PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA - ENGENHARIA DE COMPUTAÇÃO

UTILIZAÇÃO DE *GEOFENCING* PARA APRIMORAR A SEGURANÇA EM UMA CONCESSIONÁRIA DE VEÍCULOS

ALEX LOURENÇO DE CARVALHO

GOIÂNIA
2022

ALEX LOURENÇO DE CARVALHO

UTILIZAÇÃO DE *GEOFENCING* PARA APRIMORAR A SEGURANÇA EM UMA CONCESSIONÁRIA DE VEÍCULOS

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

GOIÂNIA
2022

ALEX LOURENÇO DE CARVALHO

UTILIZAÇÃO DE *GEOFENCING* PARA APRIMORAR A SEGURANÇA EM UMA CONCESSIONÁRIA DE VEÍCULOS

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 09/12/2022.

_____.
Prof. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenador de Trabalho de Conclusão de Curso

Banca examinadora:

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

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

_____.
Membro 2: Prof. Dr. Fabio Barbosa Rodrigues

Goiânia
2022

SUMÁRIO

AGRADECIMENTOS

Chegando ao final de mais uma etapa, o que resta são somente os agradecimentos pela ajuda de todos e pela experiência passada a mim durante estes cinco anos de curso.

Agradeço primeiramente aos meus pais, avós, tios e irmã, por sempre estarem comigo me incentivando e dando forças para continuar, mesmo em momentos difíceis.

Agradeço a Deus por me dar esta vida e essas oportunidades que tive, pois sem ele ninguém poderia existir.

Agradeço ao professor Marcelo Antônio, por sempre me apoiar desde a primeira disciplina em que me matriculei e que estava a ministrar, e me incentivar a estudar e aprender mais sobre o que a faculdade tem a oferecer.

Agradeço também ao professor Carlos Alexandre e Fábio por aceitarem participar desta banca, e ajudar a completar mais este ciclo.

E por último e não menos importante agradeço aos meus colegas de curso, que me ajudaram a entender o que eu tinha dificuldade e a estudar mais para também poder retribuir a ajuda.

Sem tais pessoas eu não estaria neste estágio da minha vida, o mínimo que posso fazer agora é agradecer a todos que me ajudaram a chegar aonde estou. Muito obrigado!

RESUMO

Neste trabalho são descritas as aplicações, linguagens de programação e conceitos, aprofundando principalmente na aplicação utilizada no projeto junto às definições das formas de localização, utilizando da tecnologia de *Geofencing* em um mapa, a fim de poder delimitar uma área onde o usuário responsável poderá controlar e monitorar todos os movimentos e ocorrências com um veículo, em se tratando de um ambiente de uma concessionária veicular ou externo, facilitando bastante o trabalho dos vendedores e aumentando a quantidade de opções que eles possuem ao atender um cliente.

Palavras-Chave: Aplicação, Concessionária, Monitoramento, Veículo.

ABSTRACT

In this work the applications, programming languages and concepts are described, deepening mainly in the application used in the project together with the definitions of the ways of localization, using the technology of Geofencing in a map, in order to be able to delimit an area where the person in charge can control and monitor all movements and occurrences with a vehicle, in the case of a vehicle dealership or external environment, greatly facilitating the work of salespeople and increasing the number of options they have when serving a customer.
Keywords: Application, Vehicle Dealership, Monitoring, Vehicle.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Dr.	Doutor
Ma.	Mestra
Me.	Mestre
M.E.E	Mestre em Engenharia Elétrica
Prof.	Professor
PUC	Pontifícia Universidade Católica

LISTA DE SIGLAS

API	*Application Programming Interface*
APP	*Application*
DBMS	*Database Management Systems*
GPS	*Global Positioning System*
HAL	*Hardware abstraction Layer*
IBM	*International Business Machines*
IDE	*Integrated Development Environment*
RFID	*Radio Frequency Identification*
SO	Sistema Operacional
SQL	*Structured Query Language*
SSL	*Secure Sockets Layer*
LID	Linha Internacional de Data

1 INTRODUÇÃO

A ideia do projeto é de se utilizar da tecnologia do *geofencing* para facilitar e promover mais liberdade ao cliente na hora de testar os veículos, onde a mobilidade é de suma importância para que o usuário conheça melhor o veículo em questão, ou para fim de prevenir roubos, sinalizando a saída indevida de um ambiente predeterminado pelo usuário do estabelecimento autorizado para esse fim.

Onde, para fazer o *test drive* do veículo normalmente um dos vendedores ou outro funcionário necessita ir junto com o cliente para evitar furto ou acompanhar o caminho, que será previamente combinado com o cliente e que ele irá seguir rigorosamente.

Com esta aplicação as concessionárias se beneficiarão, pois não será mais preciso de alocar um dos funcionários para fazer o acompanhamento e nem de ter o risco de acabar tendo o veículo roubado durante o tempo do *test drive*, agora o vendedor conseguindo atuar durante o teste. É possível monitorar em tempo real e definir uma região de funcionamento do veículo, junto com o horário e informações do cliente e do bem. Este monitoramento se dará através de um aplicativo mobile, onde se pode definir como um desenho ou com linhas o local ou as ruas que poderão ser utilizadas pelo cliente, pois também haverá um dispositivo *android* acoplado ao veículo o qual enviará sinais de GPS (*Global Positioning System*, ou Sistema de Posicionamento Global) em tempo real para ser monitorado no estabelecimento. Sendo assim, qualquer desvio de rota será notificado à loja, que poderá tomar providência mais rapidamente, como por exemplo se for um *Tesla*, ou similar, poderá ser implementada uma função para o desligamento do veículo em caso de tentativa de furto.

Esta aplicação beneficiará, tanto os vendedores, quanto os próprios clientes, porque os vendedores poderão descansar ou atender mais de um cliente por vez. e os clientes terão mais liberdade para testar os veículos que gostariam de adquirir, sendo assim benefício para ambos os envolvidos e não custando muito para adquirir tal aplicativo.

Foi relevante estudar esse tema porque, baseado nessas considerações, fazer o uso do *Geofencing* para esta finalidade se mostrar bastante notável e pertinente, já que o primeiro contato do cliente com o produto é o que define uma empresa. Seu uso se mostra no auxílio para manter o veículo seguro, evitando roubos e dando uma sensação de liberdade ao cliente.

Diante deste contexto, esta pesquisa respondeu a seguinte questão:

É possível aprimorar a forma em que os clientes testam os automóveis, e ainda retirar a necessidade de alocar um vendedor, utilizando automação computacional?

1.1 Objetivos

Os objetivos podem ser divididos em objetivo geral e específicos.

1.1.1 Objetivo geral

- Desenvolver uma aplicação que, com a utilização de *Geofencing,* torne possível rastrear e administrar um veículo enquanto é efetuado o teste do veículo.

1.1.2 Objetivos específicos

- Ampliar o conhecimento e estudo do *Geofencing*.
- Usar o conteúdo de *Geofencing* de forma a garantir uma localização precisa.
- Rastrear detalhadamente um veículo que esteja devidamente preparado para tal.
- Testar e validar o sistema envolvido.
- Aplicar o conhecimento da linguagem *Kotlin*.

2 Referencial Teórico

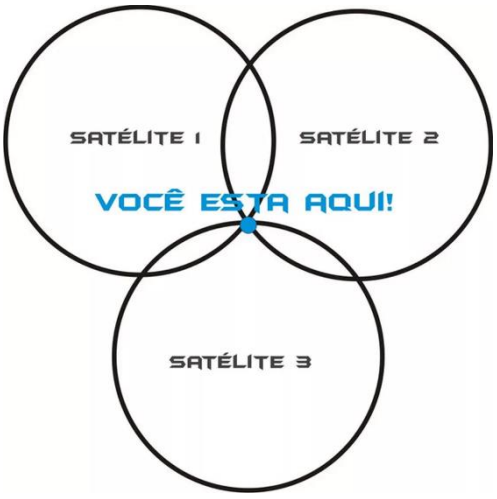
São tratados neste capítulo os fundamentos teóricos utilizados para a criação deste trabalho, com a finalidade de deixar claro os conceitos utilizados no desenvolvimento do aplicativo para *smartphones* e as formas de implementar o banco de dados.

2.1 *Global Positioning System* (GPS)

O GPS é formado por uma constelação de 24 satélites operacionais e oito “inativos” para casos de falhas, distribuídos em seis planos orbitais inclinados em 55 graus em relação ao plano horizontal do equador, cada órbita contendo pelo menos quatro satélites em operação. Cada satélite orbita a terra quase duas vezes por dia a uma altitude de aproximadamente 20.200 km. (KHOURY, 2011)

Para o cálculo de sua posição no mapa é utilizada a técnica de trilateração. Esta técnica utiliza os dados de um satélite para conseguir a localização de um ponto em uma grande área, a inserção de dados de mais outro satélite aumenta significativamente a precisão de tal ponto. Em outras palavras, quanto mais satélites forem utilizados para a coleta de tais dados, mais preciso será a localização fornecida. Todos os dispositivos de GPS utilizam pelo menos três satélites para o cálculo de sua posição. (FIODEVIDA, 2021)

Figura 1 - Trilateração

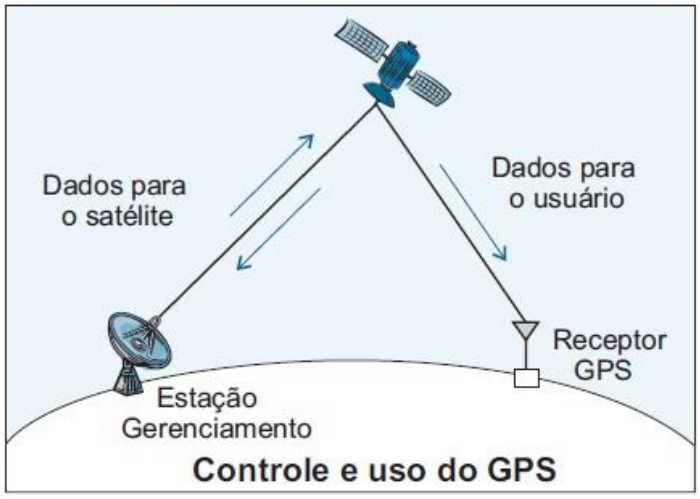


Fonte: TechTudo, 2022

O cálculo da posição também pode ser feito utilizando da técnica de triangulação, nesta técnica são utilizados de três satélites que enviam sinal ao receptor, calculando o tempo em que o sinal demora para chegar até ele, descobrindo sua posição terrestre além de também poder saber a altura em relação ao nível do mar, mas para isso seria necessário de um quarto satélite, para que esses dados sejam lidos por um aplicativo qualquer. Eles são transformados nas coordenadas geográficas Latitude e Longitude, que são as variáveis utilizadas para definir o local onde se encontra em um mapa específico do globo terrestre. (SÓ FISICA, 2022)

Tais satélites possuem um relógio com uma precisão enorme, chegando na marca de nanossegundos, este tempo é marcado e enviado junto ao sinal para o receptor efetuar o cálculo. Esse cálculo leva em conta a velocidade de propagação desse sinal. O receptor se situa no meio dessa intersecção, onde é possível identificar exatamente onde o aparelho está na Terra como na figura 2. Este envio de sinal acontece de forma constante na velocidade de 300 mil quilômetros/s que é a mesma que a velocidade da luz. (SÓ FISICA, 2022)

Figura 2 – Envio de dados de satélites ao receptor



Fonte: TechTudo, 2022

O recurso é bastante utilizado por aplicativos que envolvem mobilidade urbana (*99 Pop, Uber, Waze* etc.), aplicativos de entrega de comidas (como *o iFood*), sendo utilizado até em jogos virtuais como o *Pokémon GO.* (TIAGO, 2021)

Este tipo de tecnologia aparenta sempre estar em crescimento, tanto em aplicativos quanto em produtos ou em veículos (ainda em desenvolvimento pela *Tesla*

de *Ellon Musk*), onde no momento que se está pode ser utilizado por quase qualquer dispositivo por estar bastante acessível em comparação a tempos anteriores.

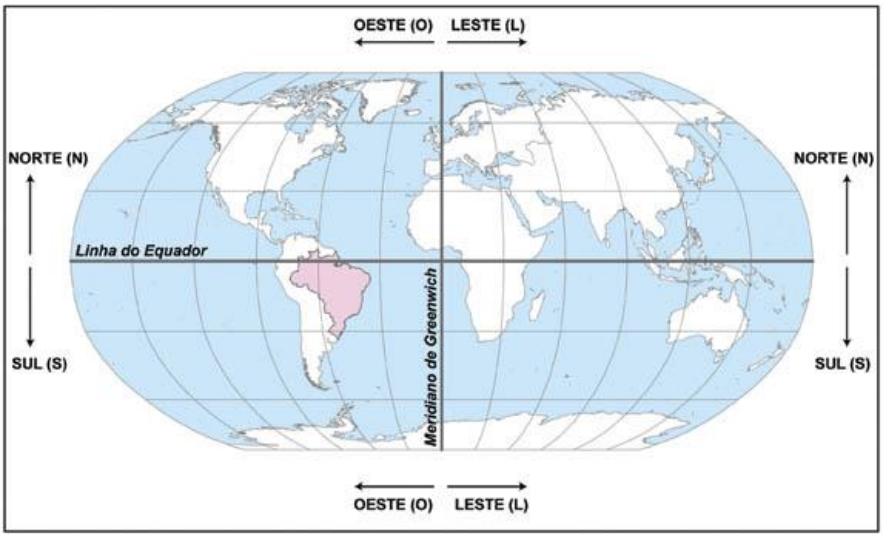
2.1.2 Coordenadas geográficas

Sendo a Terra aproximadamente uma esfera, a reta que segue pelo centro dela e marca a sua rotação é denominada de eixo polar. Os pontos extremos onde é feita a intersecção dessa reta com a superfície terrestre foram definidos como polo Sul e polo Norte. (LIMA ,2013)

Foi passado também um plano perpendicular ao eixo polar dividindo a terra em duas partes, tendo tal eixo como centro e referência. Essa circunferência é chamada de linha do equador e as partes divididas chamadas de hemisfério norte e hemisfério sul, cada uma contendo seus respectivos polos. (LIMA, 2013)

Há também outro plano dividindo o espaço terrestre só que desta vez com os polos Norte e Sul definindo dois extremos da circunferência. Tal plano se chama Meridiano de *Greenwich* e as duas partes divididas tendo como nomes hemisfério leste e hemisfério oeste, o nome se dá porque a linha desta circunferência passa pelo observatório de *Greenwich*, a sudeste de Londres. Tendo também o antimeridiano chamado de Linha Internacional de Data (LID), que se difere em 180º de distância do meridiano. (LIMA, 2013)

Figura 3 – Divisão do globo planificada



Fonte: IBGE, 2022

O sistema de localização utiliza das variáveis Latitude e Longitude, que são os produtos das duas divisões do globo citadas anteriormente. Com elas é possível a definição de um ponto em um mapa qualquer que utilize o mesmo sistema.

A Latitude é definida ao longo de Meridiano de *Greenwich* pela distância ao Equador percorrida em torno do Meridiano em graus que podem variar de 0º a 90º tanto em direção ao polo norte quanto em direção ao polo sul. Já Longitude é definida ao longo do Equador medindo a distância ao Meridiano também em graus, podendo variar de 0º a 180º, sendo direcionado, tanto para o Leste, quanto para o Oeste. Como apresentado na figura 3 (VELA, 2006)

2.2 *Geofencing*

Geofencing é a forma de usar o Sistema Posicionamento Global (GPS) onde o administrador pode configurar disparadores que enviam mensagens de texto, alerta de *e-mail* ou no telefone quando algum dispositivo com estas funções entra ou sai desta área demarcada. Em outras palavras, o administrador pode separar uma área específica e se comunicar com dispositivos que estejam no alcance. (CHAMBERLAIN, 2016)

Figura 4 - Uso do geofencing

Fonte: Autoria própria (Modificada), 2022

Embora as soluções, tanto de *hardware,* quanto de *software,* já existam há décadas, os sistemas antigos eram bastante limitados somente àqueles dispostos a investir grandes quantias em *hardware* personalizado para casos específicos. O *geofencing* foi primeiramente implementado na pecuária, em que se equipava

unidades de GPS no rebanho e se este se movesse para fora do limite imposto, quem possuía este rebanho era alertado. Foram implementados sistemas parecidos mais tarde, como por exemplo em empresas, onde eram monitorados os seus veículos. No momento que saíssem da rota era alertado ao responsável. (MAIS GEEK, 2022)

Com o crescente aumento de dispositivos eletrônicos que fazem utilização do GPS, o *geofencing* está se tornando uma prática padrão. No momento em que se pode definir uma área, as empresas enxergam oportunidades do que se pode fazer a respeito do que está ao alcance, principalmente nas áreas de *marketing* e mídias sociais. (WHITE, 2017)

No trabalho em questão, é utilizada esta tecnologia para a delimitação de uma área onde as concessionárias sejam capazes de monitorar sempre que necessário onde estão seus produtos e se está havendo alguma irregularidade com algum dos mesmos.

2.3 *Android*

O sistema operacional (SO) se refere a um *software* que tem como objetivo o gerenciamento e administração de recursos do sistema em que é instalado, envolvendo tanto a parte de gerenciamento dos arquivos quanto aplicativos de terceiros e principalmente o *hardware* do aparelho. Em outras palavras, o sistema operacional é quem faz a interface para a comunicação entre usuário e máquina. Esta seria uma definição do *Android.* (ZOOM, 2021)

Atualmente o *Android* é o sistema mais utilizado em todo o planeta, pois está presente não somente em aparelhos celulares, mas também em eletrodomésticos, carros, TVs, sistemas embarcados, envolvendo assim uma grande gama de público e empresas que aderiram ao sistema, principalmente porque a *Google* continuou com o que foi proposto para o SO que era ser gratuito e de fácil utilização. (MEYER, 2020)

Estranhamente, a ideia inicial dos criadores era de ser um inovador sistema para câmeras digitais, mas quando viram que o mercado não era o que se esperava, decidiram prosseguir somente com o mercado móvel. *Rubin* (O criador do *Android*) e sua equipe, ofereceram ao público um novo e inovador meio operacional móvel, o *Open Source* (de código aberto, que pode ser modificado), baseado em *Linux, Android.* (MEYER, 2020)

O sistema operacional *Android* foi construído com base no *Kernel Linux, Kernel* nada mais seria que o núcleo, ou em outras palavras a principal aplicação de um SO é gerenciamento da memória e do uso do tempo da CPU. O *Linux* foi escolhido para ser a base do *Android* principalmente pelo motivo de ser um *Kernel open source*. (CIRIACO, 2018)

O desenvolvimento de uma aplicação para esse sistema precisa seguir quatro pilares (Ilustrado na figura 5) para que funcione com qualidade, são eles: *Activity*, *Services*, *Content Providers* e *Broadcast Receivers.* (CORDEIRO, 2015)

Figura 5 - Pilares para desenvolvimento de app para Android

Fonte: AndroidPro, 2022

2.3.1 *Activity*

Activity é a forma de representar uma tela de interface, por exemplo um menu para opções de um aplicativo de *e-mail*, os próprios *e-mails*, ou a forma de exibição deles. Cada uma destas telas é denominada *Activity* de forma separada. (CORDEIRO, 2015)

2.3.2 *Services*

Os *Services* são os itens que irão ser executados em segundo plano, como por exemplo um para reprodução de músicas, podendo acessar e transmitir dados pela rede sem interromper as ações do usuário. (CORDEIRO, 2015)

2.3.3 *Content Providers*

Os *Content providers* são responsáveis por fornecerem os dados de um aplicativo a outro sob demanda. (CORDEIRO, 2015)

2.3.4 *Broadcast Receivers*

Os *Broadcast Receivers* são aqueles que respondem às mensagens de *broadcast* do sistema, como por exemplo um aplicativo ao baixar um dado pode permitir que outros saibam daquele dado, e o responsáveis que saberão se tal dado foi baixado são os *Broadcast Receivers.* (CORDEIRO, 2015)

Após a apresentação da figura 5 destes pilares e a explicação de cada um, também é bom falar sobre a própria arquitetura do sistema, que pode ser explicada melhor separando-a em algumas partes que serão abordadas a seguir (Figura 6).

Figura 6 - Partes da estrutura do Android

Fonte: Autoria própria, 2022

2.3.5 Aplicativos

São os aplicativos ou jogos produzidos, como por exemplo *Ifood, Whatsapp, TikTok* e *Google Maps*. Existem muitos aplicativos que já estão implantados no sistema, sendo eles Contatos, Telefone e Calculadora, mais os aplicativos de terceiros podem instalar. (CORDEIRO, 2015)

2.3.6 *Frameworks*

Esta é a camada onde são oferecidos muitos dos serviços para serem utilizados pelos desenvolvedores dos aplicativos. Segundo CORDEIRO (2015), tem-se como principais serviços:

- *Activity Manager*: Responsável por controlar o ciclo de vida das *Activities* dos aplicativos.
- Gerenciador de Recursos: Responsável pelo fornecimento dos acessos aos recursos (Configurações, *layouts* etc.) que não são codificados.
- Gerenciador de Notificações: Responsável pela exibição de alertas ao usuário.
- Sistema de Visualização: Responsável por criar as interfaces com os usuários. (CORDEIRO, 2015)

2.3.7 Bibliotecas nativas

Este é um conjunto de bibliotecas que está incluído no *Kernel* do *Linux*, contendo o *WebKit* (mecanismo de código aberto para navegadores da web), um banco de dados SQL (*Structured Query Language*, ou Linguagem de Consulta Estruturada), bibliotecas SSL (*Secure Sockets Layer*, ou Certificados de Segurança) responsáveis pela proteção na internet, bibliotecas de reprodução de áudio ou vídeo etc. (CORDEIRO, 2015)

2.3.8 *Android Runtime*

O *Android Runtime* é o compilador e tradutor de arquivos que contém o código fonte para linguagem de máquina, além de ser o responsável por gerenciar a memória e a coleção de lixo. (SADOWSKA, 2021)

2.3.9 HAL (*Hardware abstraction Layer*)

Esta é a camada onde o SO interage com o *hardware*, sendo uma comunicação geral, abstrata ou a nível detalhado de *hardware*. O HAL pode ser chamado tanto do próprio SO quanto dos drivers, sendo em ambos os casos uma maneira mais geral de comunicação com o dispositivo. (HUANG e WU, 2018)

2.3.10 *Kernel Linux*

O *Kernel* é o local onde estão localizados principalmente os *drivers* de *hardware* essenciais do dispositivo (tela, câmera, teclas etc.) além de ser o responsável pela rede e outros drivers não essenciais. (CORDEIRO, 2015)

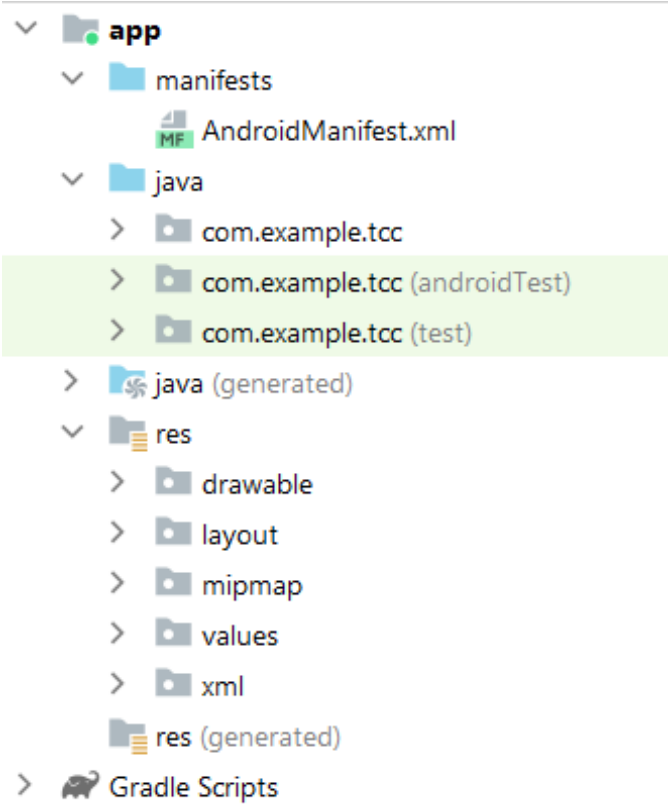
2.4 *Android Studio*

O *Android Studio*, que será usado neste trabalho, é o principal IDE (Ambiente de Desenvolvimento Integrado) para o desenvolvimento de aplicativos para sistemas *Android* baseado na IDE *IntelliJ IDEA*, oferecendo recursos como uma boa plataforma de emulação, bons *frameworks*, ferramentas de testes, um sistema flexível de compilação etc. (*ANDROID* STUDIO, 2022)

2.4.1 Estrutura de projeto

A exibição dos arquivos do projeto é organizada em módulos possíveis de acessar de maneira rápida e fácil, como apresenta a figura 7:

Figura 7 – Módulos de arquivos do projeto



Fonte: Autoria própria, 2022

No módulo *Grade Scripts* pode-se ver os arquivos de criação, já nos módulos de aplicativo é onde se encontram as seguintes pastas:

Manifests: Onde está o arquivo *AndroidManifest.xml*.

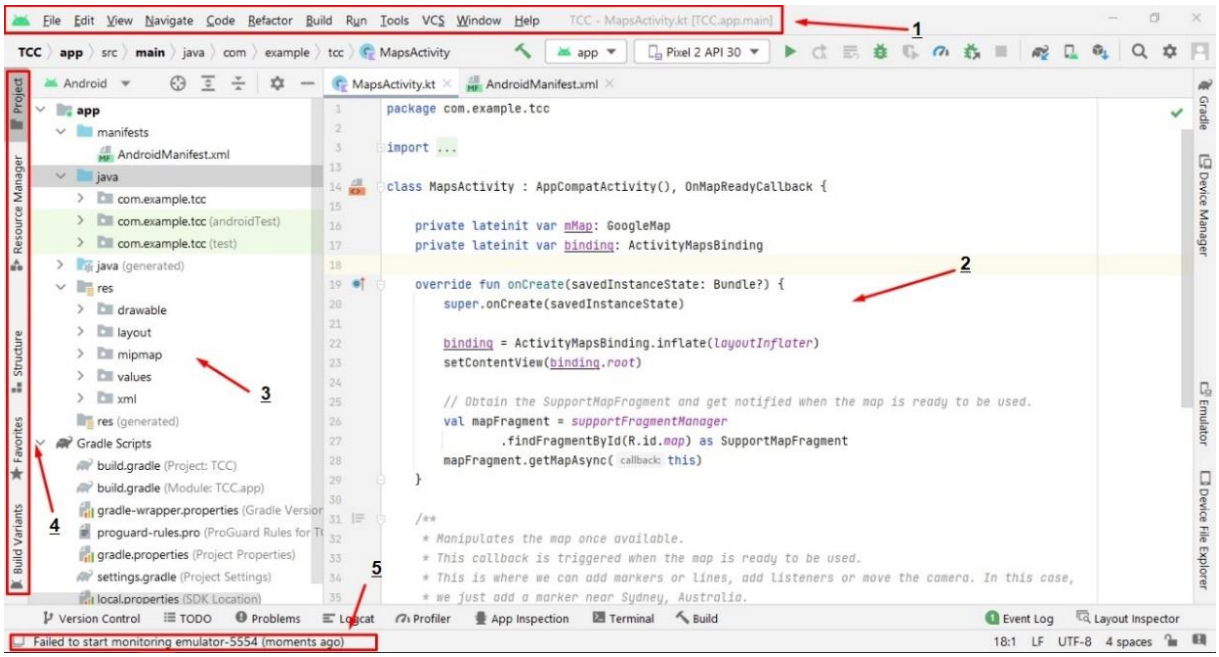
Java: Onde estão os arquivos com o código-fonte *Java*.

Res: Onde estão os recursos que não fazem parte de código, como interfaces, imagens em bitmap etc. (*ANDROID STUDIO*, 2022)

2.4.2 Interface

A interface usada é composta pelas áreas, apresentadas na figura 8:

Figura 8 – Áreas da interface



Fonte: Autoria própria. 2022

1 – Barra de ferramentas: Onde são realizadas as ações, como por exemplo iniciar as ferramentas *Android* ou executar o aplicativo.

2 – Janela do editor: Onde se escreve o código, esta janela muda conforme o tipo do arquivo, por exemplo se for um arquivo de *layout* o editor abrirá o *Layout Editor*.

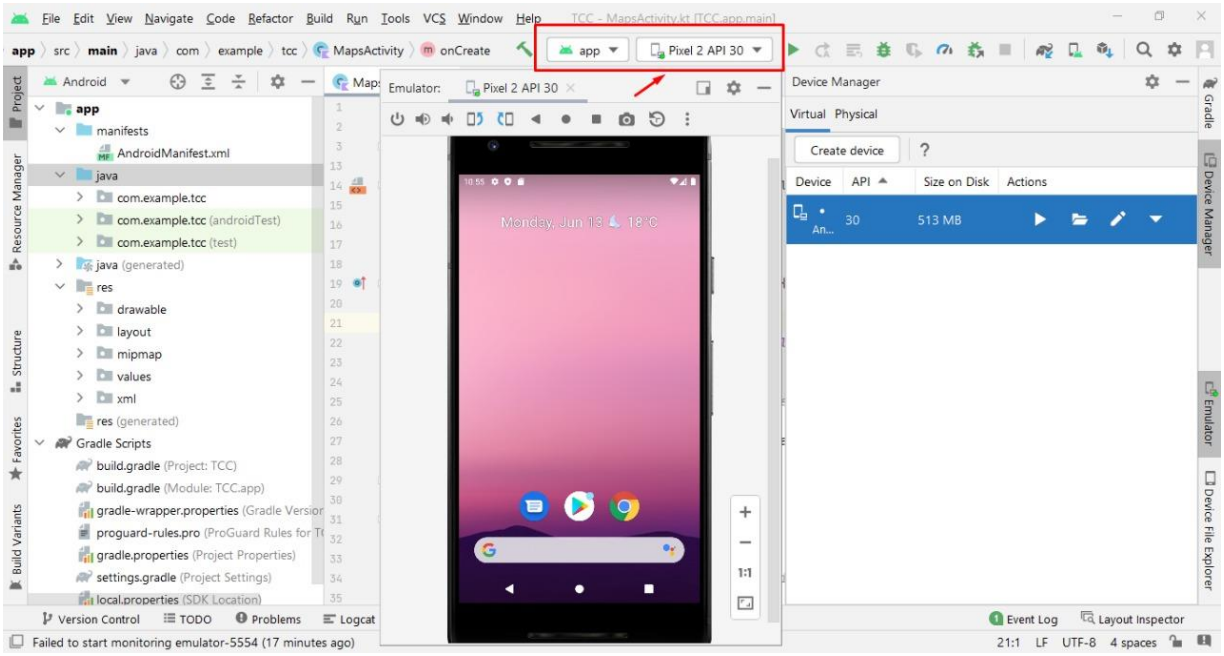
3 – Janela de ferramentas: Onde se acessam as tarefas como pesquisa, controle de versões, gerência de projetos.

4 – Barra de janela de ferramentas: Onde se usa de botões para expandir ou recolher as janelas.

5 – Barra de status: Local onde é exibido o status do projeto, incluindo do próprio ambiente e os avisos. (*ANDROID* STUDIO, 2022)

Há também a interface do emulador onde foram feitos os testes de utilização do *App* criado. Nele pode ser escolhida a versão do *Android* tanto quanto o nome do dispositivo. A versão utilizada no princípio foi a API (Interface de programação de aplicativos) 30 ou em outras palavras, o *Android* 11. A figura a seguir representa a tela do *smartphone* virtual junto a versão e o modelo da API:

Figura 9 – Interface do emulador



Fonte: Autoria própria, 2022

2.5 *Kotlin*

A linguagem utilizada para o desenvolvimento do aplicativo foi a *Kotlin*. Criada pela *JetBrains* (Mesma criadora das IDEs *IntelliJ IDEA, PHP Storm* e *Web* Storm), feita para suprir o que precisavam e não tinha em outras linguagens. (CASTIGLIONI, 2022)

A curva de aprendizado dessa linguagem é bem rápida e pequena, também sendo de natureza tipada, ela passa a oferecer bastante desempenho e segurança na hora da programação, mas há várias regras que necessitam de serem memorizadas, principalmente por também ser uma linguagem que não é preciso de muito código para o desenvolvimento, para contornar um pouco esse problema os erros são evidenciados durante a escrita do código e não na execução, facilitando bastante a correção e o retrabalho. (CAVALCANTE, 2022)

2.5.1 Sintaxes

A sintaxe da linguagem é dividida principalmente em variáveis, tipos, classes, funções, condicionais e vetores, descritos a seguir:

2.5.1.1 Variáveis

As variáveis nesta linguagem são definidas em dois tipos, sendo eles *val* e *var*.

val é a maneira que o *Kotlin* utiliza para a declaração de constantes, também a recomendada, esta forma necessita de ser inicializada logo que declarada, já *var* é a forma de declaração onde o valor da variável declarada pode ser alterado durante a programação e a execução do programa, a inicialização pode acontecer em qualquer momento do programa. (CAVALCANTE, 2022)

Ambas as declarações são escritas de acordo com a *Pascal Notation* onde o nome da variável vem antes do tipo na mesma linha, como o exemplo:

NomeVariavel : TipoVariavel.

2.5.1.2 Tipos

Para o *Kotlin* não é preciso a definição explícita do tipo, pois ele é definido quando a variável recebe o primeiro valor, sendo assim uma variável pode ser o que preferir até o momento em que ela recebe os primeiros dados. (CAVALCANTE, 2022)

Estes tipos podem variar de acordo com este valor recebido, sendo eles demonstrados na tabela 1 a seguir:

Tabela 1 - Tipos de variáveis

Tipos	**Descrição**
long	Inteiro de 64 bits
int	Inteiro de 32 bits
short	Inteiro de 16 bits
byte	Inteiro de 8 bits
double	Ponto flutuante de 64 bits
float	Ponto flutuante de 32 bits
boolean	Recebe verdadeiro ou falso como valor
char	Caracteres
string	Vetor de caracteres, criada utilizando aspas duplas ou triplas

Fonte: CAVALCANTE, 2022

2.5.1.3 Classes

As classes nessa linguagem não podem ser estendidas, sendo definidas da forma: *class NomeClasse*, *class* sendo uma palavra reservada para a definição e *NomeClasse* o nome que escolher para esta classe. (CAVALCANTE, 2022)

O construtor de classe diferente de outras linguagens, não precisa ser declarado, ele é feito declarando os atributos entre parênteses logo após o nome da classe, como por exemplo:

Class NomeClasse (var atributo1: int)

Se não houver nenhum atributo será utilizado de um construtor padrão sem a necessidade de parâmetros. (CAVALCANTE, 2022)

2.5.1.4 Funções

As funções são definidas como as classes, só que com a palavra reservada *fun* seguida do nome da função, e em parênteses após o nome são definidos os tipos de entrada, e por fim antes do código da função deve-se também definir o tipo de saída. (CAVALCANTE, 2022)

fun NomeFuncao (parametro1 : int) : int {}

2.5.1.5 Condicionais

As expressões condicionais são escritas de uma forma que é bem comum nas outras linguagens. (CAVALCANTE, 2022)

if (condicao) {} else {}

2.5.1.6 Vetores

Os vetores são um pouco diferentes do *Java*, pois são representados por uma classe com o nome de *array*, que possui como principais funções o *get* e o *set*, sendo definidos no programa da seguinte forma (CAVALCANTE, 2022):

val vetor1 = array(tamanho) {}

ou

val vetor2 = array<tipo> (tamanho, {})

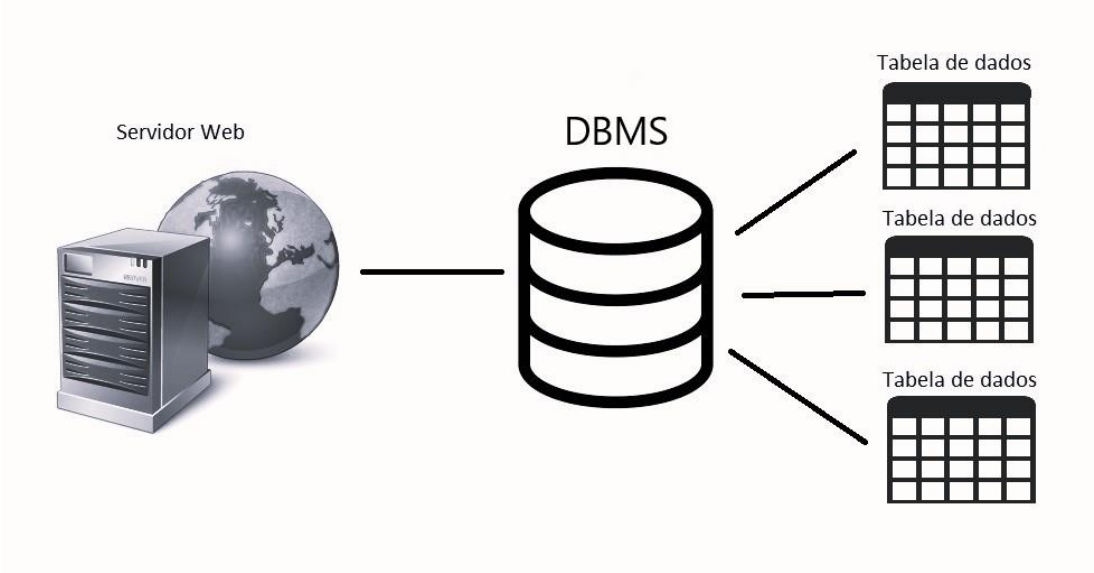
Estas são as definições principais da sintaxe *Kotlin,* que foi escolhida para este projeto por se tratar de uma linguagem desenvolvida especificamente com foco em *Android* e com compatibilidade total com o *Android Studio,* além da facilidade de escrita.

2.6 Banco de dados

As informações, configurações e marcações realizadas no projeto, são todas armazenadas em um banco de dados, sendo este do tipo relacional. (ORACLE, 2022)

Banco de dados é definido por ser um local comumente eletrônico, em que são armazenados qualquer tipo de dados, para que sejam acessados e listados posteriormente, através de uma linguagem de programação cujo nome é SQL, ou noSQL (*Not Only* SQL*)* se ele for um Banco de dados não-relacional. Estes dados são acessados utilizando de um DBMS (*Database Management Systems* ou Sistema de Gerenciamento de Banco de Dados) que por sua vez é um *software* que utiliza de métodos padrões para efetuar consulta sob estes dados, gerenciando e organizando-os da forma que o programador preferir, representado brevemente pela figura 10. (ORACLE, 2022)

Figura 10 – Banco de dados introdução



Fonte: HOMEHOST (Modificada)

As DBMS mais utilizadas atualmente incluem o *MySQL, Oracle, PostgreSQL e Microsoft Access*. (TECHLIB, 2022)

2.6.1 SQL

SQL é a linguagem que quase todos os Bancos de dados relacionais utilizam para a gerência deles. Desenvolvida pela IBM (*International Business Machines* ou Corporação Internacional de Máquinas de Negócios) entre os anos de 1970. (ORACLE, 2022)

Por se tratar de uma linguagem declarativa não é necessário ter um grande conhecimento em programação para começar a escrever consultas ou pedidos, chamados de *queries*. (SILVEIRA, 2019)

2.6.1.1 *SELECT, a principal query do SQL*

O *SELECT* é a *query* para a consulta em quase todos os bancos, sendo bastante simples a escrita, parecendo muito com uma frase escrita em inglês:

*SELECT * FROM tabela WHERE variavel < 100*

É traduzido como: Selecione (*SELECT*) todos os resultados (*) onde (*WHERE*) variável seja menor que 100 (*variavel1<100*). Neste exemplo possuindo as palavras reservadas *FROM* (Define de onde será consultado os dados) e *WHERE* (Define a condição dos dados que serão filtrados). O asterisco funciona para consultar todos os dados que se encaixarem nas condições, tendo também a opção de especificar os dados para busca alterando-o pelo nome de alguma coluna da tabela selecionada, podendo ser mais de uma separando por vírgula. (SILVEIRA, 2019)

2.6.1.2 Principais comandos

Há vários comandos úteis dentro da linguagem SQL, mas há alguns que não podem deixar de ser mencionados, pois são os responsáveis pelas consultas e alterações de dados, sendo eles apresentados na tabela 2: (SILVEIRA, 2019)

Tabela 2 - Principais comandos SQL

COMANDO	**DESCRIÇÃO**
SELECT	Responsável pela busca de linhas na tabela seguindo condições impostas (*WHERE*).
INSERT	Responsável pela inserção de novos dados nas tabelas, exemplo: *INSERT INTO* tabela (dado) *VALUES* 25. Este comando pode ser descrito como: Inserir (*INSERT*) na (*INTO*) tabela o valor (*VALUES*) 25.
UPDATE	Atualiza as linhas das tabelas de acordo com uma condição (*WHERE*), como por exemplo a alteração de um número telefônico.
DELETE	Remove as linhas e dados de uma tabela de acordo com uma condição (*WHERE*).

Referência: SILVEIRA, 2019

Há também uma enorme gama de comandos que não foram citados que podem aperfeiçoar a busca ainda mais, desde junção de tabelas a agrupamentos etc.

2.6.1.3 Modelagem

A modelagem, a forma de criação das tabelas onde serão armazenados os dados, se dá pelos comandos *CREATE TABLE* e *ALTER TABLE*. Seguindo esses comandos é possível a criação e a estruturação de qualquer tabela da maneira que preferir, exemplo: (SILVEIRA, 2019)

CREATE TABLE tabela ((Parâmetros));

2.6.2 NoSQL

Este termo refere-se a tipos de banco de dados não relacionais, onde os dados não são armazenados no formato de tabelas, tal tipo de banco de dados pode ser acessado com a utilização de APIs de linguagem idiomática e por linguagens de consulta estruturadas. (ORACLE,2022)

Tal formato de linguagem tem como benefício em relação o SQL os quesitos de processamento de grandes dados, sejam eles não relacionados, em mudança constante ou indeterminados, dados que não necessitam de esquemas ou que tais esquemas sejam definidos pela aplicação, aplicativos que necessitam de disponibilidade e desempenho acima de tudo ou aplicativos que sempre estão executando, sendo acessado por pessoas de vários países. (AZURE, 2022)

Diferente do SQL esta linguagem é organizada em diferentes tipos de modelos, a tabela 3 tem uma descrição dos quatro tipos mais utilizados na indústria, seguido da figura 11 que representa em forma visual estes modelos.

Tabela 3 - Modelos de Banco de Dados não-relacionais

Nome do modelo	**Descrição**
Chave-Valor	Utiliza-se de uma tabela *hash* para efetuar o pareamento de chaves e de valores (dados).
Documento	Também possui o conceito de chave-valor, mas organizando documentos inteiros em uma coleção que por sua vez é um grupo de dados.
Colunar	Armazenam os dados em colunas onde podem ser consultadas de forma específica cada uma dentro do banco de dados.
Grafo	Utiliza-se de grafos para armazenagem dos dados, utilizando dos conceitos de nó e bordas para representar os dados e suas conexões.

Fonte: AZURE, 2022

Figura 11 – Modelos de Banco de Dados NoSQL

Fonte: Autoria Própria

Utilizado no aplicativo para armazenar os dados enviados pelo cliente em um servidor, para que então o aplicativo do administrador colete tais dados e mostre no mapa a localização em tempo real no aparelho do veículo.

3 DESENVOLVIMENTO DO SOFTWARE

Neste capítulo são tratados os métodos utilizados e o que foi feito para o desenvolvimento do Software cujo nome decidido como *GeoDrive*.

3.1 API do *Google Maps*

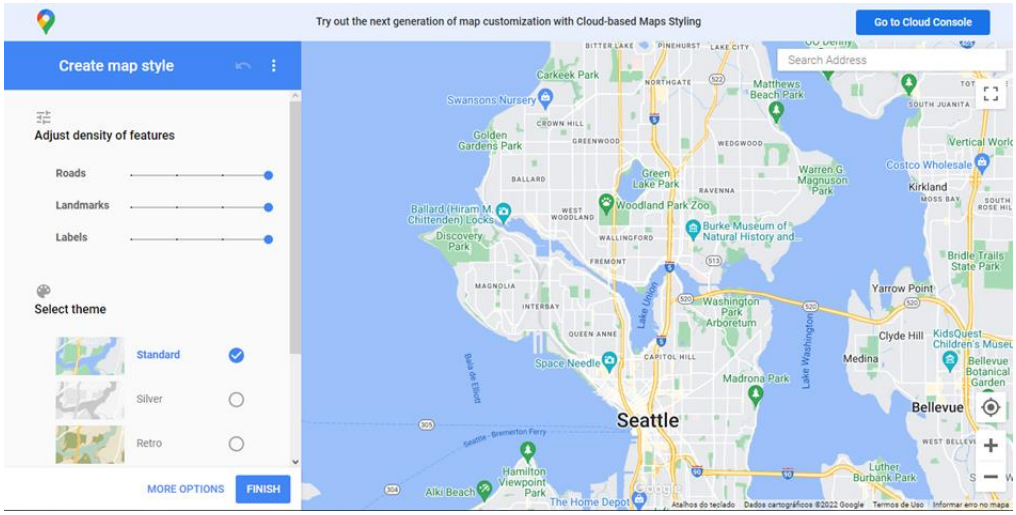
Para começar o desenvolvimento, foi necessário a criação de uma chave de acesso a API da *Google.* Tal chave pode ser encontrada ao acessar o *Console* da *Google* e solicitar a criação de uma. Essa chave é o que libera a possibilidade do uso da interface da *Google* em relação a mapas. Cada usuário possui sua própria chave e deve ser solicitada no *console* oficial.

Ela deve ser inserida no programa junto ao estilo de mapa que será utilizado.

3.2 Estilo do mapa

O estilo do mapa foi criado usando a ferramenta também da *Google, Styling Wizard* conforme a figura 12*,* onde há parâmetros que definem o tipo do mapa, tais como: cor, quantidade de nomes amostra, quantidade de ruas, dia ou noite etc. Depois de definidos os parâmetros, é gerado um código para um arquivo do formato *JSON*, tal arquivo que precisa ser colocado na pasta do projeto e então apontado dentro do código como o estilo de mapa a ser utilizado.

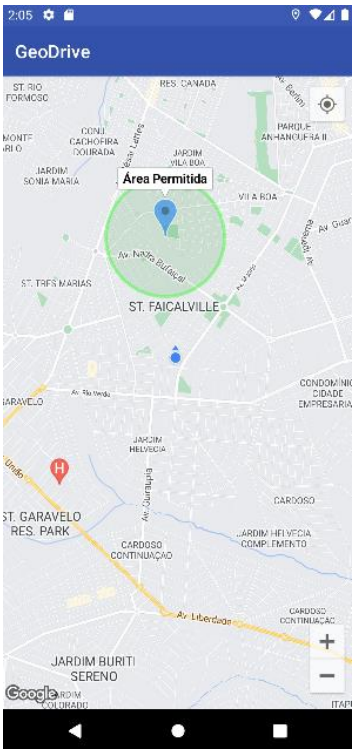
Figura 12 – Styling Wizard



Fonte: Google

Se não for utilizado deste arquivo gerado pela ferramenta, o mapa a ser definido será o mesmo utilizado no aplicativo, já existente nos celulares, *Google Maps*. O utilizado no aplicativo desenvolvido foi um estilo mais básico voltado para o estilo padrão, mas com menor poluição visual por parte das *tags* e nomes de bairros e cidades, ilustrado na figura 13.

Figura 133 – Estilo do mapa escolhido



Fonte: Autoria Própria, 2022

3.3 Código

Neste capítulo são detalhadas e explicadas algumas funções importantes do código desenvolvido:

3.3.1 *OnCreate()*

Assim que é realizada uma solicitação de criação de *geofence* é chamada uma função que tem como parâmetro o estado em que se encontra a instância atual do aplicativo. Nesta função também é criado o fragmento de mapa e seu respectivo estilo, sendo este sempre sincronizado com a *Google* verificando também se obteve sucesso em cada conexão. Por fim são criadas variáveis que recebem os serviços de integração com o provedor do *Maps* e com o cliente de *Geofencing*.

3.3.2 *onMapReady()*

Esta é a principal função do programa para o início do acesso ao *app*, pois ela é a que verifica as permissões de localização, define o estilo de mapa que será utilizado e chama as funções de criação e inspeção de *Geofencing*. Inicia definindo uma variável para receber o mapa seguido pela definição do estilo, que foi importado através de um arquivo *json* obtido no *site* de criação de estilos da própria *Google*. Após estas definições são feitas várias solicitações e verificações de permissões e se o sistema de *GPS* está ativo no aparelho e, se estiver moverá a câmera para o local que foi detectado o sinal, terminando com a chamada às funções de criação de *Geofencing*.

3.3.3 *setPoiClick()*

Esta função recebe como parâmetro o próprio mapa. Nela é criada uma variável *Listenner* que sempre verifica se foi dado um toque rápido em cima de uma *Geofence* criada para então mostrar as informações dela, como por exemplo, o nome.

3.3.4 *setLongClick()*

Como na *setPoiClick()* nesta é criada outra variável, mas neste, ela fica aguardando um toque longo em algum lugar dentro do *App*. Ao ser feito tal toque ela cria tanto o marcador quanto o raio da *geofencing*, o marcador sendo criado com uma cor pré-definida que neste caso é o ciano, e o raio é definido pelo usuário administrador, em um *pop-up* que aparece solicitando o tamanho do raio em metros. Após os parâmetros serem passados a *geofence* é criada como um círculo verde com um marcador acima da mesma junto ao seu nome.

Por fim também são armazenados os parâmetros de localização e tamanho do raio no banco de dados, para ser possível a recuperação posterior.

3.3.5 *createGeofence()*

Para esta função são passados como parâmetros as coordenadas da *Geofence* e as variáveis dos clientes. Começa chamando a função *builder()* da biblioteca da *Google*, que é a biblioteca para a criação de *Geofences* já pronta. Nessa função são passadas as configurações de local, tamanho do raio, duração para expirar, transições de entrada e saída e o comando de construção. Depois é utilizada de uma variável para tratar da requisição dessa criação, seguida por uma estrutura condicional para verificar se as permissões ainda estão habilitadas.

3.3.6 *showNotifications()*

Função criada para enviar notificações de quando alguém entra ou sai da *Geofence*, esta utiliza da função padrão do *Android*, *NotificationCompat.Builder()*, que são passados como parâmetro o ícone, o texto, o título e o estilo da notificação. Após esta criação é criada outra variável responsável por ser o gerenciador de notificação, que recebe o serviço de notificação do sistema, seguido da estrutura condicional para criação do canal de notificação junto ao serviço de notificações.

4 TESTES E RESULTADOS

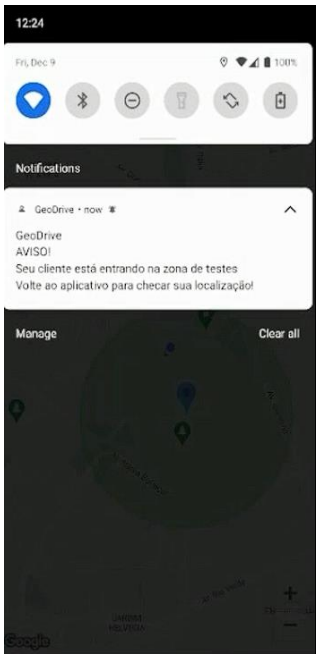
Neste capítulo são retratados os testes efetuados e os resultados da implementação:

4.1 Testes

Os testes efetuados na aplicação foram para entrada e saída de veículos, definição da *geofence* e o aparecimentos de notificações.

Para verificar o funcionamento das funções do aplicativo, é utilizado do emulador do próprio *Android Studio* que possui a função de emular também a localização fornecida ao dispositivo como mostrado na figura 14.

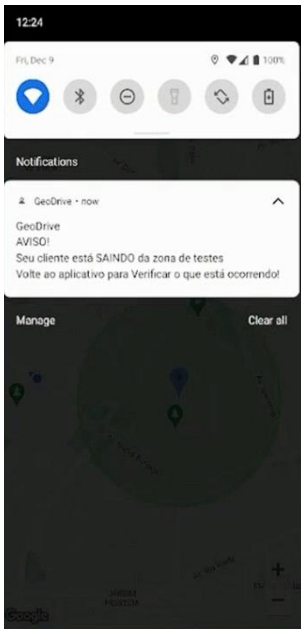
Figura 144 – Teste de entrada na Geofence



Fonte: Autoria Própria, 2022

Neste teste foi definido uma *geofence* pelo usuário e emulado o andar do veículo, e quando tal cliente adentra o local é acionada uma notificação para avisar o vendedor de tal movimentação. O mesmo ocorre para saída do veículo mostrado na figura 15.

Figura 155 – Teste de saída na Geofence



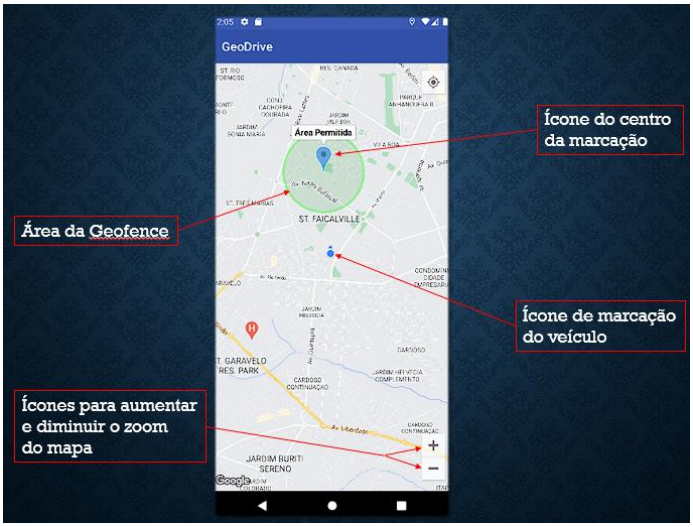
Fonte: Autoria Própria, 2022

Ambos os testes além de verificar o funcionamento da *geofence* estão também verificando o funcionamento das notificações.

4.2 Resultados

Os resultados do desenvolvimento da aplicação são satisfatórios, pois tudo que foi implementado e está funcionando em concordância com o proposto, a interface é um exemplo, pois ficou de acordo com o imaginado no começo do projeto, como mostrado na figura 16, onde tanto o mapa desenvolvido no *Styling Wizard* quanto as formas da *geofence* funcionaram de acordo.

Figura 166 – Detalhes sobre a interface



Fonte: Autoria Própria, 2022

5 CONSIDERAÇÕES FINAIS

O presente aplicativo possui a finalidade de facilitar tanto o trabalho dos vendedores quanto a apresentação do veículo aos clientes das concessionárias e revendedoras de automóveis em geral. Feito para dispositivos *Android*, através da ferramenta *Android Studio* utilizando da linguagem de programação *Kotlin*.

Levando em conta os estudos realizados sobre o Geolocalização, *Geofencing* e a ferramenta *Android Studio*, a criação de tal aplicativo foi perfeitamente possível, além de possuir uma excelente usabilidade e interatividade, aprimorando o vínculo com o cliente por dar mais liberdade na hora de testar seu veículo.

A linguagem *Kotlin* se encaixou muito bem no conceito deste trabalho por ser uma linguagem focada em otimização e funcionalidades, tendo como principal ponto as rápidas consultas e inserções que podem ser feitas pelos usuários, possuindo também uma boa estrutura de funções com uma boa definição de cada.

Foi escolhido o modo radial para a demarcação do local da *Geofence* por ser mais simples e rápido na hora de ser definido, possuindo uma curva de aprendizagem mais acessível para não ser necessário um treinamento específico para o uso do *app*, sendo assim, gastando menos tempo aprendendo, tentando dominar as funções do aplicativo e mais tempo na loja exercendo sua função e efetuando cada vez mais vendas.

O remanejamento de vendedores para cumprir funções de “guia” para clientes em ocasiões de testes já está se tornando volúvel por conta da chegada de carros autônomos, com isso a única função que um vendedor terá com o cliente será para garantir a segurança de que o veículo não seja furtado, e com este projeto essa preocupação tende a nulidade, pois a todo momento é possível observar o trajeto que o cliente está percorrendo, sendo que o responsável pelo monitoramento pode acionar as autoridades a qualquer instante conforme ocorra algum transtorno entregando as informações de local exatas do veículo, facilitando o trabalho das autoridades

Não sendo definido explicitamente a forma de rastreio para que fique a critério de cada empresa como será feito, um exemplo seria a utilização de um *Tablet* ou um dispositivo *Android* qualquer, possuindo suas configurações bloqueadas por questão de segurança, e sempre enviando seus dados de localização em tempo real para o administrador do sistema.

Caso a empresa opte pela utilização de um dispositivo *Android* também facilitará para que o cliente veja as demarcações de limite de até onde o veículo possa circular sem que acione o alarme ao administrador, possuindo um limite extra para evitar de acionar em ruas que não possuem retorno dentro da área demarcada sendo este o limite de tolerância, também permitindo que o cliente fique fora da área por um certo período, aprimorando assim os pontos fracos do sistema de *Geofence* radial.

A aplicação desenvolvida tornou possível tanto o monitoramento em tempo real do veículo, quanto a administração do que fazer perante irregularidades.

O código do aplicativo desenvolvido encontra-se no Apêndice e numerado com a mesma ordenação gerada pelo *Android Studio*.

6 TRABALHOS FUTUROS

Os trabalhos que podem ser realizados no futuro levando o trabalho em questão como base são diversos, pois este, cobre uma gama enorme de possibilidades, entre elas estão:

- Aplicação destes conceitos em outras áreas do mercado, como em lojas de eletrônicos e afins com foco na segurança e na redução de roubos.
- O aprimoramento deste trabalho adicionando mais funcionalidades no aplicativo final.
- Utilização em locais de entrada e saída monitoradas como em uma escola ou em um local de trabalho para evitar desavenças com familiares ou ajudar no ponto de trabalhadores, retirando a necessidade de relembrar todos os dias.
- Alteração da forma em que é feito o rastreio dos veículos.
- Utilização do mesmo para demarcação de território e rastreio de produtos vivos como gado, cavalos e afins.
- Utilização do dispositivo *GPS* dos próprios veículos, ou integração com veículos autônomos.
- Portabilidade para outros dispositivos, como o *IOS*, computadores de mesa e *Notebooks*.
- Adicionar uma maneira de comunicação entre o administrador e o cliente por meio de ligações, onde o cliente pode tirar suas dúvidas com o vendedor e o vendedor entrar em contato em caso de alguma rota não programada.
- Integração do dispositivo *Android* rastreador com um componente *RFID* com o fim de evitar o furto, incluindo algum tipo de tentativa de desligamento ou retirada do aparelho, também seria bem-vindo.
- Utilização do GPS do próprio veículo para o rastreamento.

Com todas estas opções capazes para a utilização deste aplicativo torna-se de vital importância que ele seja continuado e aprimorado junto ao decorrer do tempo e das necessidades que surgirão, também abrindo as portas para outras ideias

envolvendo este conceito de aplicativo, e todas que somem positivamente este trabalho sejam bem-vindas.

Dando oportunidade inclusive para vários desdobramentos do presente trabalho em outros campos relacionados à Engenharia de Computação.

7 REFERÊNCIAS

Banco de dados definido. **Oracle**, 2022. Disponível em: <https://www.oracle.com/br/database/what-is-database/>. Acesso em 9 de maio de 2022.

Banco de Dados NoSQL – O que é NoSQL?. **Microsoft Azure**, 2022. Disponível em: <https://azure.microsoft.com/pt-br/overview/nosql-database/>. Acesso em 11 de maio de 2022.

CASTIGLIONI, Matheus. Começando com *Kotlin*. **Blog do Matheus Castiglioni**, 2018. Disponível em: <https://blog.matheuscastiglioni.com.br/comecando-com-kotlin/>. Acesso em 8 de maio de 2022.

CAVALCANTE, Pablo. Introdução a *Kotlin*: Um guia para começar. **GeekHunter**, 2022. Disponível em: <https://blog.geekhunter.com.br/introducao-a-kotlin/>. Acesso em 8 de maio de 2022.

CHAMBERLAIN, Lauryn. GeoMarketing 101: *What is Geofencing?.* **GeoMaketing**, 2016. Disponível em: <https://geomarketing.com/geomarketing-101-what-is-geofencing>. Acesso em: 23 de fev. de 2022.

CIRIACO, Douglas. *Android* é *Linux*? Qual a relação do SO da *Google* com o *Linux*?. **TecMundo**, 2018. Disponível em: <https://www.tecmundo.com.br/software/127038-*Android*-*Linux*-*Kernel*.htm>. Acesso em 28 de fev. de 2022.

Conheça o *Android Studio*. **ANDROID STUDIO**, 2022. Disponível em: <https://developer.*Android*.com/studio/intro?hl=pt-br>. Acesso em 1 de maio de 2022.
CORDEIRO, Fillipe. Recursos do Sistema *Android*. **AndroidPro**, 2015. Disponível em: <https://www.androidpro.com.br/blog/desenvolvimento-*Android*/recursos-do-sistema-*Android*/>. Acesso em 28 de fev. de 2022.

DBMS. **TechLib**, 2022. Disponível em: <https://techlib.wiki/definition/dbms.html>. Acesso em 9 de maio de 2022.

GOOGLE. *Styling Wizard*, 2022. Criador de estilo de mapas. Disponível em: <https://mapstyle.withgoogle.com/>. Acesso em 05 de novembro de 2022.

GPS - O que é, como funciona. **Só Física**, 2008-2022. Disponível em: <*http://www.sofisica.com.br/conteudos/curiosidades/gps.php>.* Acesso em 6 de maio de 2022.

HUANG, Dijiang. WU, Huijin. *Hardware Abstraction Layer*. **ScienceDirect**, 2018. Disponível em: <https://www.sciencedirect.com/topics/computer-science/hardware-abstraction-layer>. Acesso em 29 de fev. de 2022.

KHOURY, Joseph. *How it is made? Global Positioning System* (GPS). **uOttawa**, 2011. Disponível em: <http://aix1.uottawa.ca/~jkhoury/GPS.pdf>. Acesso em: 20 de fev. de 2022.

MEYER, Maximiliano. A história do *Android*. **OFICINA DA NET**, 2015. Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-*Android*>. Acesso em 27 de fev. de 2022.

O que é banco de dados?. **Home Host**, 2022. Disponível em: <https://www.homehost.com.br/blog/tutoriais/mysql/o-que-e-um-banco-de-dados/>. Acesso em 10 de maio de 2022.

O que é Geofencing?. **Mais Geek**, 2022. Disponível em:<https://maisgeek.com/o-que-e-geofencing/>. Acesso em 23 de fev. de 2022.

O que é Latitude e Longitude?. **Universidade Federal de Santa Maria**, 2027. Disponível em: <http://coral.ufsm.br/cartografia/index.php?option=com_content&view=article&id=43&Itemid=39ANDROID>. Acesso em 6 de maio de 2022.

O que é NoSQL?. **Oracle**, 2022. Disponível em: <https://www.oracle.com/br/database/nosql/what-is-nosql/>. Acesso em 11 de maio de 2022.

O que é Sistema Operacional?. **ZOOM**, 2021. Disponível em: <https://www.zoom.com.br/notebook/deumzoom/o-que-e-sistema-operacional>. Acesso em 26 de fev. de 2022.

O que é Trilateração?. **FiodeVida**, 2021. Disponível em: <https://fiodevida.com/o-que-e-trilateracao/>. Acesso em 20 de fev. 2022.

SADOWSKA, Paulina. *Android Runtime – How Dalvik and ART work?*. **ProAndroidDev**, 2021. Disponível em: <https://proandroiddev.com/*Android*-runtime-how-dalvik-and-art-work-6e57cf1c50e5>. Acesso em 28 de fev. de 2022.

SILVEIRA, Paulo. O que é SQL?. **Alura**, 2019. Disponível em: <https://www.alura.com.br/artigos/o-que-e-sql>. Acesso em 10 de maio de 2022.

TECNOLOGIA. Como funciona o GPS?. **Cultura Mix**, 2015. Disponível em: <https://www.culturamix.com/tecnologia/como-funciona-o-gps/>. Acesso em 12 de abril de 2022.

Tiago. Aplicativo de Geolocalização. **Mundo DevOps**, 2021. Disponível em: <https://mundodevops.com/blog/aplicativo-de-geolocalizacao/>. Acesso em 21 de fev. de 2022.

VELA, João. Coordenadas Geográficas. **InfoEscola**, 2006. Disponível em: <https://www.infoescola.com/geografia/coordenadas-geograficas/>. Acesso em 15 de abril de 2022.

WHITE, Sarah K. *What is Geofencing? Putting location to work*. **CIO**, 2017. Disponível em: <https://www.cio.com/article/2383123/geofencing-explained.html>. Acesso em 26 de fev. de 2022.

8 ANEXO 1

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

RESOLUÇÃO nº 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Alex Lourenço de Carvalho do Curso de Engenharia da Computação, matrícula 2018.1.0033.0316-2, telefone: (62) 99365-5330 e-mail alexgoiasgol10@hotmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Utilização de Geofencing para aprimorar a segurança em uma concessionária de veículos, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 29 de Setembro de 2022.

Assinatura do autor: Alex Lourenço de Carvalho

Nome completo do autor: Alex Lourenço de Carvalho

Assinatura do professor–orientador:

Signer ID: CWA77EPEW8

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

APÊNDICE

Neste capítulo será apresentado o código desenvolvido junto de seu respectivo arquivo:

9.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.codemave.geofencing">
    map:mapId="@string/map_id"
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />        <!--
Solicitação de permissoes para o app-->
    <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Geofencing">    <!--Definição do ícone e nome
do app na bandeja do celular -->

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="Chave de segurança" />   <!-- Definição da chave API pega
no console da google para utilização do maps-->

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps">   <!-- Definição do título e
modelo do app -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
    </application>

</manifest>
```

9.2 *GeofenceReceiver.kt*

```
package com.codemave.geofencing

import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import com.google.android.gms.location.Geofence
import com.google.android.gms.location.GeofencingEvent
import com.google.android.gms.maps.model.LatLng
import com.google.firebase.FirebaseApp
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
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

class GeofenceReceiver : BroadcastReceiver() {
    lateinit var key: String
    lateinit var text: String

    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) {
                // Retrieve data from intent
                if (intent != null) {
                    key = intent.getStringExtra("key")!!
                    text = intent.getStringExtra("message")!!
                }

                val firebase = Firebase.database
                val reference = firebase.getReference("reminders")
                val reminderListener = object : ValueEventListener{
                    override fun onDataChange(snapshot: DataSnapshot) {
                        val reminder = snapshot.getValue<Reminder>()
                        if (reminder != null) {
```

```
                    MapsActivity
                        .showNotification(
                            context.applicationContext,
                            "Location\nLat: ${reminder.lat} - Lon: ${reminder.lon}"
                        )
                }
            }

            override fun onCancelled(error: DatabaseError) {
                println("reminder:onCancelled: ${error.details}")
            }

        }
        val child = reference.child(key)
        child.addValueEventListener(reminderListener)

        // remove a geofence
        val triggeringGeofences = geofencingEvent.triggeringGeofences
        MapsActivity.removeGeofences(context, triggeringGeofences)
    }
  }
}
```

9.3 *MapsActivity.kt*

```
package com.codemave.geofencing

//importação das bibliotecas
import java.util.Scanner
import android.Manifest
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.app.job.JobInfo
import android.app.job.JobScheduler
import android.content.ComponentName
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Color
import android.location.Location
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
```

```
import android.os.Bundle
import android.util.Log
import android.widget.EditText
import android.widget.Toast
import androidx.core.app.ActivityCompat
import androidx.core.app.JobIntentService
import androidx.core.app.NotificationCompat
import androidx.core.content.ContextCompat
import com.google.android.gms.location.*
import com.google.android.gms.maps.*
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.model.MapStyleOptions
import com.google.android.gms.maps.model.CircleOptions
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.Marker
import com.google.android.gms.maps.model.MarkerOptions
import com.google.android.gms.maps.model.BitmapDescriptorFactory
import com.google.android.gms.tasks.CancellationToken
import com.google.firebase.FirebaseApp
import com.google.firebase.database.ktx.database
import com.google.firebase.ktx.Firebase
import java.io.Serializable
import kotlin.random.Random
import com.google.android.gms.maps.MapView
```

//Declaração das variáveis globais e constantes

```
const val GEOFENCE_RADIUS = 600
const val GEOFENCE_ID = "REMINDER_GEOFENCE_ID"
const val GEOFENCE_EXPIRATION = 10 * 24 * 60 * 60 * 1000 // 10 days
const val GEOFENCE_DWELL_DELAY =  10 * 1000 // 10 secs // 2 minutes
const val GEOFENCE_LOCATION_REQUEST_CODE = 12345
const val CAMERA_ZOOM_LEVEL = 13f
const val LOCATION_REQUEST_CODE = 123
private val TAG: String = MapsActivity::class.java.simpleName

class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var map: GoogleMap
    private lateinit var fusedLocationClient: FusedLocationProviderClient
    private lateinit var geofencingClient: GeofencingClient


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        val mapFragment = supportFragmentManager        //Cria, inicia e notifica
quando o serviço de maps
```

```
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)

        fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
        geofencingClient = LocationServices.getGeofencingClient(this)
    }


    override fun onMapReady(googleMap: GoogleMap) {
        map = googleMap
        map.uiSettings.isZoomControlsEnabled = true
        /////////
        map.setMapStyle(MapStyleOptions.loadRawResourceStyle(this@MapsActivity,
R.raw.style)) //Definição do estilo de mapa que será utilizado
        /////////

        if (!isLocationPermissionGranted()) {
            val permissions = mutableListOf(
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.ACCESS_COARSE_LOCATION   //Se a permissão
de localização não for concedida solicitar novamente
            )
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {

permissions.add(Manifest.permission.ACCESS_BACKGROUND_LOCATION)
            }
            ActivityCompat.requestPermissions(
                this,
                permissions.toTypedArray(),
                LOCATION_REQUEST_CODE
            )
        } else {

            if (ActivityCompat.checkSelfPermission(
                    this,
                    Manifest.permission.ACCESS_FINE_LOCATION
                ) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(
                    this,
                    Manifest.permission.ACCESS_COARSE_LOCATION
                ) != PackageManager.PERMISSION_GRANTED      //prossegue com o
código se as permissões foram concedidas
            ) {
                return
            }
            this.map.isMyLocationEnabled = true  //Define como localização ativada
para a variavel do googlemap
```

```
        // Move a camera para onde foi criado a geofencing
        fusedLocationClient.lastLocation.addOnSuccessListener {
            if (it != null) {
                with(map) {
                    val latLng = LatLng(it.latitude, it.longitude)
                    moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,
CAMERA_ZOOM_LEVEL))
                }
            } else {
                with(map) {
                    moveCamera(
                        CameraUpdateFactory.newLatLngZoom(
                            LatLng(65.01355297927051, 25.464019811372978),
                            CAMERA_ZOOM_LEVEL
                        )
                    )
                }
            }
        }

        setLongClick(map)
        setPoiClick(map)
    }

    // mostra a info da geofencing ao tocar rapidamente a tela
    private fun setPoiClick(map: GoogleMap) {
        map.setOnPoiClickListener { poi ->
            map.addMarker(
                MarkerOptions()
                    .position(poi.latLng)
                    .title(poi.name)
            ).showInfoWindow()

            scheduleJob()
        }
    }

    // Define a forma de criar a geofencing como um toque longo
    private fun setLongClick(map: GoogleMap) {
        map.setOnMapLongClickListener { latlng ->

            // Cria o marcador da Geofencing
            map.addMarker(
                MarkerOptions().position(latlng)
                    .title("Área Permitida")
```

```
.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE
))
                .alpha(0.7f)
            ).showInfoWindow()
            // Cria o círculo marcador da geofencing
            map.addCircle(
               CircleOptions()
                  .center(latlng)
                  .strokeColor(Color.argb(100, 30, 255, 20))
                  .fillColor(Color.argb(35, 30, 150, 40))
                  .radius(GEOFENCE_RADIUS.toDouble())
            )

            //Efetua a conexão com o banco de dados e envia as definições do local
onde será criada a Geofencing
            val database = Firebase.database
            val reference = database.getReference("reminders")
            val key = reference.push().key
            if (key != null) {
               val reminder = Reminder(key, latlng.latitude, latlng.longitude)
               reference.child(key).setValue(reminder)
            }
            createGeoFence(latlng, key!!, geofencingClient)  // Cria a geofencing com os
valores armazenados
         }
      }

      // função para criar a Geofencing
      private fun createGeoFence(location: LatLng, key: String, geofencingClient:
GeofencingClient) {
         // seta os valores como, latitude, longetude, tamanho, etc. E começa a
construir a geofencing
         val geofence = Geofence.Builder()
            .setRequestId(GEOFENCE_ID)
            .setCircularRegion(location.latitude, location.longitude,
GEOFENCE_RADIUS.toFloat())
            .setExpirationDuration(GEOFENCE_EXPIRATION.toLong())
            .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or
Geofence.GEOFENCE_TRANSITION_DWELL)
            .setLoiteringDelay(GEOFENCE_DWELL_DELAY)
            .build()
         // trata da requisição de criação da geofencing
         val geofenceRequest = GeofencingRequest.Builder()
            .setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
            .addGeofence(geofence)
            .build()
```

```
        val intent = Intent(this, GeofenceReceiver::class.java)
            .putExtra("key", key)
            .putExtra("message", "Geofence alert - ${location.latitude},
${location.longitude}")

        val pendingIntent = PendingIntent.getBroadcast(
            applicationContext, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT
        )

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            if (ContextCompat.checkSelfPermission(
                    applicationContext,
Manifest.permission.ACCESS_BACKGROUND_LOCATION
            ) != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(
                    this,
                    arrayOf(
                        Manifest.permission.ACCESS_BACKGROUND_LOCATION
                    ),
                    GEOFENCE_LOCATION_REQUEST_CODE
                )
            } else {
                geofencingClient.addGeofences(geofenceRequest, pendingIntent)
            }
        } else {
            geofencingClient.addGeofences(geofenceRequest, pendingIntent)
        }
    }
    // verifica se as permissões foram concedidas
    private fun isLocationPermissionGranted() : Boolean {
        return ContextCompat.checkSelfPermission(
            this, Manifest.permission.ACCESS_FINE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED ||
ContextCompat.checkSelfPermission(
            applicationContext, Manifest.permission.ACCESS_COARSE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED
    }
    // Verificação dos resultados das solicitações de permissão
    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        //verifica a permissão de localização em segundo plano, se não tiver sido
concedida mostra um aviso
        if (requestCode == GEOFENCE_LOCATION_REQUEST_CODE) {
```

```
        if (permissions.isNotEmpty() && grantResults[0] !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(
                this,
                "Esta aplicação necessita de Localização em segundo plano para
funcionar em dispositivos Android 10+",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
    // continua a verificação das permissões
    if (requestCode == LOCATION_REQUEST_CODE) {
        if (
            grantResults.isNotEmpty() && (
                grantResults[0] == PackageManager.PERMISSION_GRANTED ||
                    grantResults[1] == PackageManager.PERMISSION_GRANTED)
        ) {
            if (ActivityCompat.checkSelfPermission(
                    this,
                    Manifest.permission.ACCESS_FINE_LOCATION
                ) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(
                    this,
                    Manifest.permission.ACCESS_COARSE_LOCATION
                ) != PackageManager.PERMISSION_GRANTED
            ) {
                return
            }
            map.isMyLocationEnabled = true
            onMapReady(map)
        } else {
            Toast.makeText(
                this,
                "Este aplicativo necessita de permissões de localização para funcionar",
                Toast.LENGTH_LONG
            ).show()
        }

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            if (grantResults.isNotEmpty() && grantResults[2] !=
PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(
                    this,
                    "Esta aplicação necessita de Localização em segundo plano para
funcionar em dispositivos Android 10+",
                    Toast.LENGTH_LONG
                ).show()
```

```
            }
        }
    }
}

companion object {
    // tratativa para remoção das geofencings
    fun removeGeofences(context: Context, triggeringGeofenceList:
MutableList<Geofence>) {
        val geofenceIdList = mutableListOf<String>()
        for (entry in triggeringGeofenceList) {
            geofenceIdList.add(entry.requestId)
        }

LocationServices.getGeofencingClient(context).removeGeofences(geofenceIdList)
    }
    // tratativa para envio de notificações ao entrar em uma geofencing
    fun showNotification(context: Context?, message: String) {
        val CHANNEL_ID = "REMINDER_NOTIFICATION_CHANNEL"
        var notificationId = 1589
        notificationId += Random(notificationId).nextInt(1, 30)

        //construtor da notificação
        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)  // define a
prioridade da notificação

        // chama o gerenciador de notificações do android
        val notificationManager =
context.getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager

        // Cria o canal de notificação e define o nome da mesma
        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())
    }
}
// define o agendador de trabalhos e chama o JobScheduler do Android
private fun scheduleJob() {
    val componentName = ComponentName(this, ReminderJobService::class.java)
    val info = JobInfo.Builder(321, componentName)
        .setRequiresCharging(false)
        .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)
        .setPersisted(true)
        .setPeriodic(15 * 60 * 1000)
        .build()

    val scheduler = getSystemService(JOB_SCHEDULER_SERVICE) as
JobScheduler
    val resultCode = scheduler.schedule(info)
    // Verifica se o Job foi criado ou não e envia um aviso
    if (resultCode == JobScheduler.RESULT_SUCCESS) {
        Log.d(TAG, "Job scheduled")
    } else {
        Log.d(TAG, "Job scheduling failed")
        scheduleJob()
    }
}
// função para cancelar o Job criado quando solicitada
private fun cancelJob() {
    val scheduler = getSystemService(JOB_SCHEDULER_SERVICE) as
JobScheduler
    scheduler.cancel(321)
    Log.d(TAG, "Job cancelled")
}


}
```

9.4 *Reminder.kt*

```
package com.codemave.geofencing

// Inicialização de algumas variáveis importantes
data class Reminder(
```

```
    var key: String = "",
    var lat: Double = 0.0,
    var lon: Double = 0.0
)
```

9.5 *ReminderJobService.kt*

```
package com.codemave.geofencing

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.job.JobParameters
import android.app.job.JobService
import android.content.Context
import android.os.Build
import android.util.Log
import androidx.core.app.NotificationCompat
import kotlin.random.Random

private val TAG = ReminderJobService::class.java.simpleName

class ReminderJobService : JobService() {
    var jobCancelled = false

    override fun onStartJob(params: JobParameters?): Boolean {
        Log.d(TAG, "Job started")
        doBackgroundWork(params)
        return true
    }

    private fun doBackgroundWork(params: JobParameters?) {
        Thread {
            kotlin.run {
                if (jobCancelled) {
                    return@Thread
                }
                showNotification(applicationContext, "Reminder job service scheduler")
                jobFinished(params, true)
            }
        }.start()
    }

    override fun onStopJob(params: JobParameters?): Boolean {
        Log.d(TAG, "Job cancelled before completion")
        jobCancelled = true
```

```
        return true
    }

    fun showNotification(context: Context?, message: String) {
        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())
    }

}
```

9.6 *Activity_maps.xml*

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

```
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```

9.7 *strings.xml*

```
<resources>
    <string name="app_name">Geofencing</string>
    <string name="title_activity_maps">GeoDrive</string>
    <string name="map_id">34f901b6a0527da5</string>
    <string name="firebase_db_url">https://tcc2-360423-default-
rtdb.firebaseio.com/</string>
</resources>
```

9.8 *build.gradle (Project)*

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
buildscript {
    ext.kotlin_version = "1.4.30"
    repositories {
        google()
        mavenCentral()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.2.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
        classpath 'com.google.gms:google-services:4.3.13'
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
        jcenter()

    }
```

```
}

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

8.9 *gradle.build (Module)*

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'com.google.gms.google-services'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.codemave.geofencing"
        minSdkVersion 21
        targetSdkVersion 30
        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 "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    implementation 'com.google.android.gms:play-services-location:17.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    // Firebase
    implementation platform('com.google.firebase:firebase-bom:30.2.0')
    implementation 'com.google.firebase:firebase-analytics-ktx'
    implementation 'com.google.firebase:firebase-database-ktx'
}
```