeofence = argumento.raio.toFloat()
            action.tipoGeofence = "Galinheiro"
            action.retorno = "addGeofence"
            action.dado1 =  quantidade_de_animais_imput.text.toString()
            action.dado2 = tipo_alimentaçao_imput.text.toString()
            action.dado3 = alimento_por_dia_imput.text.toString()
            action.dado4 = autoCompleteTextView3.text.toString()
            findNavController().navigate(action)
        }
    }
}
```

APÊNDICE H – Código EditGeofenceFragment.kt

```
package com.example.myapplication
```

```
import android.app.AlertDialog
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.google.android.GSM.maps.model.LatLng
import com.google.android.material.dialog.MaterialAlertDialogBuilder
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import kotlinx.android.synthetic.main.fragment_edit_geofence.*
import kotlinx.android.synthetic.main.fragment_geofences.toolbar

class EditGeofenceFragment : Fragment() {

    val argumento by navArgs<EditGeofenceFragmentArgs>()
    var dados: String = ""
    var nome: String = "Defalt"
    var raio: String = "Defalt"
    var lati: String = "Defalt"
    var long: String = "Defalt"
    var tipo: String = "Defalt"
    var cultura: String = "Defalt"
    var area: String = "Defalt"
    var semente: String = "Defalt"
    var data: String = "Defalt"
    var animais: String = "Defalt"
```

```
    var alimento: String = "Defalt"
    var alimentoDia: String = "Defalt"
    var litroLetite: String = "Defalt"
    var finalidade: String = "Defalt"


    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val mDatabase = FirebaseDatabase.getInstance()
        val mAuth = FirebaseAuth.getInstance()
        val mDatabaseReference = mDatabase!!.reference!!.child("Users")
        var currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
        currentUserDb = currentUserDb.child("Geofence").child(argumento.nome)
        if (currentUserDb != null){
            var getdata = object : ValueEventListener{
                override fun onDataChange(snapshot: DataSnapshot) {
                    nome = snapshot.child("nome").getValue().toString()
                    raio = snapshot.child("raio").getValue().toString()
                    lati = snapshot.child("lati").getValue().toString()
                    long = snapshot.child("long").getValue().toString()
                    tipo = snapshot.child("tipo").getValue().toString()
                   when(tipo){
                      "Galinheiro"->{
                         animais = snapshot.child("Dados").child("animais").getValue().toString()
                         alimento = snapshot.child("Dados").child("alimento").getValue().toString()
                         alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                         finalidade =
snapshot.child("Dados").child("finalidade").getValue().toString()
                      }
                      "Mangueiro"->{
                         animais = snapshot.child("Dados").child("animais").getValue().toString()
```

```
                    alimento = snapshot.child("Dados").child("alimento").getValue().toString()
                    alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                    finalidade =
snapshot.child("Dados").child("finalidade").getValue().toString()
                }
                "Gado de Corte"->{
                    animais = snapshot.child("Dados").child("animais").getValue().toString()
                    alimento = snapshot.child("Dados").child("alimento").getValue().toString()
                    alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                    finalidade =
snapshot.child("Dados").child("finalidade").getValue().toString()
                }
                "Gado de Leite"->{
                    animais = snapshot.child("Dados").child("animais").getValue().toString()
                    alimento = snapshot.child("Dados").child("alimento").getValue().toString()
                    alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                    litroLetite =
snapshot.child("Dados").child("quantidadeLeite").getValue().toString()
                }
                "Cultura"->{
                    cultura = snapshot.child("Dados").child("tipo").getValue().toString()
                    area = snapshot.child("Dados").child("area").getValue().toString()
                    semente = snapshot.child("Dados").child("semente").getValue().toString()
                    data = snapshot.child("Dados").child("data").getValue().toString()
                }
            }
             printValues()
            }
        override fun onCancelled(error: DatabaseError) {
            TODO("Not yet implemented")
        }
```

```
        }
        currentUserDb.addValueEventListener(getdata)
    }
    return inflater.inflate(R.layout.fragment_edit_geofence, container, false)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    toolbar.setNavigationOnClickListener {
        requireActivity().onBackPressed()
    }
    toolbar.setTitle(argumento.nome)
}

private fun printValues() {
    dados =  "Nome:                    " + nome + "\n"
    dados += "Tipo:                      " + tipo + "\n"
    when(tipo){
        "Galinheiro" ->{

            dados += "N. de Animais:      " + animais + "\n"
            dados += "Tipo de Alimento: " + alimento + "\n"
            dados += "Alimentação/Dia:  " + alimentoDia + "\n"
            dados += "Finalidade:              " + finalidade + "\n\n"
        }
        "Mangueiro"->{
            dados += "N. de Animais:      " + animais + "\n"
            dados += "Tipo de Alimento: " + alimento + "\n"
            dados += "Alimentação/Dia:  " + alimentoDia + "\n"
            dados += "Finalidade:              " + finalidade + "\n\n"
        }
        "Gado de Corte"->{
            dados += "N. de Animais:      " + animais + "\n"
            dados += "Tipo de Alimento: " + alimento + "\n"
```

```
            dados += "Alimentação/Dia:  " + alimentoDia + "\n"
            dados += "Finalidade:              " + finalidade + "\n\n"
        }
        "Gado de Leite"->{
            dados += "N. de Animais:      " + animais + "\n"
            dados += "Tipo de Alimento: " + alimento + "\n"
            dados += "Alimentação/Dia:  " + alimentoDia + "\n"
            dados += "Litros de Leite/Dia: " + litroLetite + "\n\n"
        }
        "Cultura"->{
            dados += "Tipo de Cultura:    " + cultura + "\n"
            dados += "Area Plantada:       " + area + "\n"
            dados += "Semente Gasto:     " + semente + "\n"
            dados += "Data de Colheita:    " + data + "\n\n"
        }
    }
    dados += "Raio:           " + raio + "\n"
    dados += "Latitude:    " + lati + "\n"
    dados += "Logitude:    " + long + "\n"
    dados_text.text = dados
  }
}
```

APÊNDICE I – Código EditMangueiroFragment.kt

```
package com.example.myapplication

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
```

```
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import kotlinx.android.synthetic.main.fragment_add_geofence.*
import kotlinx.android.synthetic.main.fragment_add_geofence.autoCompleteTextView3
import kotlinx.android.synthetic.main.fragment_edit_cultura.*
import kotlinx.android.synthetic.main.fragment_edit_cultura.Salvar
import kotlinx.android.synthetic.main.fragment_edit_cultura.toolbar
import kotlinx.android.synthetic.main.fragment_edit_gado_de_corte.*

class EditMangueiroFragment : Fragment() {

    val argumento by navArgs<EditCulturaFragmentArgs>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_edit_mangueiro, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val languages = arrayOf("Abate", "Recria", "Abate e Recria")
        val arrayAdapter = ArrayAdapter(requireContext(),R.layout.dropdown_item, languages)
        autoCompleteTextView3.setAdapter(arrayAdapter)

        toolbar.setNavigationOnClickListener {
            requireActivity().onBackPressed()
        }
        Salvar.setOnClickListener {
            var action =
EditMangueiroFragmentDirections.actionEditMangueiroFragmentToMapsFragment()
            action.nomeGeofence = argumento.nome
```

```
            action.raioGeofence = argumento.raio.toFloat()
            action.tipoGeofence = "Mangueiro"
            action.retorno = "addGeofence"
            action.dado1 =  quantidade_de_animais_imput.text.toString()
            action.dado2 = tipo_alimentaçao_imput.text.toString()
            action.dado3 = alimento_por_dia_imput.text.toString()
            action.dado4 = autoCompleteTextView3.text.toString()
            findNavController().navigate(action)
        }
    }
}
```

APÊNDICE J – Código GeofenceReceiver.kt

```
package com.example.myapplication

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.os.Build
import android.util.Log
import android.widget.Toast
import androidx.core.app.NotificationCompat
import com.google.android.GSM.location.Geofence
import com.google.android.GSM.location.GeofencingEvent
import com.google.android.GSM.maps.model.LatLng
import com.google.firebase.FirebaseApp
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
```

```
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.google.firebase.database.ktx.database
import com.google.firebase.database.ktx.getValue
import com.google.firebase.ktx.Firebase
import kotlinx.android.synthetic.main.fragment_edit_geofence.*
import kotlin.random.Random

class GeofenceReceiver : BroadcastReceiver() {

    var dados: String = ""
    var nome: String = "Defalt"
    var raio: String = "Defalt"
    var lati: String = "Defalt"
    var long: String = "Defalt"
    var tipo: String = "Defalt"
    var cultura: String = "Defalt"
    var area: String = "Defalt"
    var semente: String = "Defalt"
    var data: String = "Defalt"
    var animais: String = "Defalt"
    var alimento: String = "Defalt"
    var alimentoDia: String = "Defalt"
    var litroLetite: String = "Defalt"
    var finalidade: String = "Defalt"

    override fun onReceive(context: Context?, intent: Intent?) {
        if (context != null) {
            val geofencingEvent = GeofencingEvent.fromIntent(intent)
            val geofencingTransition = geofencingEvent.geofenceTransition

            if (geofencingTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
geofencingTransition == Geofence.GEOFENCE_TRANSITION_DWELL) {
                //Recuperar dados do intent
```

```
var teste  =    geofencingEvent.triggeringGeofences
var key = teste.get(0).requestId
if (intent != null) {

    val mDatabase = FirebaseDatabase.getInstance()
    val mAuth = FirebaseAuth.getInstance()
    val mDatabaseReference = mDatabase!!.reference!!.child("Users")
    var currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
    currentUserDb = currentUserDb.child("Geofence").child(key)
    if (currentUserDb != null){
        var getdata = object : ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                nome = snapshot.child("nome").getValue().toString()
                raio = snapshot.child("raio").getValue().toString()
                lati = snapshot.child("lati").getValue().toString()
                long = snapshot.child("long").getValue().toString()
                tipo = snapshot.child("tipo").getValue().toString()
                when(tipo){
                    "Galinheiro"->{
                        animais =
snapshot.child("Dados").child("animais").getValue().toString()
                        alimento =
snapshot.child("Dados").child("alimento").getValue().toString()
                        alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                        finalidade =
snapshot.child("Dados").child("finalidade").getValue().toString()
                    }
                    "Mangueiro"->{
                        animais =
snapshot.child("Dados").child("animais").getValue().toString()
                        alimento =
snapshot.child("Dados").child("alimento").getValue().toString()
```

```
                    alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                    finalidade =
snapshot.child("Dados").child("finalidade").getValue().toString()
                }
                "Gado de Corte"->{
                    animais =
snapshot.child("Dados").child("animais").getValue().toString()
                    alimento =
snapshot.child("Dados").child("alimento").getValue().toString()
                    alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                    finalidade =
snapshot.child("Dados").child("finalidade").getValue().toString()
                }
                "Gado de Leite"->{
                    animais =
snapshot.child("Dados").child("animais").getValue().toString()
                    alimento =
snapshot.child("Dados").child("alimento").getValue().toString()
                    alimentoDia =
snapshot.child("Dados").child("alimentoDia").getValue().toString()
                    litroLetite =
snapshot.child("Dados").child("quantidadeLeite").getValue().toString()
                }
                "Cultura"->{
                    cultura = snapshot.child("Dados").child("tipo").getValue().toString()
                    area = snapshot.child("Dados").child("area").getValue().toString()
                    semente =
snapshot.child("Dados").child("semente").getValue().toString()
                    data = snapshot.child("Dados").child("data").getValue().toString()
                }
            }
            var message = printValues()
```

```
Toast.makeText(context, message, Toast.LENGTH_SHORT).show()


val CHANNEL_ID = "REMINDER_NOTIFICATION_CHANNEL"
var notificationId = 1589
notificationId += Random(notificationId).nextInt(1, 30)

val notificationBuilder =
NotificationCompat.Builder(context!!.applicationContext, CHANNEL_ID)
    .setSmallIcon(R.drawable.ic_alarm)
    .setContentTitle(context.getString(R.string.app_name))
    .setContentText(message)
    .setStyle(
        NotificationCompat.BigTextStyle()
            .bigText(message)
    )
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)

val notificationManager =
context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    val channel = NotificationChannel(
        CHANNEL_ID,
        context.getString(R.string.app_name),
        NotificationManager.IMPORTANCE_DEFAULT
    ).apply {
        description = context.getString(R.string.app_name)
    }
    notificationManager.createNotificationChannel(channel)
}
notificationManager.notify(notificationId, notificationBuilder.build())
```

```
                }
                override fun onCancelled(error: DatabaseError) {
                    TODO("Not yet implemented")
                }
            }
            currentUserDb.addValueEventListener(getdata)
        }
    }

    val triggeringGeofences = geofencingEvent.triggeringGeofences
    MapsFragment.removeGeofences(context, triggeringGeofences)

  }
 }
}

private fun printValues() : String{
 dados =  "Nome:                    " + nome + "\n"
 dados += "Tipo:                      " + tipo + "\n"
 when(tipo){
   "Galinheiro" ->{

     dados += "N. de Animais:      " + animais + "\n"
     dados += "Tipo de Alimento: " + alimento + "\n"
     dados += "Alimentação/Dia:  " + alimentoDia + "\n"
     dados += "Finalidade:              " + finalidade + "\n\n"
   }
   "Mangueiro"->{
     dados += "N. de Animais:      " + animais + "\n"
     dados += "Tipo de Alimento: " + alimento + "\n"
     dados += "Alimentação/Dia:  " + alimentoDia + "\n"
     dados += "Finalidade:              " + finalidade + "\n\n"
   }
```

```
            "Gado de Corte"->{
                dados += "N. de Animais:      " + animais + "\n"
                dados += "Tipo de Alimento: " + alimento + "\n"
                dados += "Alimentação/Dia:  " + alimentoDia + "\n"
                dados += "Finalidade:              " + finalidade + "\n\n"
            }
            "Gado de Leite"->{
                dados += "N. de Animais:      " + animais + "\n"
                dados += "Tipo de Alimento: " + alimento + "\n"
                dados += "Alimentação/Dia:  " + alimentoDia + "\n"
                dados += "Litros de Leite/Dia: " + litroLetite + "\n\n"
            }
            "Cultura"->{
                dados += "Tipo de Cultura:    " + cultura + "\n"
                dados += "Area Plantada:       " + area + "\n"
                dados += "Semente Gasto:     " + semente + "\n"
                dados += "Data de Colheita:    " + data + "\n\n"
            }
        }
        dados += "Raio:           " + raio + "\n"
        dados += "Latitude:    " + lati + "\n"
        dados += "Logitude:    " + long + "\n"
        return dados
    }

}
```

APÊNDICE K – Código GeofencesFragment.kt

```
package com.example.myapplication

import android.content.Intent
import android.net.Uri
```

```
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.FragmentManager
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import androidx.recyclerview.widget.LinearLayoutManager
import com.google.android.GSM.maps.model.LatLng
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import kotlinx.android.synthetic.main.fragment_geofences.*
import kotlinx.coroutines.launch
import java.lang.StringBuilder
import com.google.android.GSM.location.Geofence

class GeofencesFragment : Fragment() {

    private lateinit var liveAdapter: LiveAdapter
    var list: ArrayList<Live> = ArrayList()



    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val mDatabase = FirebaseDatabase.getInstance()
        val mAuth = FirebaseAuth.getInstance()
```

```
        val mDatabaseReference = mDatabase!!.reference!!.child("Users")
        var currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
        currentUserDb = currentUserDb.child("Geofence")
        var getdata = object : ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                list.clear()
                for (i in snapshot.children){
                    var nome = i.child("nome").getValue().toString()
                    var raio = i.child("raio").getValue().toString()
                    var lati = i.child("lati").getValue().toString()
                    var long = i.child("long").getValue().toString()
                    //Log.i("teste", nome)
                    var live = Live(
                        nome,
                        raio,
                        lati,
                        long,
                        "https://img2.gratispng.com/20180802/zxa/kisspng-geo-fence-gps-navigation-
systems-gps-tracking-unit-5b631a23649c80.2292979915332214114121.jpg"
                    )
                    list.add(live)
                }
                initRecyclerView()
            }
            override fun onCancelled(error: DatabaseError) {
                TODO("Not yet implemented")
            }
        }
        currentUserDb.addValueEventListener(getdata)
        return inflater.inflate(R.layout.fragment_geofences, container, false)
    }

    override fun onViewStateRestored(savedInstanceState: Bundle?) {
        super.onViewStateRestored(savedInstanceState)
```

```
            setupToolbar()
    }

    private fun initRecyclerView() {
        liveAdapter = LiveAdapter { live ->
            openLink(live.nome)
        }

        geofences_recyclerView.apply {
            layoutManager = LinearLayoutManager(requireContext())
            adapter = liveAdapter
        }
        liveAdapter.setList(list)
    }

    private fun openLink(nome: String) {
        val action =
GeofencesFragmentDirections.actionGeofencesFragmentToEditGeofenceFragment()
        action.nome = nome
        findNavController().navigate(action)
    }

    private fun setupToolbar() {
      toolbar.setNavigationOnClickListener {

findNavController().navigate(GeofencesFragmentDirections.actionGeofencesFragmentToMa
psFragment())
        }
    }
}
```

APÊNDICE L – Código Geofencing.kt

```
package com.example.myapplication

import android.graphics.Bitmap
import android.util.Log
import com.google.android.GSM.maps.model.LatLng
import com.google.android.GSM.maps.model.LatLngBounds
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.*
import com.google.maps.android.SphericalUtil
import kotlinx.coroutines.delay
import kotlin.math.sqrt


class Geofencing {
    var id: Long = 0L
    var nome: String = "Defalt"
    var raio: Float = 1F
    var tipo: String = "Defalt"
    var geoLatLng: LatLng = LatLng(0.0, 0.0)
    var preparadoParaAdd = false
    constructor()

    fun resetValues() {
        id = 0L
        nome = "Default"
        tipo = "Default"
        raio = 1F
        geoLatLng = LatLng(0.0, 0.0)
        preparadoParaAdd = false
    }

    private fun toMap(): Map<String, Any?> {
```

```
        return mapOf(
            "nome" to nome,
            "raio" to raio,
            "lati" to geoLatLng.latitude,
            "long" to geoLatLng.longitude,
            "tipo" to tipo

        )
    }

    fun upDatabase(){
        val mDatabase = FirebaseDatabase.getInstance()
        val mAuth = FirebaseAuth.getInstance()
        val mDatabaseReference = mDatabase!!.reference!!.child("Users")
        val currentUserDb =
mDatabaseReference!!.child(mAuth!!.currentUser!!.uid).child("Geofence").child(nome)
        val postValues = toMap()
        currentUserDb.updateChildren(postValues)
    }

    fun salvar(){
        val mDatabase = FirebaseDatabase.getInstance()
        val mAuth = FirebaseAuth.getInstance()
        val mDatabaseReference = mDatabase!!.reference!!.child("Users")
        val currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
        currentUserDb.child("Geofence").setValue(this)
    }

    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)
    }
}
```

APÊNDICE M – Código Live.kt

```
package com.example.myapplication

data class Live (
    var nome: String,
    var raio: String,
    var latitude: String,
    var longitude: String,
    var imagem: String

){
}
```

APÊNDICE N – Código LiveAdapter.kt

```
package com.example.myapplication

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import kotlinx.android.synthetic.main.res_item_live.view.*

class LiveAdapter(
    private val onItemClicked: (Live) -> Unit
```

```
) : RecyclerView.Adapter<RecyclerView.ViewHolder>(){
    private var items: List<Live> = ArrayList()
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
RecyclerView.ViewHolder {
        return LiveViewHolder(
            LayoutInflater.from(parent.context).inflate(R.layout.res_item_live, parent, false)
        )
    }
    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
        when (holder) {
            is LiveViewHolder -> {
                holder.bind(items[position], onItemClicked)
            }
        }
    }

    override fun getItemCount(): Int {
        return items.size
    }

    fun setList(liveList: List<Live>) {
        this.items = liveList
    }

    class LiveViewHolder constructor(
        itemView: View,
    ) : RecyclerView.ViewHolder(itemView) {
        private val liveTitle = itemView.nome
        private val liveAuthor = itemView.raio
        private val liveThumbnail = itemView.thumbnail
        private val latitude = itemView.latitude
        private val longitude = itemView.longitude

        fun bind(live: Live, onItemClicked: (Live) -> Unit) {
```

```
            liveTitle.text = live.nome
            liveAuthor.text = "Raio: " + live.raio
            latitude.text = "Latitude: " + live.latitude
            longitude.text = "Longitude: " + live.longitude
            val requestOptions = com.bumptech.glide.request.RequestOptions()
                .placeholder(R.drawable.ic_launcher_background)
                .error(R.drawable.ic_launcher_background)
            Glide.with(itemView.context)
                .applyDefaultRequestOptions(requestOptions)
                .load(live.imagem)
                .into(liveThumbnail)
            itemView.setOnClickListener {
                onItemClicked(live)
            }
        }
    }
}
```

APÊNDICE O – Código LoginFragment.kt

```
package com.example.myapplication

import android.Manifest
import android.os.Bundle
import android.text.InputType
import android.text.TextUtils
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import android.widget.Toast
```

```
import androidx.navigation.fragment.findNavController
import com.google.firebase.auth.FirebaseAuth
import com.vmadalin.easypermissions.EasyPermissions
import kotlinx.android.synthetic.main.fragment_create_account.*
import kotlinx.android.synthetic.main.fragment_login.*

class LoginFragment : Fragment() {

    private val TAG = "LoginFragment"
    private var email: String? = null
    private var password: String? = null
    private var etEmail: EditText? = null
    private var etPassword: EditText? = null
    private var mAuth: FirebaseAuth? = null

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_login, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        mostrar_senha.setOnClickListener{
            if(mostrar_senha.isChecked){

logpassword.setInputType(InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD)
;
            }
            else{
                logpassword.setInputType(InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD);
```

```
            }
        }

        login.setOnClickListener {
            etEmail = logusername as EditText
            etPassword = logpassword as EditText
            mAuth = FirebaseAuth.getInstance()
            loginUser()
        }

        registrar.setOnClickListener{
            findNavController().navigate(R.id.action_loginFragment_to_createAccountFragment)
        }
    }

    private fun loginUser(){
        email = etEmail?.text.toString()
        password = etPassword?.text.toString()

        if (!TextUtils.isEmpty(email) && !TextUtils.isEmpty(password)){
            Log.d(TAG,"Loggin do usuario")
            mAuth!!.signInWithEmailAndPassword(email!!, password!!).addOnCompleteListener
{
                task ->
                if(task.isSuccessful){
                    Log.d(TAG,"Logado com sucesso")
                    logar()
                }else{
                    Log.e(TAG,"erro ao logar", task.exception)
                    Toast.makeText(context, "Autenticação Falhou.",
Toast.LENGTH_SHORT).show()
                }
            }
        }else{
```

```
            Toast.makeText(context, "Entre com mais detalhes",
Toast.LENGTH_SHORT).show()
        }

    }

    private fun logar() {
        if (EasyPermissions.hasPermissions(context,
Manifest.permission.ACCESS_FINE_LOCATION)){

findNavController().navigate(LoginFragmentDirections.actionLoginFragmentToMapsFragme
nt())
        }else {
            findNavController().navigate(R.id.action_loginFragment_to_permissionFragment)
        }
    }
}
```

APÊNDICE P – Código MainActivity.kt

```
package com.example.myapplication

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

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

APÊNDICE Q – Código MapsFragment.kt

```
package com.example.myapplication

import android.Manifest
import android.annotation.SuppressLint
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
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.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.google.android.GSM.location.*
import com.google.android.GSM.maps.CameraUpdateFactory
import com.google.android.GSM.maps.GoogleMap
import com.google.android.GSM.maps.OnMapReadyCallback
import com.google.android.GSM.maps.SupportMapFragment
import com.google.android.GSM.maps.model.*
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.vmadalin.easypermissions.EasyPermissions
import kotlinx.android.synthetic.main.fragment_maps.*
import kotlinx.coroutines.delay
```

```
import kotlinx.coroutines.launch

lateinit var geofencingClient: GeofencingClient
private lateinit var fusedLocationClient: FusedLocationProviderClient

class MapsFragment : Fragment(), OnMapReadyCallback,
GoogleMap.OnMapLongClickListener{

    private lateinit var map: GoogleMap
    private lateinit var circle: Circle
    val PERMISSION_BACKGROUND_LOCATION_REQUEST_CODE = 2
    val argumento by navArgs<MapsFragmentArgs>()
    val geofencing = Geofencing()
    val geoList: ArrayList<String?> = ArrayList()
    val list: ArrayList<Live> = ArrayList()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        geofencingClient = LocationServices.getGeofencingClient(context)
        fusedLocationClient = LocationServices.getFusedLocationProviderClient(context)
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_maps, container, false)
    }

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

```
        mapFragment?.getMapAsync(this)
        add_geofence_fab.setOnClickListener{
            findNavController().navigate(R.id.action_mapsFragment_to_addGeofenceFragment)
        }

        geofences_fab.setOnClickListener {

findNavController().navigate(MapsFragmentDirections.actionMapsFragmentToGeofencesFra
gment())
        }
    }

    override fun onViewStateRestored(savedInstanceState: Bundle?) {
        super.onViewStateRestored(savedInstanceState)
        if (argumento.retorno == "addGeofence") {
            geofencing.nome = argumento.nomeGeofence
            geofencing.raio = argumento.raioGeofence
            geofencing.tipo = argumento.tipoGeofence
            geofencing.preparadoParaAdd = true
            displayInfoMessage()
        }
    }

    private fun displayInfoMessage() {
        lifecycleScope.launch {
            infoMessage_textView.visibility = View.VISIBLE
            delay(2000)
            infoMessage_textView.animate().alpha(0f).duration = 800
            delay(1000)
            infoMessage_textView.visibility = View.GONE
        }
    }
```

```
@SuppressLint("MissingPermission")
override fun onMapReady(googleMap: GoogleMap) {
    map = googleMap!!
    map.mapType = GoogleMap.MAP_TYPE_SATELLITE
    map.isMyLocationEnabled = true
    map.setOnMapLongClickListener(this)
    map.uiSettings.apply {
        isMyLocationButtonEnabled = true
        isMapToolbarEnabled = false
    }
    observeDatabase()
}



private fun observeDatabase() {
    val mDatabase = FirebaseDatabase.getInstance()
    val mAuth = FirebaseAuth.getInstance()
    val mDatabaseReference = mDatabase!!.reference!!.child("Users")
    var currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
    currentUserDb = currentUserDb.child("Geofence")
    currentUserDb.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(dataSnapshot: DataSnapshot) {
            geoList.clear()
            if (dataSnapshot.children != null){
                for (data in dataSnapshot.children) {
                    geoList.add(data.key)
                }
                getDatabase()
            }
        }
        override fun onCancelled(databaseError: DatabaseError) {
            Log.i("teste", "getUser:onCancelled", databaseError.toException())
        }
```

```
        })
    }

    private fun getDatabase() {
        val mDatabase = FirebaseDatabase.getInstance()
        val mAuth = FirebaseAuth.getInstance()
        val mDatabaseReference = mDatabase!!.reference!!.child("Users")
        var currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
        currentUserDb = currentUserDb.child("Geofence")
        for (data in geoList.iterator()){
            currentUserDb = currentUserDb.child(data.toString())
            currentUserDb.addValueEventListener(object : ValueEventListener {
                override fun onDataChange(dataSnapshot: DataSnapshot) {
                    var nome = dataSnapshot.child("nome").getValue().toString()
                    var raio = dataSnapshot.child("raio").getValue().toString().toFloat()
                    var geoLatLng =
LatLng(dataSnapshot.child("lati").getValue().toString().toDouble(),
                        dataSnapshot.child("long").getValue().toString().toDouble())
                    drawMarker(geoLatLng, nome)
                    drawCircle(geoLatLng, raio)
                    createGeoFence(geoLatLng,nome, geofencingClient,raio.toString())
                }
                override fun onCancelled(databaseError: DatabaseError) {
                    Log.i("teste", "getUser:onCancelled", databaseError.toException())
                }
            })
            currentUserDb = currentUserDb.getParent()!!
        }
    }

    override fun onMapLongClick(p0: LatLng) {
        if (EasyPermissions.hasPermissions(context,
Manifest.permission.ACCESS_BACKGROUND_LOCATION)){
            if (geofencing.preparadoParaAdd && p0 != null){
```

```
            geofencing.geoLatLng = p0
            setupGeofence(p0)
        }else{
            Toast.makeText(requireContext(), "Você precisa criar uma nova geocerca
primeiro.", Toast.LENGTH_SHORT).show()
        }
    }else{
        EasyPermissions.requestPermissions(
            this,
            "A permissão de localização em segundo plano é essencial para este aplicativo. Sem
ele, não seremos capazes de lhe fornecer nosso serviço.",
            PERMISSION_BACKGROUND_LOCATION_REQUEST_CODE,
            Manifest.permission.ACCESS_BACKGROUND_LOCATION
        )
    }
}

private fun setupGeofence(location: LatLng) {
    lifecycleScope.launch {
            drawCircle(location, geofencing.raio)
            drawMarker(location, geofencing.nome)
            createGeoFence(location,geofencing.nome,
geofencingClient,geofencing.raio.toString())
            zoomToGeofence(circle.center, circle.radius.toFloat())
            delay(1000)
            geofencing.upDatabase()
            delay(3000)
        val mDatabase = FirebaseDatabase.getInstance()
        val mAuth = FirebaseAuth.getInstance()
        val mDatabaseReference = mDatabase!!.reference!!.child("Users")
        val currentUserDb = mDatabaseReference!!.child(mAuth!!.currentUser!!.uid)
            .child("Geofence")
            .child(argumento.nomeGeofence)
            .child("Dados")
```

```
    when(argumento.tipoGeofence){
        "Galinheiro"->{
            currentUserDb.child("animais").setValue(argumento.dado1)
            currentUserDb.child("alimento").setValue(argumento.dado2)
            currentUserDb.child("alimentoDia").setValue(argumento.dado3)
            currentUserDb.child("finalidade").setValue(argumento.dado4)
        }
        "Mangueiro"->{
            currentUserDb.child("animais").setValue(argumento.dado1)
            currentUserDb.child("alimento").setValue(argumento.dado2)
            currentUserDb.child("alimentoDia").setValue(argumento.dado3)
            currentUserDb.child("finalidade").setValue(argumento.dado4)
        }
        "Gado de Corte"->{
            currentUserDb.child("animais").setValue(argumento.dado1)
            currentUserDb.child("alimento").setValue(argumento.dado2)
            currentUserDb.child("alimentoDia").setValue(argumento.dado3)
            currentUserDb.child("finalidade").setValue(argumento.dado4)
        }
        "Gado de Leite"->{
            currentUserDb.child("animais").setValue(argumento.dado1)
            currentUserDb.child("alimento").setValue(argumento.dado2)
            currentUserDb.child("alimentoDia").setValue(argumento.dado3)
            currentUserDb.child("quantidadeLeite").setValue(argumento.dado4)
        }
        "Cultura"->{
            currentUserDb.child("tipo").setValue(argumento.dado1)
            currentUserDb.child("area").setValue(argumento.dado2)
            currentUserDb.child("semente").setValue(argumento.dado3)
            currentUserDb.child("data").setValue(argumento.dado4)
        }
    }
    geofencing.resetValues()
}
```

```
    }

    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))
        )
    }

    private fun drawMarker(location: LatLng, name: String) {
        map.addMarker(
            MarkerOptions().position(location).title(name)

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE))
        )
    }

    private fun zoomToGeofence(center: LatLng, radius: Float) {
        map.animateCamera(
            CameraUpdateFactory.newLatLngBounds(
                geofencing.getBounds(center, radius), 10
            ), 1000, null
        )
    }

    @SuppressLint("MissingPermission")
    private fun createGeoFence(location: LatLng, key: String, geofencingClient:
GeofencingClient, GEOFENCE_RADIUS: String) {
        val geofence = Geofence.Builder()
            .setRequestId(key)
            .setCircularRegion(location.latitude, location.longitude,
GEOFENCE_RADIUS.toFloat())
            .setExpirationDuration((24 * 60 * 60 * 1000).toLong())
```

```
            .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or
Geofence.GEOFENCE_TRANSITION_DWELL)
            .setLoiteringDelay(1000)
            .build()
        val geofenceRequest = GeofencingRequest.Builder()
            .setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
            .addGeofence(geofence)
            .build()
        val intent = Intent(context, GeofenceReceiver::class.java)
            .putExtra("key", key)
            .putExtra("message", "Geofence alert - ${location.latitude}, ${location.longitude}")
        val pendingIntent = PendingIntent.getBroadcast(
            context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT
        )
        geofencingClient.addGeofences(geofenceRequest, pendingIntent)
        Log.i("teste", "criou geofencinh: " + key)
    }

    companion object {
        fun removeGeofences(context: Context, triggeringGeofenceList:
MutableList<Geofence>) {
            val geofenceIdList = mutableListOf<String>()
            for (entry in triggeringGeofenceList) {
                geofenceIdList.add(entry.requestId)
            }
            LocationServices.getGeofencingClient(context).removeGeofences(geofenceIdList)
        }
    }
}
```

APÊNDICE R – Código PermissionFragment.kt

```
package com.example.myapplication
```

```
import android.Manifest
import android.content.pm.PackageManager
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.navigation.fragment.findNavController
import kotlinx.android.synthetic.main.fragment_permission.*
import com.vmadalin.easypermissions.EasyPermissions


class PermissionFragment : Fragment() {

    val PERMISSION_LOCATION_REQUEST_CODE = 1

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

    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        continue_button.setOnClickListener{
            if (EasyPermissions.hasPermissions(context,
Manifest.permission.ACCESS_FINE_LOCATION)){
                findNavController().navigate(R.id.action_permissionFragment_to_mapsFragment)
```

```
        }else{
            EasyPermissions.requestPermissions(
                this,
                "This application cannot work without Location Permission.",
                PERMISSION_LOCATION_REQUEST_CODE,
                Manifest.permission.ACCESS_FINE_LOCATION
            )
        }
    }

  }
}
```

APÊNDICE S – Código de 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 drawable - ic_add.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
```

```
    android:viewportHeight="24"
    android:tint="@color/white">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M19,13h-6v6h-2v-6H5v-2h6V5h2v6h6v2z"/>
</vector>
```

Pasta drawable – ic_alarm.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M22,5.72l-4.6,-3.86 -1.29,1.53
4.6,3.86L22,5.72zM7.88,3.39L6.6,1.86 2,5.71l1.29,1.53 4.59,-
3.85zM12.5,8L11,8v6l4.75,2.85 0.75,-1.23 -4,-2.37L12.5,8zM12,4c-4.97,0 -9,4.03 -
9,9s4.02,9 9,9c4.97,0 9,-4.03 9,-9s-4.03,-9 -9,-9zM12,20c-3.87,0 -7,-3.13 -7,-7s3.13,-7 7,-7
7,3.13 7,7 -3.13,7 -7,7z"/>
</vector>
```

Pasta drawable – ic_back.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="@color/white"
    android:autoMirrored="true">
  <path
      android:fillColor="@android:color/white"
```

```
    android:pathData="M20,11H7.83l5.59,-5.59L12,4l-8,8 8,8 1.41,-1.41L7.83,13H20v-
2z"/>
</vector>
```

Pasta drawable – ic_delete.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="24dp"
   android:height="24dp"
   android:viewportWidth="24"
   android:viewportHeight="24"
   android:tint="@color/white"
   android:autoMirrored="true">
  <path
    android:fillColor="@android:color/white"
    android:pathData="M6,19c0,1.1 0.9,2 2,2h8c1.1,0 2,-0.9 2,-2V7H6v12zM19,4h-3.5l-1,-
1h-5l-1,1H5v2h14V4z"/>
</vector>
```

Pasta drawable – ic_edit_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vector
   android:height="108dp"
   android:width="108dp"
   android:viewportHeight="108"
   android:viewportWidth="108"
   xmlns:android="http://schemas.android.com/apk/res/android">
   <path android:fillColor="#3DDC84"
        android:pathData="M0,0h108v108h-108z"/>
   <path android:fillColor="#00000000" android:pathData="M9,0L9,108"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
   <path android:fillColor="#00000000" android:pathData="M19,0L19,108"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
   <path android:fillColor="#00000000" android:pathData="M29,0L29,108"
```

```
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M39,0L39,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M49,0L49,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M59,0L59,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M69,0L69,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M79,0L79,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M89,0L89,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M99,0L99,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,9L108,9"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,19L108,19"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,29L108,29"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,39L108,39"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,49L108,49"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,59L108,59"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,69L108,69"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,79L108,79"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,89L108,89"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,99L108,99"
```

```
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,29L89,29"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,39L89,39"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,49L89,49"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,59L89,59"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,69L89,69"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,79L89,79"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M29,19L29,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M39,19L39,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M49,19L49,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M59,19L59,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M69,19L69,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M79,19L79,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
</vector>
```

Pasta drawable – ic_edit_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="@color/white"
```

```
    android:autoMirrored="true">
    <path
        android:fillColor="@android:color/white"
        android:pathData="M3,17.25V21h3.75L17.81,9.94l-3.75,-
3.75L3,17.25zM20.71,7.04c0.39,-0.39 0.39,-1.02 0,-1.41l-2.34,-2.34c-0.39,-0.39 -1.02,-0.39 -
1.41,0l-1.83,1.83 3.75,3.75 1.83,-1.83z"/>

</vector>
```

Pasta drawable – ic_history.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M13,3c-4.97,0 -9,4.03 -9,9L1,12l3.89,3.89 0.07,0.14L9,12L6,12c0,-
3.87 3.13,-7 7,-7s7,3.13 7,7 -3.13,7 -7,7c-1.93,0 -3.68,-0.79 -4.94,-2.06l-
1.42,1.42C8.27,19.99 10.51,21 13,21c4.97,0 9,-4.03 9,-9s-4.03,-9 -9,-9zM12,8v5l4.28,2.54
0.72,-1.21 -3.5,-2.08L13.5,8L12,8z"/>
</vector>
```

Pasta drawable – ic_hourglass.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
```

```
    android:pathData="M6,2v6h0.01L6,8.01 10,12l-4,4 0.01,0.01L6,16.01L6,22h12v-5.99h-
0.01L18,16l-4,-4 4,-3.99 -0.01,-0.01L18,8L18,2L6,2zM16,16.5L16,20L8,20v-3.5l4,-4
4,4zM12,11.5l-4,-4L8,4h8v3.5l-4,4z"/>
</vector>
```

Pasta drawable – ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="108dp"
   android:height="108dp"
   android:viewportWidth="108"
   android:viewportHeight="108">
   <path
      android:fillColor="#3DDC84"
      android:pathData="M0,0h108v108h-108z" />
   <path
      android:fillColor="#00000000"
      android:pathData="M9,0L9,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M19,0L19,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M29,0L29,108"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
   <path
      android:fillColor="#00000000"
      android:pathData="M39,0L39,108"
      android:strokeWidth="0.8"
```

```
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M49,0L49,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M59,0L59,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M69,0L69,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M79,0L79,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M89,0L89,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M99,0L99,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,9L108,9"
```

```
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,19L108,19"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,29L108,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,39L108,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,49L108,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,59L108,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,69L108,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
```

```
    android:pathData="M0,79L108,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,89L108,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,99L108,99"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,29L89,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,39L89,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,49L89,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,59L89,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
```

```
    android:fillColor="#00000000"
    android:pathData="M19,69L89,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,79L89,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M29,19L29,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M39,19L39,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M49,19L49,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M59,19L59,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M69,19L69,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
```

```
    <path
        android:fillColor="#00000000"
        android:pathData="M79,19L79,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
</vector>
```

Pasta drawable – ic_launcher_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-
1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
        <aapt:attr name="android:fillColor">
            <gradient
                android:endX="85.84757"
                android:endY="92.4963"
                android:startX="42.9492"
                android:startY="49.59793"
                android:type="linear">
                <item
                    android:color="#44000000"
                    android:offset="0.0" />
                <item
                    android:color="#00000000"
                    android:offset="1.0" />
            </gradient>
        </aapt:attr>
    </path>
    <path
        android:fillColor="#FFFFFF"
```

```
        android:fillType="nonZero"
        android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -
1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328
38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028
71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-
2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328
43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5
1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"
        android:strokeWidth="1"
        android:strokeColor="#00000000" />
</vector>
```

Pasta drawable – ic_login_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108"
    android:tint="#000000">
  <group android:scaleX="2.61"
      android:scaleY="2.61"
      android:translateX="22.68"
      android:translateY="22.68">
    <path
        android:fillColor="@android:color/white"
        android:pathData="M11,7L9.6,8.4l2.6,2.6H2v2h10.2l-2.6,2.6L11,17l5,-
5L11,7zM20,19h-8v2h8c1.1,0 2,-0.9 2,-2V5c0,-1.1 -0.9,-2 -2,-2h-8v2h8V19z"/>
  </group>
</vector>
```

Pasta drawable – ic_notification.xml

```
vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
```

```
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M12,22c1.1,0 2,-0.9 2,-2h-4c0,1.1 0.89,2 2,2zM18,16v-5c0,-3.07 -
1.64,-5.64 -4.5,-6.32L13.5,4c0,-0.83 -0.67,-1.5 -1.5,-1.5s-1.5,0.67 -1.5,1.5v0.68C7.63,5.36
6,7.92 6,11v5l-2,2v1h16v-1l-2,-2z"/>
</vector>
```

Pasta drawable – ic_welcome_image.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="158.189dp"
    android:height="198.05dp"
    android:viewportWidth="158.189"
    android:viewportHeight="198.05">
  <path
      android:pathData="M40.337,181.996 L1.579,195.811L1.579,87.683l38.758,-13.815Z"
      android:fillColor="#cfd1d2"/>
  <path
      android:pathData="M117.853,181.996l-38.758,13.815L79.095,87.683l38.758,-13.815Z"
      android:fillColor="#cfd1d2"/>
  <path
      android:pathData="M79.095,195.811l-38.758,-13.815L40.337,73.868L79.095,87.683Z"
      android:fillColor="#fff"/>
  <path
      android:pathData="M156.611,195.811l-38.758,-
13.815L117.853,73.868L156.611,87.683Z"
      android:fillColor="#fff"/>
  <path
      android:fillColor="#FF000000"
      android:pathData="M158.189,198.05l-40.337,-14.378L79.095,197.487l-38.758,-
13.815L0,198.05L0,86.569l40.337,-14.377 38.758,13.815 38.758,-13.815
```

```
40.337,14.377ZM40.337,180.317l38.758,13.815L117.852,180.317l37.18,13.252L155.032,88.
796l-37.18,-13.252 -38.758,13.815 -38.758,-13.815L3.158,88.796L3.158,193.573Z"/>
  <path
      android:pathData="M116.181,39.056a37.087,37.087 0,1 0,-69.787
17.111h0l30.1,57.705a2.933,2.933 0,0 0,5.2 0l29.411,-56.474a36.937,36.937 0,0 0,1.728 -
3.318l0.01,-0.019h0A36.939,36.939 0,0 0,116.181 39.056ZM79.094,58.516a19.3,19.3 0,1
1,19.3 -19.3A19.3,19.3 0,0 1,79.094 58.516Z"
      android:fillColor="#eb4f6f"/>
  <path
      android:fillColor="#FF000000"
      android:pathData="M79.092,117.033a4.5,4.5 0,0 1,-4 -2.425l-30.154,-57.81c-0.095,-
0.178 -0.188,-0.357 -0.28,-0.537l-0.87,-1.668h0.068a38.677,38.677 0,0 1,35.239 -
54.593,38.838 38.838,0 0,1 38.664,39.077l-1.579,-0.016 1.579,0.016a38.453,38.453 0,0 1,-
5.272 19.089l-29.392,56.438a4.5,4.5 0,0 1,-4 2.428ZM48.992,57.751
L77.892,113.151a1.355,1.355 0,0 0,2.4 0l29.45,-56.544a35.471,35.471 0,0 0,1.653 -
3.176l0.028,-0.056a35.511,35.511 0,1 0,-63.641
2.055l1.247,2.324ZM79.092,60.095a20.878,20.878 0,1 1,20.879 -20.878A20.9,20.9 0,0
1,79.095 60.095ZM79.092,21.495a17.722,17.722 0,1 0,17.721 17.722A17.742,17.742 0,0
0,79.095 21.495Z"/>
</vector>
```

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:id="@+id/nav_graph"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".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="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:navGraph="@navigation/navigation"
        />

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

Pasta layout – dropdown_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:padding="14dp"
    android:textColor="@color/black"
    android:textStyle="bold"
    android:textSize="16sp"
    xmlns:android="http://schemas.android.com/apk/res/android" />
```

Pasta layout – fragment_add_geofence.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=".AddGeofenceFragment">

    <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_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navigationIcon="@drawable/ic_back"
        app:title="Geofence"
        app:titleTextColor="@color/white" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout2"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropdow
nMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="13dp"
        android:hint="Tipo de Geofencing"
        app:layout_constraintBottom_toTopOf="@+id/geofence_name_textInputLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

        <AutoCompleteTextView
```

```
        android:id="@+id/autoCompleteTextView3"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:inputType="none" />
</com.google.android.material.textfield.TextInputLayout>

<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="Nome"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.322">


    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/geofence_name_et"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<TextView
    android:id="@+id/slider_value_textView"
```

```
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="28dp"
    android:text="1 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_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="@+id/slider" />

<com.google.android.material.slider.Slider
    android:id="@+id/slider"
    updateSliderValueTextView="@{sliderValueTextView}"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"
    android:stepSize="1.0"
    android:value="1.0"
    android:valueFrom="1.0"
    android:valueTo="1000.0"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.639" />

<TextView
    android:id="@+id/geofence_radius_desc"
    android:layout_width="0dp"
```

```
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Escolha o raio da geofence"
        android:textAlignment="center"
        app:layout_constraintEnd_toEndOf="@+id/slider"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/slider"
        app:layout_constraintTop_toBottomOf="@+id/slider" />

    <TextView
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="32dp"
        android:text="PROXIMO"
        android:textColor="@color/view_state_background_color"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />





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

Pasta layout – fragment_create_account.xml

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

```
android:layout_height="match_parent"
xmlns:app="http://schemas.android.com/apk/res-auto"
tools:context=".CreateAccountFragment">

<EditText
    android:id="@+id/createname"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="160dp"
    android:layout_marginEnd="24dp"
    android:autofillHints="Name"
    android:hint="Nome"
    android:imeActionLabel="Sign in"
    android:imeOptions="actionDone"
    android:inputType="textEmailAddress"
    android:selectAllOnFocus="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/createemail"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="24dp"
    android:autofillHints="Email"
    android:hint="Email"
    android:imeActionLabel="Sign in"
    android:imeOptions="actionDone"
    android:inputType="textEmailAddress"
    android:selectAllOnFocus="true"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/createname" />

    <EditText
        android:id="@+id/createpassword"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="24dp"
        android:autofillHints="Password"
        android:hint="Senha"
        android:imeActionLabel="Sign in"
        android:imeOptions="actionDone"
        android:inputType="textPassword"
        android:selectAllOnFocus="true"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/createemail" />

    <Button
        android:id="@+id/createlogin"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="24dp"
        android:text="Criar Conta"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/createpassword" />

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

Pasta layout – fragment_edit_cultura.xml

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

    <androidx.constraintlayout.widget.ConstraintLayout
        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_toBottomOf="@+id/toolbar">

        <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_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:navigationIcon="@drawable/ic_back"
            app:title="Cultura"
            app:titleTextColor="@color/white" />

        <com.google.android.material.textfield.TextInputLayout
```

```
    android:id="@+id/tipo_cultura"
    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="Tipo de Cultura"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/toolbar"
    app:layout_constraintVertical_bias="0.12">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/tipo_cultura_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/area_plantada"
    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="Area Plantada"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/tipo_cultura"
    app:layout_constraintVertical_bias="0.16">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/area_plantada_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/quantida_sementes"
    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="Quantidade de Semente"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/area_plantada"
    app:layout_constraintVertical_bias="0.2">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/quantida_sementes_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/data_prevista"
    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="Data Prevista Para Colheita"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/quantida_sementes"
    app:layout_constraintVertical_bias="0.24">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/data_prevista_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="date"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<TextView
    android:id="@+id/Salvar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="32dp"
    android:layout_marginBottom="32dp"
    android:text="SALVAR"
    android:textColor="@color/view_state_background_color"
    android:textSize="18sp"
    android:textStyle="bold"
```

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


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

Pasta layout – fragment_edit_gado_de_corte.xml

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

    <androidx.constraintlayout.widget.ConstraintLayout
        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_toBottomOf="@+id/toolbar">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="3dp"
            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="Gado de Corte"
    app:titleTextColor="@color/white" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/quantidade_de_animais"
    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="Quantidade de Animais"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/toolbar"
    app:layout_constraintVertical_bias="0.12">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/quantidade_de_animais_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tipo_alimentaçao"
    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="Tipo de Alimento"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/quantidade_de_animais"
    app:layout_constraintVertical_bias="0.16">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/tipo_alimentaçao_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/alimento_por_dia"
    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="Quntidade de Alimento Por Dia"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/tipo_alimentaçao"
    app:layout_constraintVertical_bias="0.2">
```

```
        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/alimento_por_dia_imput"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLength="20" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout2"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropdow
nMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="13dp"
        android:hint="Finalidade de Produção"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/alimento_por_dia"
        app:layout_constraintVertical_bias="0.24">
        <AutoCompleteTextView
            android:id="@+id/autoCompleteTextView3"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:inputType="none" />
    </com.google.android.material.textfield.TextInputLayout>

    <TextView
        android:id="@+id/Salvar"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="32dp"
        android:text="Salvar"
        android:textColor="@color/view_state_background_color"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</FrameLayout>
```

Pasta layout – fragment_edit_gado_de_leite.xml

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

    <androidx.constraintlayout.widget.ConstraintLayout
        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_toBottomOf="@+id/toolbar">

        <androidx.appcompat.widget.Toolbar
```

```
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="3dp"
    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="Gado de Leite"
    app:titleTextColor="@color/white" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/quantidade_de_animais"
    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="Quantidade de Animais"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/toolbar"
    app:layout_constraintVertical_bias="0.12">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/quantidade_de_animais_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:inputType="number"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tipo_alimentaçao"
    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="Tipo de Alimento"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/quantidade_de_animais"
    app:layout_constraintVertical_bias="0.16">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/tipo_alimentaçao_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/alimento_por_dia"
    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="Quntidade de Alimento Por Dia"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/tipo_alimentaçao"
    app:layout_constraintVertical_bias="0.2">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/alimento_por_dia_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/quantidade_de_leite"
    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="Quntidade de Leite Por Dia"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/alimento_por_dia"
    app:layout_constraintVertical_bias="0.24">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/quantidade_de_leite_imput"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLength="20" />
        </com.google.android.material.textfield.TextInputLayout>

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

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

Pasta layout – fragment_edit_galinheiro.xml

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

    <androidx.constraintlayout.widget.ConstraintLayout
        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_toBottomOf="@+id/toolbar">

<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="3dp"
    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="Galinheiro"
    app:titleTextColor="@color/white" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/quantidade_de_animais"
    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="Quantidade de Animais"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
```

```
    app:layout_constraintTop_toTopOf="@+id/toolbar"
    app:layout_constraintVertical_bias="0.12">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/quantidade_de_animais_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tipo_alimentaçao"
    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="Tipo de Alimento"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/quantidade_de_animais"
    app:layout_constraintVertical_bias="0.16">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/tipo_alimentaçao_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
```

```
        android:id="@+id/alimento_por_dia"
        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="Quntidade de Alimento Por Dia"
        app:counterEnabled="true"
        app:counterMaxLength="20"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/tipo_alimentaçao"
        app:layout_constraintVertical_bias="0.2">
        <com.google.android.material.textfield.TextInputEditText
          android:id="@+id/alimento_por_dia_imput"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:maxLength="20" />
      </com.google.android.material.textfield.TextInputLayout>

      <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout2"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropdow
nMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="13dp"
        android:hint="Finalidade de Produção"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@+id/alimento_por_dia"
            app:layout_constraintVertical_bias="0.24">
            <AutoCompleteTextView
                android:id="@+id/autoCompleteTextView3"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:inputType="none" />
        </com.google.android.material.textfield.TextInputLayout>

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

    </androidx.constraintlayout.widget.ConstraintLayout>

</FrameLayout>
```

Pasta layout – fragment_edit_geofence.xml

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

```
android:layout_width="match_parent"
android:layout_height="match_parent"
xmlns:app="http://schemas.android.com/apk/res-auto"
tools:context=".EditGeofenceFragment">


<androidx.constraintlayout.widget.ConstraintLayout
    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_toBottomOf="@+id/toolbar">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="3dp"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintBottom_toTopOf="@+id/dados_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navigationIcon="@drawable/ic_back"
        app:title="Nome da Geofence"
        app:titleTextColor="@color/white" />

    <TextView
        android:id="@+id/dados_text"
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"
        android:layout_marginStart="10dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="81dp"
        android:lineSpacingExtra="10sp"
        android:text=""
        android:textSize="24sp"
        android:textStyle="normal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar" />


    </androidx.constraintlayout.widget.ConstraintLayout>


</FrameLayout>
```

Pasta layout – fragment_edit_mangueiro.xml

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

    <androidx.constraintlayout.widget.ConstraintLayout
        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_toBottomOf="@+id/toolbar">

<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="3dp"
    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="Manqueiro"
    app:titleTextColor="@color/white" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/quantidade_de_animais"
    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="Quantidade de Animais"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/toolbar"
    app:layout_constraintVertical_bias="0.12">
```

```
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/quantidade_de_animais_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tipo_alimentaçao"
    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="Tipo de Alimento"
    app:counterEnabled="true"
    app:counterMaxLength="20"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/quantidade_de_animais"
    app:layout_constraintVertical_bias="0.16">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/tipo_alimentaçao_imput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="20" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/alimento_por_dia"
    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="Quntidade de Alimento Por Dia"
        app:counterEnabled="true"
        app:counterMaxLength="20"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/tipo_alimentaçao"
        app:layout_constraintVertical_bias="0.2">
        <com.google.android.material.textfield.TextInputEditText
          android:id="@+id/alimento_por_dia_imput"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:maxLength="20" />
      </com.google.android.material.textfield.TextInputLayout>

      <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout2"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropdow
nMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="13dp"
        android:hint="Finalidade de Produção"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
```

```
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@+id/alimento_por_dia"
            app:layout_constraintVertical_bias="0.24">
            <AutoCompleteTextView
                android:id="@+id/autoCompleteTextView3"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:inputType="none" />
        </com.google.android.material.textfield.TextInputLayout>

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

    </androidx.constraintlayout.widget.ConstraintLayout>

</FrameLayout>
```

Pasta layout – fragment_geofences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
tools:context=".GeofencesFragment">

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="55dp"
        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"
        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" />


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

```
</FrameLayout>
```

Pasta layout – fragment_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".LoginFragment"
    android:id="@+id/container"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp">

    <ImageView
        android:id="@+id/welcome_imageView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="78dp"
        android:layout_marginEnd="24dp"
        android:layout_marginBottom="100dp"
        android:src="@drawable/ic_login_foreground"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints" />

    <EditText
        android:id="@+id/logusername"
        android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="24dp"
    android:autofillHints="Email"
    android:hint="Email"
    android:imeActionLabel="Sign in"
    android:imeOptions="actionDone"
    android:inputType="textEmailAddress"
    android:selectAllOnFocus="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/welcome_imageView" />

<EditText
    android:id="@+id/logpassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="24dp"
    android:autofillHints="Password"
    android:hint="Senha"
    android:imeActionLabel="Sign in"
    android:imeOptions="actionDone"
    android:inputType="textPassword"
    android:selectAllOnFocus="true"

    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/logusername" />

<CheckBox
    android:id="@+id/mostrar_senha"
```

```
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="200dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="24dp"
    android:text="Mostrar senha"
    android:selectAllOnFocus="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/logpassword"
    />

<Button
    android:id="@+id/login"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="24dp"
    android:text="Entrar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/mostrar_senha" />

<Button
    android:id="@+id/registrar"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="78dp"
    android:layout_marginEnd="24dp"
    android:text="Registrar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
```

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

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

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 xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/map"
        android:name="com.google.android.GSM.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MapsFragment" />

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

```
    app:tint="@color/white" />

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

<TextView
    android:id="@+id/infoMessage_textView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"
    android:text="Pressione longamente no mapa para adicionar uma cerca geográfica"
    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:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".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="Continue"
        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="Para usar este aplicativo, primeiro precisamos solicitar sua permissão de
localização. Caso contrário, não seremos capazes de lhe fornecer nosso serviço. Se você
concorda e deseja continuar, toque no botão abaixo."
        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_required_textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:text="Permissão necessária"
        android:textAlignment="center"
        android:textColor="#eb4f6f"
        android:textSize="26sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/permission_description_textView"
        app:layout_constraintEnd_toEndOf="@+id/permission_description_textView"
        app:layout_constraintStart_toStartOf="@+id/permission_description_textView" />

    <ImageView
```

```
        android:id="@+id/welcome_imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="100dp"
        android:src="@drawable/ic_welcome_image"
        app:layout_constraintBottom_toTopOf="@+id/permission_required_textView"
        app:layout_constraintEnd_toEndOf="@+id/permission_required_textView"
        app:layout_constraintStart_toStartOf="@+id/permission_required_textView" />



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

Pasta layout – res_item_live.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"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">


    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:cardCornerRadius="2dp"
        app:cardElevation="8dp"
        app:cardPreventCornerOverlap="false"
        app:cardUseCompatPadding="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
```

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">


    <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/thumbnail"
        android:layout_width="120dp"
        android:layout_height="120dp"
        android:scaleType="centerCrop"
        android:layout_margin="0dp"
        android:adjustViewBounds="true"
        android:padding="0dp"
        app:layout_constraintBottom_toBottomOf="@id/white_background"
        app:layout_constraintStart_toStartOf="@id/white_background"
        app:layout_constraintTop_toTopOf="@id/white_background" />


    <LinearLayout
        android:id="@+id/container1"
        android:layout_width="0dp"
        android:layout_height="120dp"
```

```
android:orientation="vertical"

app:layout_constraintBottom_toBottomOf="@id/white_background"
app:layout_constraintStart_toEndOf="@+id/thumbnail"
app:layout_constraintLeft_toLeftOf="parent">

<TextView
    android:id="@+id/nome"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Nome da Geofence"
    android:textColor="#000"
    android:textSize="19sp" />

<TextView
    android:id="@+id/raio"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:text="Raio"
    android:textSize="15sp" />

<TextView
    android:id="@+id/latitude"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:text="Latitude"
    android:textSize="15sp" />

<TextView
    android:id="@+id/longitude"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
            android:layout_marginTop="10dp"
            android:text="longitude"
            android:textSize="15sp" />

        </LinearLayout>

    </androidx.constraintlayout.widget.ConstraintLayout>

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

Pasta navigation – navigation.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/navigation"
  app:startDestination="@+id/loginFragment">
  <fragment
    android:id="@+id/loginFragment"
    android:name="com.example.myapplication.LoginFragment"
    android:label="fragment_login"
    tools:layout="@layout/fragment_login" >
    <action
      android:id="@+id/action_loginFragment_to_createAccountFragment"
      app:destination="@id/createAccountFragment" />
    <action
      android:id="@+id/action_loginFragment_to_permissionFragment"
      app:destination="@id/permissionFragment" />
    <action
      android:id="@+id/action_loginFragment_to_mapsFragment"
      app:destination="@id/mapsFragment"
      app:popUpTo="@+id/navigation"
      app:popUpToInclusive="true"/>
```

```
</fragment>
<fragment
    android:id="@+id/createAccountFragment"
    android:name="com.example.myapplication.CreateAccountFragment"
    android:label="fragment_create_account"
    tools:layout="@layout/fragment_create_account" />
<fragment
    android:id="@+id/permissionFragment"
    android:name="com.example.myapplication.PermissionFragment"
    android:label="fragment_permission"
    tools:layout="@layout/fragment_permission" >
    <action
        android:id="@+id/action_permissionFragment_to_mapsFragment"
        app:destination="@id/mapsFragment" />
</fragment>
<fragment
    android:id="@+id/mapsFragment"
    android:name="com.example.myapplication.MapsFragment"
    android:label="fragment_maps"
    tools:layout="@layout/fragment_maps" >
    <action
        android:id="@+id/action_mapsFragment_to_addGeofenceFragment"
        app:destination="@id/addGeofenceFragment" />
    <action
        android:id="@+id/action_mapsFragment_to_geofencesFragment"
        app:destination="@id/geofencesFragment"
        app:popUpTo="@+id/navigation"
        app:popUpToInclusive="true"/>
    <argument
        android:name="nomeGeofence"
        app:argType="string"
        android:defaultValue="Defalt" />
    <argument
        android:name="raioGeofence"
```

```
        app:argType="float"
        android:defaultValue="1" />
    <argument
        android:name="retorno"
        app:argType="string"
        android:defaultValue="null" />
    <argument
        android:name="tipoGeofence"
        app:argType="string"
        android:defaultValue="Defalt" />
    <argument
        android:name="dado1"
        app:argType="string"
        android:defaultValue="Defalt" />
    <argument
        android:name="dado2"
        app:argType="string"
        android:defaultValue="Defalt" />
    <argument
        android:name="dado3"
        app:argType="string"
        android:defaultValue="Defalt" />
    <argument
        android:name="dado4"
        app:argType="string"
        android:defaultValue="Defalt" />
</fragment>
<fragment
    android:id="@+id/addGeofenceFragment"
    android:name="com.example.myapplication.AddGeofenceFragment"
    android:label="fragment_add_geofence"
    tools:layout="@layout/fragment_add_geofence" >
    <action
        android:id="@+id/action_addGeofenceFragment_to_editMangueiroFragment"
```

```
        app:destination="@id/editMangueiroFragment" />
    <action
        android:id="@+id/action_addGeofenceFragment_to_editCulturaFragment"
        app:destination="@id/editCulturaFragment" />
    <action
        android:id="@+id/action_addGeofenceFragment_to_editGadoDeCorteFragment"
        app:destination="@id/editGadoDeCorteFragment" />
    <action
        android:id="@+id/action_addGeofenceFragment_to_editGadoDeLeiteFragment"
        app:destination="@id/editGadoDeLeiteFragment" />
    <action
        android:id="@+id/action_addGeofenceFragment_to_editGalinheiroFragment"
        app:destination="@id/editGalinheiroFragment" />
</fragment>
<fragment
    android:id="@+id/geofencesFragment"
    android:name="com.example.myapplication.GeofencesFragment"
    android:label="fragment_geofences"
    tools:layout="@layout/fragment_geofences" >
    <action
        android:id="@+id/action_geofencesFragment_to_editGeofenceFragment"
        app:destination="@id/editGeofenceFragment" />
    <action
        android:id="@+id/action_geofencesFragment_to_mapsFragment"
        app:destination="@id/mapsFragment"
        app:popUpTo="@+id/navigation"
        app:popUpToInclusive="true"/>
</fragment>
<fragment
    android:id="@+id/editGeofenceFragment"
    android:name="com.example.myapplication.EditGeofenceFragment"
    android:label="fragment_edit_geofence"
    tools:layout="@layout/fragment_edit_geofence" >
    <argument
```

```
        android:name="nome"
        app:argType="string"
        android:defaultValue="Defalt" />
    </fragment>
    <fragment
        android:id="@+id/editCulturaFragment"
        android:name="com.example.myapplication.EditCulturaFragment"
        android:label="fragment_edit_cultura"
        tools:layout="@layout/fragment_edit_cultura" >
        <argument
            android:name="nome"
            app:argType="string"
            android:defaultValue="Defalt" />
        <action
            android:id="@+id/action_editCulturaFragment_to_mapsFragment"
            app:destination="@id/mapsFragment" />
        <argument
            android:name="raio"
            app:argType="string"
            android:defaultValue="0" />
    </fragment>
    <fragment
        android:id="@+id/editGadoDeCorteFragment"
        android:name="com.example.myapplication.EditGadoDeCorteFragment"
        android:label="fragment_edit_gado_de_corte"
        tools:layout="@layout/fragment_edit_gado_de_corte" >
        <argument
            android:name="nome"
            app:argType="string"
            android:defaultValue="Defalt" />
        <action
            android:id="@+id/action_editGadoDeCorteFragment_to_mapsFragment"
            app:destination="@id/mapsFragment" />
        <argument
```

```
        android:name="raio"
        app:argType="string"
        android:defaultValue="0" />
    </fragment>
    <fragment
        android:id="@+id/editGadoDeLeiteFragment"
        android:name="com.example.myapplication.EditGadoDeLeiteFragment"
        android:label="fragment_edit_gado_de_leite"
        tools:layout="@layout/fragment_edit_gado_de_leite" >
        <argument
            android:name="nome"
            app:argType="string"
            android:defaultValue="Defalt" />
        <action
            android:id="@+id/action_editGadoDeLeiteFragment_to_mapsFragment"
            app:destination="@id/mapsFragment" />
        <argument
            android:name="raio"
            app:argType="string"
            android:defaultValue="0" />
    </fragment>
    <fragment
        android:id="@+id/editGalinheiroFragment"
        android:name="com.example.myapplication.EditGalinheiroFragment"
        android:label="fragment_edit_galinheiro"
        tools:layout="@layout/fragment_edit_galinheiro" >
        <argument
            android:name="nome"
            app:argType="string"
            android:defaultValue="Defalt" />
        <action
            android:id="@+id/action_editGalinheiroFragment_to_mapsFragment"
            app:destination="@id/mapsFragment" />
        <argument
```

```
            android:name="raio"
            app:argType="string"
            android:defaultValue="0" />
        </fragment>
        <fragment
            android:id="@+id/editMangueiroFragment"
            android:name="com.example.myapplication.EditMangueiroFragment"
            android:label="fragment_edit_mangueiro"
            tools:layout="@layout/fragment_edit_mangueiro" >
            <argument
                android:name="nome"
                app:argType="string"
                android:defaultValue="Defalt" />
            <action
                android:id="@+id/action_editMangueiroFragment_to_mapsFragment"
                app:destination="@id/mapsFragment" />
            <argument
                android:name="raio"
                app:argType="string"
                android:defaultValue="0" />
        </fragment>

    </navigation>
```

Pasta values – colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="blue_500">#4285F4</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF6200EE</color>
    <color name="blue_700">#0059c1</color>
    <color name="blue_transparent">#304285F4</color>
```

```
    <color name="red">#eb4f6f</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="gray">#CCCCCC</color>
</resources>
```

Pasta values – google_maps_api.xml

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:


https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyT
ype=CLIENT_SIDE_ANDROID&r=0A:4D:7E:68:1A:BA:89:45:F5:41:7D:0A:93:6E:92:B7:
93:3E:56:37%3Bcom.example.myapplication

    You can also add your credentials to an existing key, using these values:

    Package name:
    com.example.myapplication

    SHA-1 certificate fingerprint:
    0A:4D:7E:68:1A:BA:89:45:F5:41:7D:0A:93:6E:92:B7:93:3E:56:37

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
```

```
    <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">AIzaSyDWAtL8guZnNsg3P17OflXSbzM7R0rG5BI</string>
</resources>
```

Pasta values – strings.xml

```
resources>
    <string name="app_name">My Application</string>
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
</resources>
```

Pasta themes– themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.MyApplication"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Pasta themes– themes.xml (night)

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
```

```
    <style name="Theme.MyApplication"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_200</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/black</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```