PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA POLITÉCNICA

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

O USO DE GEOFENCING NA BUSCA DE PRODUTOS E FARMÁCIAS

FELIPE PEREIRA BRANDÃO

GOIÂNIA-GO

2022

FELIPE PEREIRA BRANDÃO

O USO DE GEOFENCING NA BUSCA DE PRODUTOS E FARMÁCIAS

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

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

GOIÂNIA

2022

O USO DE GEOFENCING NA BUSCA DE PRODUTOS E FARMÁCIAS

Este trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, em____/____/_____.

_____
Prof. Ma. 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. Me. Carlos Alexandre Ferreira de Lima

_____
Prof. Dr. Fábio Barbosa Rodrigues

GOIÂNIA
2022

AGRADECIMENTOS

A Deus pela minha vida por todas as bençãos que ele provê.

Especialmente a meus pais, por estarem sempre comigo nessa caminhada, que me ajudaram e me incentivaram, sempre se esforçando para serem sempre meus maiores exemplos.

Aos meus amigos e colegas de faculdade, que sempre me apoiaram e me ajudaram nos momentos de dificuldade, que me incentivaram e me fizeram sempre continuar.

Ao meu Professor Me. Marcelo Antonio Adad Araújo por ter aceitado me acompanhar nessa jornada, graças ao seu apoio, seus ensinamentos passados, sua confiança e empenho, esse trabalho pôde ser realizado.

Aos professores do Curso de Engenharia da Computação que me ensinaram e me deram todo conhecimento necessário durante esses anos para a realização deste trabalho, agradeço com profunda gratidão pelo profissionalismo.

A todas as pessoas que influenciaram direta e indiretamente em minha caminhada acadêmica.

“Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.”

-Alan Turing

RESUMO

A busca de produtos em farmácias usando *geofences* visa levar aos usuários de smartphones *Android* uma maneira fácil e rápida de encontrar medicamentos e outros produtos em locais próximos. Com o auxílio do sistema GPS, o aplicativo desenvolvido localiza dentro de um perímetro virtual baseado em sua localização, mostrando ao usuário uma lista de estabelecimentos e seus produtos, onde o cliente pode escolher em qual lugar realizar suas compras. Com isso, esse aplicativo busca levar facilidade e comodidade, fornecendo informações rápidas na palma da mão.

Palavras-Chave: *Andriod*, *Geofencing*, GPS, Farmácia.

ABSTRACT

Product search in pharmacies using geofences aims to give Android smartphone users an easy and fast way to find medicines and other products in nearby locations. With the help of the GPS system, the developed application locates within a virtual perimeter based on its location, showing the user a list of establishments and their products, where the customer can choose where to make their purchases. With this, this application seeks to bring ease and convenience, providing quick information in the palm of your hand.

Keywords: Andriod. Geofencing. GPS. Pharmacy

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

Prof(a).	Professor(a)
M.E.E	Mestre em Engenharia Elétrica
Ma.	Mestra
Me	Mestre
Dr.	Doutor
km	Quilômetros

LISTA DE SIGLAS

API	*Application Programming Interface*
APP	Aplicativos
DD	Graus Decimais
DMS	Graus, minutos, segundos
FP	Funcional Paradigma de Programação
GPS	*Global Positioning System*
IDE	Ambiente de Desenvolvimento Integrado
IoT	Internet das Coisas
JVM	Máquina virtual Java
JSON	*JavaScript Object Notation*
NoSQL	*Not Only Structured Query Language*
OO	Orientação a Objetos
SGBD	Sistemas Gerenciadores de Banco de Dados
SQL	*Structured Query Language*
SDK	*Software Development Kit*
TCC2	Trabalho de Conclusão de Curso para o segundo semestre
TCC	Trabalho de Conclusão de Curso
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO

A busca por inovações tecnológicas está presente na humanidade desde a Revolução Industrial. Essa busca faz com que tecnologias sejam desenvolvidas objetivando sempre resolver problemas no cotidiano social. No passar dos anos, percebe-se o quanto a humanidade se tornou apta a utilizar e manusear novas tecnologias, procurando sempre melhorar o que já existe e criar novas formas de facilitar o dia a dia (LUIZ, 2008).

Com a chegada do século XXI aconteceu o que se chama de a “Sociedade da Informação”, onde cada um pode ter na palma de sua mão o que antes era praticamente impossível de se obter: conhecimento ilimitado, informação a pronto uso, comodidade e facilidades a toda hora (LUIZ, 2008).

É comum encontrar nos aparelhos celulares mais atuais uma infinidade de aplicações que facilitam a vida dos usuários nas mais diversas áreas. O uso da tecnologia traz grandes inovações a todo momento. Tais inovações impactam na convivência em sociedade, onde muitos tentam se adequar às novidades tecnológicas buscando sempre avançar numa corrida pela informação (LUIZ, 2008).

Com isso em mente, pode-se ressaltar que, quanto mais fácil a informação chega no usuário, mais tranquilo ele fica em relação às suas rotinas, sendo assim, diversos aplicativos e sistemas foram criados e elaborados, sendo um deles, o sistema de geolocalização via satélite., (FAGGIAN, 2019).

O GPS (*Global Positioning System* –Sistema de Posicionamento Global), um dos sistemas mais populares nos dias atuais, é talvez por ser o primeiro a garantir uma precisão significativa para uso civil, estando à mão de qualquer pessoa com um smartphone moderno (FAGGIAN, 2019).

O uso desse sistema de geolocalização permite encontrar qualquer posição no globo terrestre, o que garante conforto, segurança e precisão para os usuários que precisam se localizar, ou fazer uma viagem. Esse sistema, somado com a praticidade dos aparelhos celulares, traz uma tranquilidade aos usuários, que podem ter sempre um mapa à mão (ZANOTTA, 2011).

Aproveitando o quão impactante os sistemas de GPS são para a humanidade, novos sistemas sempre aparecem para complementar suas aplicações, uma delas é a possibilidade de delimitar perímetros virtuais, sistema

que recebe o nome de *Geofencing*. Essas cercas virtuais, permitem que decisões sejam tomadas e padrões sejam mapeados nos dispositivos, sendo assim, a *geofence* é uma grande inovação quando voltada para o comércio eletrônico, estratégias de marketing, além de outras funções voltadas para segurança (WHITE, 2017).

A indústria farmacêutica é um pilar importante na construção científica da sociedade em que se vive, e como é um setor com fins lucrativos, tem num de seus objetivos a promoção e venda de seus produtos da melhor forma possível e com maior abrangência territorial concebível. Nisso surge a necessidade do marketing, que é definido como uma função dos negócios que reconhece as necessidades dos clientes, analisando-as, de modo que possa se retirar a rentabilidade (AGOSTINHO, 2015).

Este trabalho de conclusão de curso tem por finalidade o desenvolvimento de um aplicativo *mobile* que faça a busca de farmácias e os produtos que elas oferecem, utilizando para esse fim, a tecnologia de cercas virtuais conhecida como *geofence*. Essa tecnologia permite, por meio de geolocalização do usuário, criar ao seu entorno uma área que pode ser controlada pelo mesmo. A busca consistirá em vasculhar no banco de dados, os produtos disponíveis e oferecer ao usuário a localização da farmácia, a distância entre eles, o valor do produto e suas características.

O aplicativo desenvolvido guarda as informações sobre as farmácias e seus produtos em um banco de dados. Utilizando a tecnologia de geolocalização, é possível encontrar qualquer ponto no globo terrestre, isso faz com que o aplicativo possa encontrar qualquer loja cadastrada no banco de dados em qualquer lugar da terra, desde que esteja dentro da área da *geofence*. Esta aplicação foi desenvolvida utilizando a plataforma *Android Studio*. Essa plataforma oferece suporte para várias linguagens de programação, entretanto, o desenvolvimento do aplicativo descrito foi feito em *Kotlin*.

1.1 Objetivos

Esta seção descreve os objetivos gerais e específicos do presente trabalho.

1.1.1 Objetivo Geral

O objetivo geral deste Trabalho de Conclusão de Curso, é desenvolver um

sistema que utilize *Geofencing* para realizar buscas por farmácias e produtos, levando informações ao usuário sobre os itens pesquisados.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Utilizar de *geofencing* na busca de melhores ofertas na proximidade,
- Mostar todas as lojas cadastradas dentro da *geofence* determinada,
- Realizar a comparação de preço e distância, informando o usuário asfarmácias próximas,
- Desenvolver um sistema que facilite ao usuário o momento de suprir suas necessidades de compras.

1.2 Justificativa

O presente trabalho utiliza uma nova forma de localização e busca de facilidades na área farmacêutica. Aproveitar da utilização do smartphone no ambiente diário de todas as pessoas, a construção desse aplicativo visa comunicar ao usuário de uma forma simples, em que a *geofencing* pode ser empregada nos diversos usos, inclusive na aplicação específica do presente trabalho.

As possibilidades de ganho funcionam tanto para os usuários que visam buscar e encontrar farmácias e produtos, quanto para as próprias farmácias, que podem aproveitar as criações dos perímetros virtuais para criar uma competição saudável entre as concorrentes, criando ofertas e divulgando promoções.

A busca de melhoria na vida e comodidade das pessoas hoje em dia é vista como um principal foco para os desenvolvedores de aplicativos, sempre inovando para que o usuário tenha a melhor experiência.

1.3 Procedimentos metodológicos

Inicialmente será realizada uma revisão bibliográfica, baseadas em artigos científicos, monografias, dissertações de mestrado, teses de doutorado, livros, revistas e outros meios de informação. A segunda parte será a construção dos requisitos para a criação da aplicação e implementação do software. Por fim,

serão conduzidos testes de qualidade e controle, onde será usado o *feedback* adquirido para implementar melhorias e novas funções. Após isso, implementar-se-á o aplicativo, o qual é desenvolvido por meio do *Android Studio.*

1.4 Resultados esperados

Espera-se que os resultados deste trabalho possam auxiliar:

- Com a praticidade dos celulares, encontrar farmácias que atendam os desejos e necessidades do usuário, de forma rápida.
- Identificar possíveis *geofences* numa área onde se localizam várias farmácias o que trará ao usuário uma melhor visualização de uma área caso queria percorrer por si só entre farmácias.
- Listar produtos para facilitar ao usuário a comparação de preço e distância, o que trará mais praticidade.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém a fundamentação teórica necessária para a elaboração deste trabalho, com conceitos e definições importantes para uma melhor compreensão desse projeto, de forma a incluir pesquisas bibliográficas relacionadas ao tema do presente trabalho de conclusão de curso.

2.1 Sistema de Posicionamento Global

O Sistema de Posicionamento Global, popularmente conhecido como GPS pelasua praticidade, tornou-se indispensável para a sociedade atual. A precisão e a rapidez com que um receptor GPS determina a posição de um ponto localizado no globo terrestre, tem sido de grande utilidade para a navegação (terrestre, marítima e aérea), cartografia, geodinâmica, entre tantos procedimentos em áreas científicas, comerciais e outras que necessitem de posicionamento (ZANOTTA, 2011).

O GPS foi desenvolvido pelo Departamento de Defesa dos Estados Unidos e sua utilização se restringia a uso militar. Posteriormente, tornou-se disponível para uso civil. O presente sistema funciona com base em 32 satélites, com pelo menos 24 deles em operação, distribuídos ao redor do globo em 6 planos orbitais e com altitude de aproximadamente 20.200 km, como mostrado na Figura 1 (FAGGIAN, 2019).

Figura 1 – Disposição dos satélites ao redor do planeta

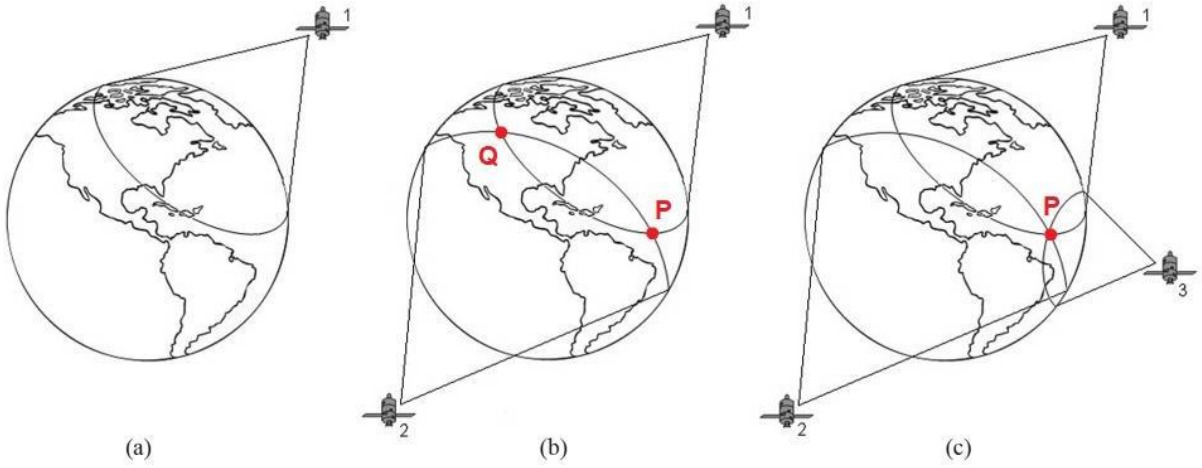
Fonte: Adaptado de Dilião, 2014.

Para serem determinados quaisquer pontos da superfície terrestre com

precisão, são necessários que pelo menos quatro satélites, tendo em vista que o quarto satélite serve apenas para sincronização de horário, tenham visão da área em questão e não compartilhem o mesmo plano orbital. Os satélites têm suas rotas monitoradas por bases fixas terrestres e simultaneamente emitem sinais codificados que fornecem dados como posição, horário e código identificador (FAGGIAN, 2019).

Com a informação de um único satélite, o usuário poderia saber somente que está localizado num determinado círculo sobre a superfície terrestre, como ilustrado na Figura 2. Se mais um satélite for adicionado ao cálculo, o receptor se torna capaz de definir a posição de dois pontos possíveis de localização (P e Q, conforme a Figura 2b). Tendo as informações registradas simultaneamente por três satélites, um receptor pode finalmente definir apenas um ponto (P) localizado na superfície da terra(Figura 2c) (ZANOTTA, 2011).

Figura 2 – Cones de atuação de cada satélite formando os pontos de intersecção: a) Apenas um satélite, b) Dois satélites, c) Três satélites.



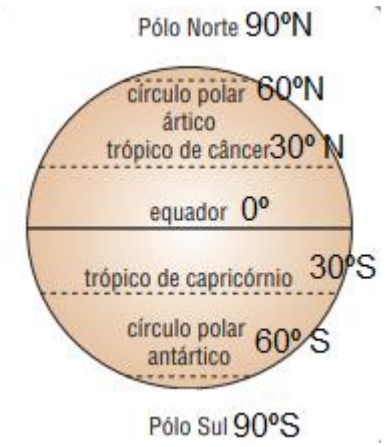
Fonte: Adaptado de Zanotta, 2011.

O GPS utiliza um sistema de coordenadas geográficas que fazem uso de linhas imaginárias que subdividem o globo terrestre baseadas nas linhas do Equador e no Meridiano de *Greenwich*. Isso torna possível a utilização de códigos geográficos denominados de latitude e longitude (MACHADO, 2015).

A Latitude tem como referência a linha do Equador. Ela é a linha imaginária que está localizada na porção central da esfera terrestre, mais especificamente no centro da zona climática tropical e é perpendicular ao eixo de rotação do globo. Paralelo ou paralelo geográfico é todo o círculo menor perpendicular ao eixo da

Terra e, portanto, paralelo ao Equador. Sobre um determinado paralelo (trópico de câncer, círculo polar ártico, trópico de capricórnio e círculo polar antártico por exemplo), a latitude é constante. Sobre o Equador, a latitude é igual a zero graus, medindo-se de 0º a 90º, para norte (positiva) e 0º a -90º para sul (negativa) demostrado na figura 3 (CARVALHO, 2008).

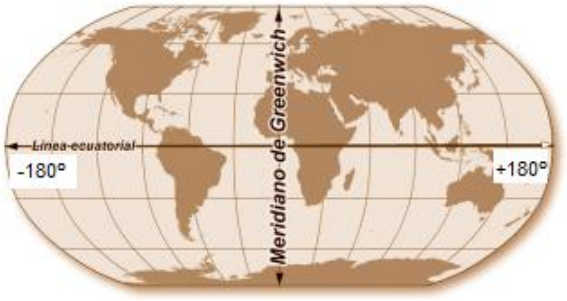
Figura 3. Paralelos representados no globo.



Fonte: Adaptado de Carvalho, 2008.

A longitude também é uma linha de referência, neste caso o primeiro meridiano ou Meridiano de *Greenwich* localizado no globo terrestre na posição vertical. As linhas imaginárias posicionadas verticalmente (meridianos) determinam a longitude, que é definida como a distância em graus de qualquer ponto da superfície terrestre até o primeiro meridiano ou Meridiano de *Greenwich*. A longitude varia de 0° (no Meridiano de Greenwich) a 180° para leste, e de 0º a -180° para oeste como apresentado na figura 4 (CARVALHO, 2008).

Figura 4. Representação do Meridiano de Greenwich na terra.



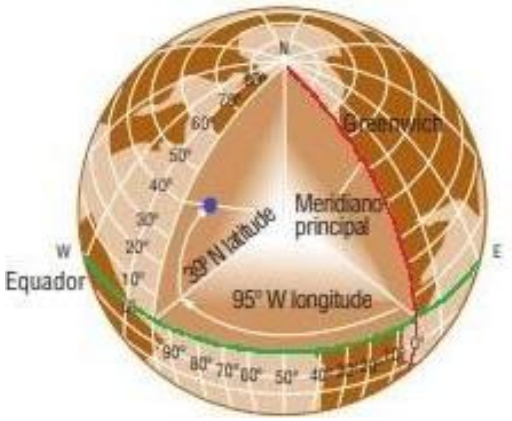
Fonte: Adaptado de Carvalho, 2008.

Um sistema de coordenadas geográficas é um sistema de referência usado

para posicionar e medir aspectos geográficos. As posições do mundo real são medidas em graus de longitude e latitude. Os valores podem ser positivos ou negativos dependendo do seu quadrante no globo. Como unidade de medida, cada grau é composto de 60 minutos e cada minuto é composto de 60 segundos. As medidas são em graus, minutos e segundos que podem ser representados por DMS ou em graus decimais que podem ser representados por DD. Por exemplo, 34º 30’00” é igual a 34,5º só em graus. Os valores de longitude variam de 0º até 180º tanto a leste (+) quanto a oeste (–) começando no Meridiano de Greenwich. Os valores de latitude variam de 0º até 90º no Hemisfério Norte, indo do Equador até o Polo Norte. No Hemisfério Sul, a latitude varia de 0º até -90º, indo do Equador até o Polo Sul (CARVALHO, 2008).

As coordenadas geográficas localizam, de forma direta, qualquer ponto sobre a superfície terrestre, sendo necessário apenas indicar o hemisfério: N (norte) ou S (sul); E (leste) ou W (oeste), na figura 5 está um exemplo usando as coordenadas 95º 25’13” W e 39º 33’13” N (CARVALHO, 2008).

Figura 5. Longitude = 95º 25’13” W e Latitude = 39º 33’13” N, ponto azul.



Fonte: Adaptado de Carvalho, 2008.

2.2 *Geofencing*

O sistema que permite utilizar a localização geoposicionada do usuário em um aplicativo, a partir de dados móveis obtidos por telefonia, GPS, Wi-Fi e outros tipos de tecnologias é chamado de *geofencing.* Esse sistema faz com que ações sejam tomadas quando o usuário adentra uma cerca virtual configurada com base em coordenadas geográficas denominada *geofence* (WHITE, 2017).

Uma *geofence* não serve apenas para determinar uma área num

determinado mapa. Algumas *geofences* podem ser configuradas a monitorar as atividades, das mais diversas, em áreas seguras, permitindo que o sistema de gerenciamento seja capaz de perceber quando algum dispositivo esteja nas imediações de uma área específica. As organizações também podem usar *geofencing* para monitorar funcionários em campo, automatizar cartões de ponto e controlar a propriedade da empresa (WHITE, 2017).

O sistema funciona baseado em uma localização na qual um perímetro virtual, cerca virtual, é criado e delimitado por meio de coordenadas geográficas disponíveis em um dispositivo que possua o sistema GPS. Essa área, conhecida como*geofence* é transmitida para os usuários para que possam tomar decisões e fazer algumas ações baseadas na sua finalidade (WHITE, 2017).

2.3 Sistema Operacional *Android*

Os sistemas operacionais móveis estão presentes em todos os *smartphones*. Os sistemas operacionais têm grande importância na tecnologia atual, assim, os celulares, *tablets, ipads*, entre outros dispositivos móveis, conseguem desempenhar suas funções de acordo com as necessidades do usuário (ALMEIDA, 2014).

Sistema operacional móvel é um conjunto de programas com a função de gerenciar os recursos, e fazer a comunicação entre *hardware* e *software* para dispositivos móveis, também é desenvolvido para fornecer uma interface amigável aousuário final (ALMEIDA, 2014).

“O Android teve o seu início no ano de 2003 com o foco voltado para as câmeras digitais, porém, com a baixa demanda de mercado, esse sistema mudou para o mercado de telefonia móvel. Esta plataforma teve como base o sistema operacional Linux” (FAUSTINO, 2017).

O sistema operacional *Android* foi desenvolvido por Andy Rubin, Rich Miner, Nick Sears e Chris White no intuito de montar uma empresa, a *Android Inc*. A ideia inicial era criar de um sistema gratuito para o público e de fácil compreensão para osprogramadores, então eles usaram como base de utilização o *kernel* do sistema operacional Linux. O foco principal do grupo começou no ramo das câmeras digitais, mas devido à escassez do mercado naquela época, migraram para o mercado dos dispositivos móveis, essa escassez se deu ao fato do preço elevado do equipamento e do nicho de mercado ser pequeno

(FAUSTINO, 2017).

O Linux, no qual foi baseado o sistema *Android,* é um sistema operacional de código aberto (*open source*). Isto significa que ele pode ser modificado e distribuído por qualquer pessoa. Uma de suas vantagens é a livre distribuição, ou seja, não é necessário pagar para obter nenhuma de suas inúmeras versões (FAUSTINO, 2017).

A maior característica do *open source* é o livre acesso aos códigos fontes dos softwares, possibilitando aos usuários o acesso, edição e cópias dos códigos disponíveis, além das suas atualizações (FAUSTINO, 2017).

2.4 Linguagem de programação

Para implementar um algoritmo em um computador, é necessário descrevê-lo de uma forma que o computador esteja apto a executar o que foi proposto. Isso é feito por intermédio de uma “linguagem de programação”. O próprio conjunto de instruções de um processador pode ser entendido como uma “linguagem de programação”. Foram desenvolvidas, ao longo da história, diversas linguagens de programação, cada qual, a seu tempo, introduzindo facilidades e recursos que foram tornando a tarefa de programar mais fácil e menos susceptível a erros (GUDWIN, 1997).

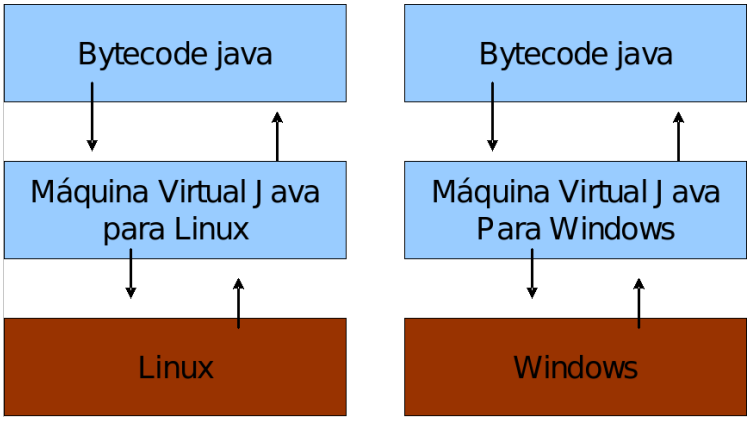
2.4.1 Java

Java é a linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems, capaz de criar aplicativos para *desktop*, aplicações comerciais, softwares robustos, completos e independentes, aplicativos para a Web. Java é muito parecida com C e C++, Java é interpretada enquanto C e C++ são compiladas, ela caracteriza-se por eliminar atributos considerados complexos (CLARO, 2008).

A principal característica do Java é ser multiplataforma. A linguagem Java é interpretada para um “*bytecode”*, esse código é executado por uma máquina virtual Java (JVM) posteriormente. Essa máquina virtual atua como um interpretador. Ele carrega o *bytecode*, interpreta esse código para se adequar ao computador que está sendo usado e depois executa a aplicação. Tudo isso é feito muito rápido, é como se a aplicação fosse nativa do sistema como mostra a figura 6 (SOUSA,

2017).

Figura 6: Conceito de compilação Java



Fonte: CAELUM, 2016

2.4.2 Kotlin

Kotlin é uma linguagem de programação que pode ser interpretada utilizando o código-fonte do *JavaScript*. Essa linguagem foi lançada ao público em fevereiro de 2016. Uma equipe de programadores do grupo *JetBrains* desenvolveu e criou sua primeira versão, sua sede, situada em São Petersburgo, Rússia. Na verdade, o nomeKotlin vem da Ilha de Kotlin em São Petersburgo (BANERJEE, 2018).

Em maio de 2018, a equipe do *Google Android* anunciou que a linguagem Kotlin é a linguagem oficial para o desenvolvimento *Android*. Os desenvolvedores têm usado Kotlin para construir aplicativos *Android* em anos anteriores, mas o *Google* anunciou um suporte para desenvolvimento em Kotlin (BANERJEE, 2018).

Kotlin é uma linguagem multiparadigma, isso é, ela compatível com Orientação a Objetos (OO) e Funcional Paradigma de Programação (FP). Com isso, o desenvolvedor pode fazer o uso da OO e FP, ou simplesmente combiná-los, como acontece na maioria das linguagens modernas da indústria de software (OLIVEIRA, 2019).

O uso da linguagem de programação Kotlin, possui as características segundosua documentação.

- Menor quantidade de código combinado com maior legibilidade. Isso garante que se passe menos tempo escrevendo os códigos e entendendo os códigos de outros desenvolvedores;
- Linguagem e ambientação madura. Kotlin tem se desenvolvido

continuamente e com isso, garante uma ferramenta de desenvolvimento completa. Isso faz com que seja usado por muitas empresas para desenvolver aplicativos Android;

- Interoperabilidade com Java. Garante que não seja necessário alterar os códigos Java para Kotlin, o que facilita a criação de um aplicativo desenvolvidoem várias linguagens.
- Segurança do código. Como possui menos códigos, isso leva ao desenvolvedor uma forma mais simples de leitura. Essa legibilidade melhor garante uma redução nos erros;
- Aprendizagem fácil. A linguagem Kotlin é considerada por muitos de fácil entendimento (KOTLIN, 2021).

2.5 Banco de dados

Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e que possuem um significado implícito. Um banco de dados possui fontes nas quais os dados são adquiridos, esses dados contêm níveis de interação com os eventos do mundo real e um público que, está ativamente interessado em seus conteúdos (ELMASRI, 2005).

É comum o uso de Sistemas Gerenciadores de Bancos de Dados (SGBD), sãoferramentas que permitem aos usuários criar e manter um banco de dados. O SGBD é um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados. A definição de um banco de dados implica em especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados (ELMASRI, 2005)

Outras funções importantes do SGBD são a proteção e a manutenção do bancode dados. Essa proteção inclui:

- Proteger o sistema contra o mau funcionamento ou falhas no hardware ou software;
- Verificar a segurança contra acessos não autorizados ou maliciosos;
- Capacidade de manter um sistema de banco de dados que permita a evolução dos requisitos que se alteram ao longo do tempo (ELMASRI, 2005).

Os sistemas SGBD são divididos em dois diferentes tipos, o sistema de

gerenciamento do banco de dados relacional, ou Banco de dados *Structured Query Language* (SQL), e o sistema de gerenciamento de banco de dados não-relacional conhecido também como *Not Only Structured Query Language* (NoSQL) (ROVEDA, 2021)

Com a finalidade de fazer a relação dos dados que o mundo real oferece, foi criado na década de 70 o Banco de Dados Relacional. Esse modelo é uma forma intuitiva e direta de representar dados em tabelas, essas podem ser chamadas de relações. Cada coluna nessa tabela representa um campo diferente de dados e informações. Esse modelo conta também com o processo de Normalização, que é um conjunto de regras específicas para cada tabela. Essas regras facilitam o armazenamento, tornando-o consistente, o que aumenta a eficiência ao acessar os dados, reduzindo as chances de ocorrer redundâncias e de obter dados inconsistentes. Com a construção desse modelo, a IBM, criou a linguagem para a manipulação desses bancos de dados, conhecidas como *Structured Query Language* (SQL) (MARILIA, 2012).

Os desenvolvedores notaram que o modelo relacional foca nos relacionamentos entre entidades, que são os campos correlacionados ao banco de dados, e isso torna mais “burocrática” a implementação de novas funcionalidades, além disso, o principal problema que o modelo relacional encontra é atender a demanda da escalabilidade, que consiste em o quão bem o modelo consegue se adaptar para não ter queda de desempenho quando um número maior de clientes utiliza o banco de dados ou acontece algum cenário inesperado (MARILIA, 2012).

Pensando em solucionar o problema com escalabilidade, performance e disponibilidade (facilidade de utilização), foi desenvolvido um banco de dados não relacional. Esse modelo conta com o modelo relacional como base, e sua característica é eliminar a necessidade de certas regras estruturadas. Com a quebra dessas regras, houve ganho de performance flexibilizando os sistemas para melhorar disponibilidade dos dados para cada aplicação. A linguagem utilizada para implementação desse banco de dados é derivada do modelo relacional, e recebe o nome de *Not Only Structured Query Language* (NoSQL) (MARILIA, 2012).

2.6 *Application Programming Interface*

API sigla em inglês para *Application Programming Interface* que significa Interface de Programação de Aplicações. Uma API permite que sistemas se comuniquem com outros serviços sem a necessidade de implementar uma na outra. Essa simplificação acelera o desenvolvimento de aplicações, garantindo economia de tempo e dinheiro (REDHAT, 2021).

De acordo com o RedHat, Inc.:

> “Com as APIs, você libera o acesso aos seus recursos sem abrir mão da segurança de controle. É possível conectar APIs, bem como criar aplicações que fazem uso dos dados ou funcionalidades disponibilizadas por elas, usando uma plataforma de integração distribuída que ligue todos os elementos, incluindo sistemas legados e dispositivos de Internet das Coisas (IoT)” (REDHAT, 2021).

2.7 Marketing

O uso das estratégias digitais em campanhas de comunicação e marketing dasempresas se diversifica no uso sistemático de ferramentas, como a integração da mobilidade e portabilidade dos aparelhos celulares. Com o acesso fácil à internet e com a otimização dos sistemas de busca, instituiem-se novas formas de comunicação integrada (OKADA, 2011).

Marketing é o processo de planejar e executar a concepção, estabelecimento de preços, promoção e distribuição de ideias, produtos e serviços a fim de criar trocas que satisfaçam metas individuais e organizacionais (CHURCHILL, 2017).

O uso do Marketing de busca, visa aumentar o potencial de uma empresa quando se trata de expandir sua autonomia. Isso proporciona uma audiência em sua grade de ofertas e serviços, com isso, pode-se maximizar a visibilidade da empresa através daInternet e aumentar o a taxa de conversão dos visitantes em clientes (OKADA, 2011).

2.8 *Google Firebase*

O Google desenvolveu a plataforma de *Google Firebase* que foi lançada em 2016. Essa plataforma oferece inúmeras ferramentas para o desenvolvimento de aplicativos para Android, iOS e aplicativos web. São ferramentas e serviços

como mensagens, banco de dados, autenticação de segurança, além de vários outros (MORONEY, 2017).

Atualmente vem se desenvolvendo banco de dados *Firebase* que é um dos bancos de dados *Not Structured Query Language* (*NoSql*). O *Firebase* é um dos bancos de dados em tempo real adequado para aplicativos que exigem atualizações de dados. O desenvolvimento de um banco de dados pelo *Google Firebase* visa facilitar o desenvolvimento de aplicativos. O *Firebase Realtime Database* é um banco de dados *NoSql* baseado em nuvem que sincroniza dados em todos os clientes em tempo real. Os dados são analisados em tempo real do Firebase são armazenados como *JavaScript Object Notation* (*JSON*) que é um formato compacto de troca de dados simples e rápida entre sistemas, que utiliza texto legível a humanos, no formato atributo-valor(SUDIARTHA, 2019).

Nos tópicos a seguir serão apresentados alguns dos serviços e produtos que o Google Firebase oferece para desenvolvimento.

2.8.1 Realtime Database

É um banco de dados em *Not Only Structured Query Language (NoSQL)*, ou seja, não utiliza tabelas com linhas e colunas para armazenamento dos dados. Mesmo sendo um banco de dados hospedado na nuvem, ele é capaz de guardar os dados offline, armazenando dados na memória do dispositivo que está sendo usado e atualizando esses dados assim que o dispositivo se reconectar à internet. Esse banco de dados funciona definindo regras de como os dados são estruturados e quais usuários podem acessar esses dados, chamadas de regras de segurança. (MORONEY, 2017).

2.8.2 Cloud Storage para Firebase

É um serviço de armazenamento que possui baixo custo e permite que desenvolvedores e usuários baixem e carreguem arquivos como vídeos, fotos e áudio, independentemente da qualidade da rede em que o dispositivo está (KHAWAS; SHAH, 2018).

2.8.3 Firebase Authentication

Existe uma variedade de aplicativos que exigem que os usuários se registrem e criem credenciais para usá-los e, como muitos desses aplicativos exigem muitas informações pessoais ao se registrar, podem fazer com que os usuários não queiram se registrar e não façam o uso do aplicativo. *O Firebase Authentication* foi criado visando resolver esse problema, diminuindo a dificuldade e o custo de implementação do sistema para criação e manutenção de credenciais de login.

A ferramenta disponibiliza uma API para se cadastrar e fazer login nos aplicativos que a utilizam, por exemplo, por meio de contas do Google e Facebook, ou até mesmo fazer login com e-mail e senha. Aplicativos que possuem esse tipo de autenticação e permitem que o usuário realize login com as credenciais que ele já possui estão se tornando cada vez mais populares. Esta ferramenta também pode ser integrada a outros serviços do *Google Firebase* como o *Realtime Database*, o que faz com que os dados de cada usuário sejam armazenados melhorando a forma de controle dos dados sabendo quais fontes os mesmos terão acesso (MORONEY, 2017).

2.9 *Android Studio*

Disponibilizado em 2013, o *Android Studio* é mais uma ferramenta desenvolvida pela Google. Essa ferramenta de ambiente de desenvolvimento integrado (IDE) possui poderoso editor de código que oferece suporte a recursos como edição inteligente, refatoração avançada de código e análise profunda de código estático (DEVELOPERS, 2013).

O *Android Studio* oferece também um ambiente para desenvolvimento completo. Nele, todos os modelos e conceitos encontrados em toda programação Android é descrito e documentado. Além disso, é um recurso com muitas ferramentas que potencializam a criação de códigos cada vez mais complexos (HELLMANN, 2016)

3 PROPOSTA DE SOLUÇÃO

Este capítulo apresenta a proposta de solução que foi implementada na realização e desenvolvimento desse Trabalho de Conclusão de Curso II de forma a expor as ferramentas que serão utilizadas, além de conceitos tratados anteriormente para a construção do aplicativo que se pretende implementar.

3.1 Kotlin

A escolha de utilizar a linguagem Kotlin no desenvolvimento desse aplicativo foi baseada nas vantagens que essa linguagem oferece.

Durante a fase de planejamento desse trabalho, algumas linguagens de programação foram discutidas, até se chegar à decisão de utilizar o Kotlin. Por ser uma linguagem nova, tem-se o desafio, e por ser uma linguagem derivada do Java, tem-se a experiência. Juntando esses fatores, foi pensado nessa linguagem como um alicerce que carrega legado e inovação ao mesmo tempo.

É de se esperar que como é baseado em Java, possua muitos complementos e facilitações em materiais de estudo, o que torna de certa forma, fácil de aprender ede desenvolver.

A facilidade de escrita do código e sua interoperabilidade com o Java, fazem dessa linguagem uma excelente escolha para a criação de aplicativos móveis (KOTLIN, 2021).

Diante todas as vantagens, citadas anteriormente no Capítulo 2, a linguagem Kotlin foi adotada para o desenvolvimento do código fonte do sistema proposto neste trabalho.

3.2 Android Studio (IDE)

A escolha da IDE utilizada foi pensada de forma que fosse garantido um suporte viável e adequado. Buscando sempre as vantagens que uma IDE possa oferecer para garantir que a programação não seja interrompida ou cancelada.

O Android Studio é o ambiente de desenvolvimento integrado oficial para a construção de aplicações Android. Ela oferece um acervo completo de ferramentas que são úteis e necessárias na programação. Além disso, esse ambiente ainda oferece as seguintes vantagens:

- Possui emulador rápido, ou seja, a capacidade com que ele simule um aparelho móvel, além de possuir uma grande quantidade de recursos;
- Possui um ambiente unificado que possibilita o desenvolvimento para todos osdispositivos Android;
- Proporciona a utilização de ferramentas que permitem a aplicação de mudanças de código sem reiniciar o aplicativo ou fechar o emulador;
- Oferece várias opções de Frameworks, que são estruturas genéricas do código que podem ser adicionadas no programa em desenvolvimento e ferramentas de teste;
- Possui suporte a linguagem Kotlin, que é adotada na criação do sistema proposto nesse trabalho (DEVELOPERS, 2021).

Com essas vantagens, foi escolhido a IDE Android Studio, sendo a melhor opção para a construção do aplicativo aqui proposto.

3.3 *Firebase Realtime Database*

O Google *Firebase* foi a plataforma escolhida para o desenvolvimento desse sistema. Nessa plataforma foi utilizada a ferramenta *Realtime Database* para armazenar os dados utilizados pelo aplicativo. O *Realtime Database* é um banco de dados hospedado na nuvem do tipo *NoSQL*. Seus dados são armazenados no formato JSON. A vantagem de ser baseado em nuvem é que sempre que os usuários estiverem conectados à rede, o banco de dados será sincronizado em tempo real, recebendo asso, as atualizações com os dados mais recentes (FIREBASE, 2020).

Por mais que os dados sejam armazenados em nuvem, o SDK (*Software Development Kit*) do *Realtime Database,* permite que os dados do aplicativo também sejam salvos na memória de armazenamento do smartphone, mantendo os dados do aplicativo mesmo quando não houver conexão com a Internet, e atualizando os dados no momento em que a conexão é reestabelecida (FIREBASE, 2020).

Para adicionar o *Firebase* ao desenvolvimento deve se ter instalado o programa *Android Studio* em sua versão mais recente. O aplicativo a ser desenvolvido necessita de possuir pelo menos nível 16 da API, esse nível é um identificador inteiro exclusivo que cada versão do *Android* recebe. O nível 16 trata-se *Gradle* na versão 4.1 ou suas versões mais recentes, sendo o *Gradle* uma ferramenta de automação de compilação de código aberto projetada para ser flexível

o suficiente para criar quase qualquer tipo de software. E, por fim, se faz necessário o uso de um smartphone ou de um emulador para realizar a execução do aplicativo. A conexão do aplicativo *Android* ao *Firebase* pode ser feita de duas maneiras: utilizando o próprio Console do *Firebase*, que é a opção recomendada, ou utilizando o *Firebase* Assistente do Android Studio, e neste caso poderá exigir uma configuração adicional (FIREBASE, 2021)

O intuito de utilizar um banco de dados não estruturado é facilitar a criação dos dados na nuvem.

3.4 O aplicativo

Essa seção tem por finalidade apresentar todos os componentes presentes no código-fonte do aplicativo Android, incluindo seus métodos e funcionalidades além de exemplos da interface do aplicativo desenvolvido.

3.4.1 Visão geral do aplicativo

O processo de construção do aplicativo se dá em pastas e arquivos separados que comunicam entre si. Esses arquivos são armazenados em um diretório raiz que leva o nome do projeto.

O *AndroidManifest.xml* é um arquivo que permite descrever a funcionalidade e os requisitos do aplicativo. Nele é inserido o comando *android.permission.ACCESS_FINE_LOCATION.* Essa permissão faz com que o aplicativo seja autorizado a receber os dados do receptor GPS do *smartphone* junto com o *Wi-Fi* e os dados móveis, possibilitando uma precisão maior, geralmente de 50 metros a 3 metros (DEVELOPERS, 2021c, 2021d).

O comando *android.permission.ACCESS_COARSE_LOCATION* também possibilitará que o aplicativo tenha acesso à localização do usuário, mas essa permissão faz uso apenas dos dados móveis e Wi-Fi, e assim, terá uma precisão menor, estimado em cerca de 1,6 km (DEVELOPERS, 2021c, 2021d).

Para a criação do presente trabalho foi criado um banco de dados para representar as farmácias e seus produtos. Esse banco contém 3 farmácias contendo nome, latitude e longitude e seu endereço, cada farmácia contém 5 produtos que possuem um nome popular, seus componentes e informações como

características do produtor, indicador se é genérico ou original, o tipo de seu uso e seu valor. A figura 7 mostra o banco de dados e todos seus componentes em forma de ramificações.

Figura 7 – Banco de Dados expandido mostrando todas as informações.

Fonte: elaborado pelo autor, 2022.

3.4.2 Maps Activity

O *MainActivity* é a primeira *activity* chamada, ela cria uma pequena imagem de apresentação do aplicativo, logo após, chama a atividade *MapsActivity.*

O *MapsActicity* é a principal interface do aplicativo, essa interface irá aparecer quando se executa o mesmo. Ela contém as funcionalidades e serviços

necessários a serem utilizados na implementação da *geofencing* e do *Google Maps.* Para usar o *Maps* é necessário inserir uma chave de API, que deve ser obtida por meio do registro na plataforma *Google Cloud*, que possui uma biblioteca de APIs, incluindo o SDK do Maps para Android (DEVELOPERS, 2021).

Quando criada uma nova atividade no *Android Studio*, são gerados dois arquivos. São eles o *mapsActivity.kt* que contém os métodos e funcionalidades que permitirá que sejam realizadas manipulações no mapa, possibilitando também, o controle de ações por parte do usuário (DEVELOPERS, 2021b).

O *activity_maps.xml* é o arquivo onde são encontrados os elementos de interface da tela principal do aplicativo. Nesse arquivo é encontrado um *fragment* do Google Maps. Um *fragment* se trata de uma parte da interface de usuário dentro de uma *Activity*, sendo considerado uma seção modular da mesma, na qual possui o próprio ciclo de vida e os próprios comportamentos. Ao executar o aplicativo, o *fragment* será instanciado, e irá gerar o mapa do *Google* automaticamente (DEVELOPERS, 2019).

O arquivo *google_maps_api.xml* serve para armazenar a chave de API que será utilizada no aplicativo em questão (DEVELOPERS, 2021b)

O arquivo *mapsActivity* é o ambiente de desenvolvimento principal dessa aplicação, nele, será implementado o *fragment* do Google *Maps,* e onde será implementado os métodos*.* Os métodos desenvolvidos nessa atividade são *onCreate, onResume, onMapReady,* e *onResquestPermissionResult.* Além desses, também são implementados aqui métodos e funções para uso do próprio aplicativo como *mapPoints, getTheAddress* e *geofenceMarker.*

O método *onCreate* é chamado quando o aplicativo é aberto. Nele será feita a solicitação de permissão de para receber os dados da localização. Após isso, irá ler os dados do banco de dados do *Realtime Database*, guardando em uma lista que armazena os dados. Nesse método também irá iniciar algumas variáveis, outra função importante é a inicialização do próprio mapa.

Dentro do método *onCreate* será a primeira vez em que é chamado o *addValueEventListener*. Esse método é chamado para detectar alterações que acontecem no banco de dados. É utilizado o método *onDataChange* para ler um snapshot (uma varável estática para leitura de um banco de dados em um momento específico) do conteúdo de um determinado caminho no momento do evento. Esse método será acionado uma vez quando o *listener* for anexado e

sempre que os dados forem alterados, incluindo os dos nós filhos, como apresentado na figura 7. O retorno de chamada do evento recebe um *snapshot* que contém todos os dados no local, incluindo dados dos nós filhos. O método *onCancelled* é ativado caso ocorra algum erro na leitura do banco de dados (FIREBASE, 2020c).

No método *onResume* realiza-se primeiramente uma verificação, para validar se as permissões foram aceitas, o que será necessário para utilizar os métodos seguintes. Logo após, será chamado o método *lastLocation,* que retornará a última localização em que o dispositivo esteve, após isso, utilizando a classe *LocationRequest* é criado o intervalo com que o aplicativo ficará solicitando a localização do usuário, de forma que, quanto menor esse intervalo, mais é exigido do dispositivo GPS, e com isso, maior será o consumo da bateria.

Também é definida a prioridade deste aplicativo, neste trabalho, definida como balanceada, ou seja, possuirá uma precisão moderada, de até 100 metros. Logo após obter a localização atual do usuário, também é inserido um marcador vermelho em sua localização, juntamente de um círculo, que será a representação gráfica da *geofence* (DEVELOPERS, 2021f).

A função *geofenceMarker,* é chamada sempre que um marcador personalizado é criado, para também criar a área de busca em torno da localização utilizando comandos da própria biblioteca do Google *Maps*.

A função *getTheAddress* utiliza a biblioteca do *Kotlin* chamada *Geocoder* para retornar o endereço da localização atual, apenas para deixar mais fácil a visualização da localização do usuário.

O método *onRequestPermissionsResult* é chamado para verificação das permissões sempre que o aplicativo é iniciado novamente.

O método *onMapReady* será chamado assim que o mapa inicializado no método *onCreate* estiver pronto. Nesse método faz-se as configurações do mapa, a localização do usuário já é enviada, o zoom máximo também é definido como 20f e o zoom mínimo é 5f, essas definições de zoom são classificadas da seguinte forma:

- 1f corresponde ao nível de zoom mundial,
- 5f continentes,
- 10f cidades,
- 15f ruas,

- 20f edifícios.

Nesse método, o usuário final também pode usar os métodos de manipulação do mapa, como usar gestos para diminuir ou aumentar o zoom do mapa, controle de zoom por meio de botões, inclinação e rotação do mapa e adição de um marcador personalizado, pressionando por alguns segundos em algum local do mapa. Também permite que o usuário aumente ou diminua o tamanho da *geofence* utilizando a *SeekBar* abaixo do mapa (DEVELOPERS, 2021e).

Após a implementação de todas as funcionalidades e componentes descritos acima, a interface inicial do aplicativo é apresentada conforme a Figura 8.

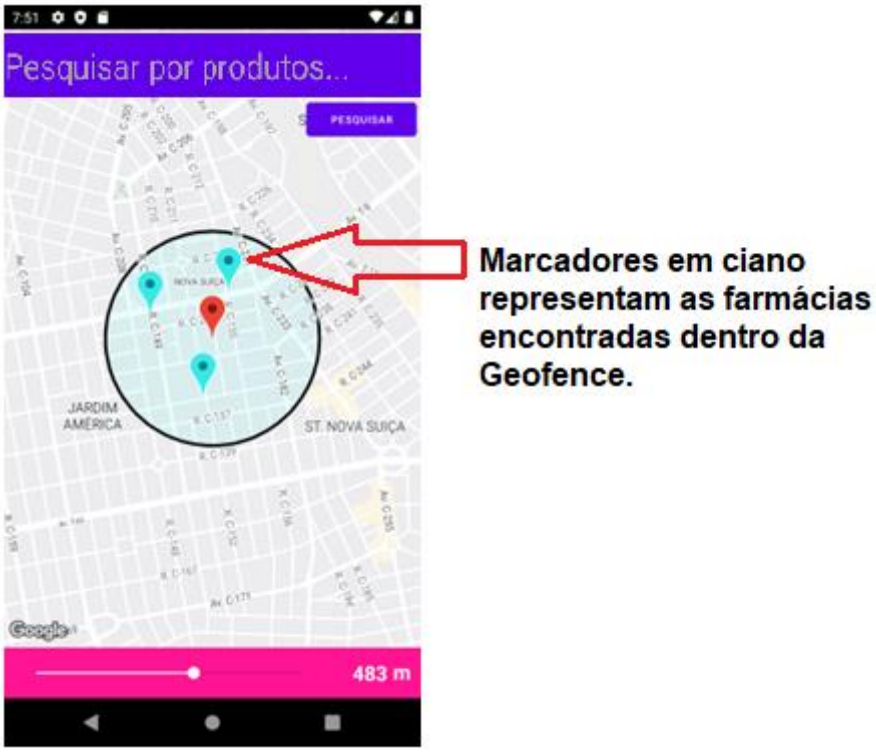
Figura 8 – Tela inicial do aplicativo

Fonte: Elaborado pelo autor, 2022

A função *mapPoints* é chamada em vários métodos. Sua primeira chamada é dentro do método *onCreate*. Essa função utiliza o comando da biblioteca *Location*, chamado *distanceBetween*, que grava em um vetor as informações das distâncias entre as farmácias encontradas e o usuário, para uso posterior. Além

disso, essa função irá realizar uma verificação na área do mapa dentro da *geofence* definida pelo usuário e detectará as lojas que estejam inseridas no banco de dados, e apresentá-las ao usuário na forma de marcadores na cor ciano caso encontre uma ou mais lojas, conforme exibido na Figura 9.

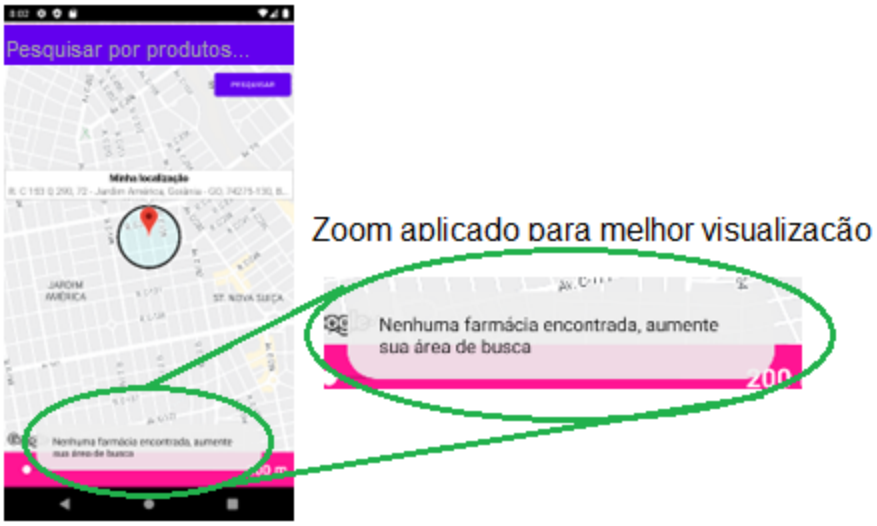
Figura 9 – Aplicativo encontrando as farmácias inseridas no banco de dados e apresentando-as no mapa.



Fonte: Elaborado pelo autor, 2022.

Caso não seja encontrada nenhuma farmácia dentro da *geofence* determinada, uma mensagem aparece para o usuário, pedindo para aumentar a área de busca, como exemplificado na figura 10.

Figura 10 – Interface do aplicativo quando não apresenta farmácias dentro da *geofence.*



Fonte: Elaborado pelo autor, 2022.

Nessa interface também foi inserido um campo de pesquisa juntamente com um botão, que ao ser pressionado, será chamada uma nova atividade nomeada *ProductsActivity,* onde exibirá o resultado da pesquisa feita, mostrando uma lista mostrando todos os produtos encontrados e suas respectivas farmácias.

Caso não seja encontrado nenhum produto nas farmácias pesquisadas, uma interface informando ao usuário será exibida, com a mensagem “Nenhum produto encontrado”, como apresentado na figura 11.

Figura 11 – Interface após uma busca sem resultados.

Fonte: Elaborado pelo autor, 2022.

Caso o usuário decida clicar em um marcador ciano, que representa as farmácias localizadas, ele verá um novo botão na interface principal. Ao pressionar o botão a atividade *ProductsActivity* será chamada, passando as informações necessárias para mostrar todos os produtos disponíveis na farmácia selecionada. Como apresentado nas figuras 12 e 13 a seguir.

Figura 12 – Interface ao clicar em um marcador.

Fonte: Elaborado pelo autor, 2022.

Figura 13 – Interface que lista os produtos na farmácia selecionada.

Fonte: Elaborado pelo autor, 2022.

3.4.2 ProducstActivity

ProducstActivity é a segunda interface criada para este aplicativo, e ela possui dois arquivos, sendo eles o *ProducstActivity.* e *activity_products.xml*. Esta *Activity* contém os métodos e variáveis, usadas para guardar e exibir os produtos buscados pelo usuário na forma de uma lista.

O arquivo *activity_products.xml* contém os elementos de interface desta *Activity*. Essa interface possui apenas um *RecyclerView*. O *RecyclerView* cria uma lista de itens a serem exibidos ao usuário, neste caso os itens serão os produtos buscados, e como o próprio nome sugere, ele recicla o elemento exibido, então quando o elemento é rolado para fora da visualização, sua visualização não é destruída, de forma a agilizar o sistema de janelas, garantindo melhor desempenho. O *RecyclerView* irá reutilizar esta visualização para exibir novos elementos que aparecem na tela conforme o usuário rola a lista (DEVELOPERS, 2021g).

O arquivo *ProducstActivity* contém a implementação do *RecyclerView.* A implementação está contida no método *onCreate.* Assim que esta *Activity* é criada, este método é chamado. Ele recebe os dados enviados pela *Maps Activity* e inseri-los em suas devidas variáveis. Após isso, ele irá fazer uma busca no banco de dados, guardando também em suas devidas variáveis, buscando apenas pelos produtos buscados no *MapsActivity.* Feito isso, a classe *MyAdapter* será instanciada passando as variáveis mencionadas anteriormente como parâmetro e fará a configuração do *RecyclerView.*

A classe *MyAdapter* está implementada no arquivo de mesmo nome, esse arquivo funciona em conjunto com o arquivo *product_item.xml*. No arquivo *MyAdapter* está contida a implementação da classe em questão, onde são implementados os métodos que irão inicializar cada item a ser exibido na lista do *RecyclerView*. O uso desta classe se faz necessário pois o *RecyclerView* necessita que seja definido um adaptador para associar os dados de cada item às visualizações que serão criadas pelo *RecyclerView* (DEVELOPERS, 2021g)

O arquivo *product_item.xml* contém o *layout* do formato no qual cada item será exibido na lista do *RecyclerView.* Para cada item da lista, será exibida a característica do produto, seu preço, o nome da farmácia à qual aquele

determinado produto pertence e a distância que a loja está do centro da geofence. O *RecyclerView* então, irá replicar este layout para cada produto que for sendo exibido, criando assim, uma lista.

Após a implementação das funcionalidades descritas, a interface do aplicativo contendo a lista de produtos será apresentada conforme a figura 14.

Figura 14 – Interface do aplicativo contendo a lista de produtos pesquisados.

Fonte: Elaborado pelo autor, 2022.

Todos os códigos desenvolvidos estão apresentados nos Apêndices de A até K no final do presente trabalho.

4 CONCLUSÃO E PERSPECTIVAS FUTURAS

O trabalho desenvolvido teve por finalidade desenvolver um aplicativo de busca por produtos farmacêuticos e farmácias dentro de uma determinada área virtual denominada *geofence.*

O uso do Google *Firebase* possibilitou a criação de um banco de dados usado para simular as farmácias e produtos nas mesmas, garantindo recuperar esses dados em tempo real. O *Firebase* oferece uma plataforma de fácil uso e uma documentação de entendimento rápido, o que leva ao programador agilidade e praticidade no momento do desenvolvimento. A configuração do *Realtime Database,* tanto para as regras, quanto para a criação do banco de dados em *JSON* foi feita de forma fácil por meio do console do próprio *Firebase.*

O uso da Interface de desenvolvimento *Android Studio* foi o que possibilitou o desenvolvimento desse aplicativo de forma mais ágil, sua iteração com a linguagem *Kotlin* se mostrou eficiente e as documentações bem explicadas, o que traz segurança na hora de criar os códigos. A linguagem *Kotlin* foi uma opção que se mostrou interessante na criação de aplicativos, trazendo uma forma diferente na criação dos métodos em relação ao clássico *Java.* O uso das APIs necessárias durante o desenvolvimento da aplicação foi simples de se adquirir e eficientes em seus resultados, tudo sendo oferecido pelo próprio Google. Utilizar meios tão conhecidos oferece uma comunidade rica em soluções para as diversas dúvidas que foram encontradas durante o desenvolvimento.

Desenvolver esse aplicativo trouxe à tona muitos conhecimentos desenvolvidos durante toda a graduação, além de proporcionar um aumento de conhecimento em várias áreas voltadas para o desenvolvimento de aplicativos. Além de todo conhecimento prático em programação, foi adquirido muito conhecimento teórico no entendimento das funções e lógicas de programação. Além disso, este projeto apresenta um grande potencial para ser continuado possibilitando a adição de novas funcionalidades.

Levando em consideração que no mundo moderno todos possuem aparelhos e dispositivos que permitem uma conexão com internet, muitas maneiras de deixar a vida mais fácil surgiram, sendo assim, o intuito desse aplicativo de levar facilidade e praticidade ao procurar por produtos e localizar as farmácias foi cumprido, com a utilização de um banco de dados que faz a sincronização em tempo real traz

velocidade e praticidade no momento de pesquisar por qualquer produto.

Utilizar *geofences* na construção desse aplicativo abriu várias portas e mostrou várias maneiras em que a tecnologia de *geofencing* pode atuar no mundo moderno, o sistema desenvolvido traz uma dessas várias aplicações possíveis. Utilizar um sistema em que uma área virtual é criada em torno do usuário ou de uma área que ele deseja, traz uma sensação de segurança, para que sempre os resultados mostrados sejam o mais próximo possível da área escolhida. Sendo assim, era esperado que o uso desses perímetros trouxesse uma melhor visualização do entorno de cada usuário.

As propriedades de programação do *Android Studio* permitem criar as mais diversas funcionalidades em aplicativos, uma delas, é a facilidade em criar listas de forma clara e simples de visualizar, para que quem utiliza o aplicativo, possa fazer suas pesquisas e receber rapidamente os resultados esperados, listando informações necessárias para que o usuário possa fazer suas escolhas de forma mais sábia e calcular melhor suas rotas e gastos.

Com isso é possível entender que o aplicativo desenvolvido durante a criação desse trabalho de conclusão de curso está em sintonia com os objetivos propostos. Além de ser uma porta de entrada para vários tipos de *software*s parecidos, para vários tipos de estabelecimentos comerciais diferentes.

Como projetos futuros sugere-se implementar um sistema de busca que permita buscar os produtos não apenas por seu nome, mas também por seus componentes, implementar uma tela que se abra ao clicar em algum produto, mostrando imagens do mesmo e suas características como: componentes, via de uso, bula, entre outras informações.

Outra ideia é criar uma busca por lojas e produtos utilizando-se de mais de uma *geofence* ao mesmo tempo ou implementar um sistema de autenticação para que as próprias farmácias cadastrem suas informações e seus produtos.

5 REFERÊNCIAS BIBLIOGRAFICAS

AGOSTINHO, Henrique Leitão. **Marketing digital na indústria farmacêutica**. Dissertação de Mestrado Integrado em Ciências Farmacêuticas. Instituto SuperiorDe Ciências Da Saúde Egas Moniz. Almada, 2015.

ALMEIDA, Igor P. de; ASSUNÇÃO, Letícia R.; SIMÕES, Thállys L.; LIMA, Joselice F.**Visão Sobre Dispositivos e Sistemas Operacionais Móveis**. Instituto Federal do Norte de Minas Gerais (IFNMG). Janúária, 2014. Disponível em <http://anais.simposioinformatica.ifnmg.edu.br/ojs/index.php/anaisviiisimposio/article / view/45>. Acesso em: 23 set. 2021.

BANERJEE, Madhurima et al. **A COMPARATIVE STUDY: JAVA VS KOTLIN PROGRAMMING IN ANDROID APPLICATION DEVELOPMENT**. International Journal of Advanced Research in Computer Science, [S.l.], v. 9, n. 3, p. 41-45, Junho2018. Disponível em: <http://ijarcs.info/index.php/Ijarcs/article/view/5978/4908>. Acesso em: 28 set. 2021.

BARETH, Ulrich; KÜPPER, Axel; FREESE, Behrend. **Geofencing and BackgroundTracking - The Next Features in LBS**. 2011.- Informatik schafft Communities. Berlin.

CAELUM. **O que é JAVA?**. 2016. Disponível em: https://www.caelum.com.br/apostila-javaorientacao-objetos/o-que-e-java. Acesso em:06 out. 2021.

CARVALHO, Edilson Alves de.; ARAÚJO, Paulo César de. **Leituras cartográficas einterpretações estatísticas I: geografia**. 2008. Natal, RN. EDUFRN, 2008.

CHURCHILL, Gilbert A. **Marketing: criando valor para os clientes**. 3 ed. SãoPaulo. Saraiva, 2017.

CLARO, Daniela Barreiro; SOBRAL, João Bosco Mangueira. **Programação emJAVA**. Florianópolis. Copyleft Pearson Education, 2008.

DEVELOPERS. **Fragmentos**. 2019. Disponível em: <https://developer.android.com/guide/components/fragments?hl=pt-br>. Acesso em: 12 mar. 2022.

DEVELOPERS. **Conheça o Android Studio,** 2021a. Disponível em: <https://developer.android.com/studio/intro?hl=pt-br>. Acesso em: 12 mar. 2022.

DEVELOPERS. **Guia de início rápido do SDK do Maps para Android**, 2021b. <Disponível em: https://developers.google.com/maps/documentation/android-

sdk/start>. Acesso em: 12 mar. 2022.

DEVELOPERS. **Solicitar permissões de localização**, 2021c. Disponível em: <https://developer.android.com/training/location/permissions?hl=pt-br>. Acesso em: 15 mar. 2022.

DEVELOPERS. **Dados de local**, 2021d. Disponível em: <https://developers.google.com/maps/documentation/android-sdk/location?hl=pt-br>. Acesso em: 15 mar. 2022.

DEVELOPERS. **Câmera e visualização**, 2021e. Disponível em: <https://developers.google.com/maps/documentation/android-sdk/views?hl=pt-br>. Acesso em: 19 mar. 2022.

DEVELOPERS. **LocationRequest**, 2021f. Disponível em: <https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest>. Acesso em: 09 abr. 2022.

DEVELOPERS. **Criar listas dinâmicas com o RecyclerView**, 2021g. Disponível em: <https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=pt-br>. Acesso em: 12 abr. 2022.

DEVELOPERS. **Android Studio: An IDE built for Android**, 2013. Disponível em: < https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html>. Acesso em: 27 mar. 2022.

DIMARZIO, Jerome. **Beginning Android Programming with Android Studio**. John Wiley & Sons, 2016.

DOMPIERI, Márcia Helena Galina; SILVA, Marcos Aurélio Santos da; JÚNIOR, Lauro Rodrigues Nogueira. **Sistemas de referência terrestre e posicionamentopor satélite**. 2015. 35 f. Embrapa, Embrapa Tabuleiros Costeiros, Aracaju, 2015. Disponível em: <https://www.infoteca.cnptia.embrapa.br/infoteca/bitstream/doc/1042182/1/Doc197.p df>. Acesso em: 01 set. 2021.

ELMASRI, Ramez; NAVATHE, Shamkant B**. Sistemas de banco de dados**. SãoPaulo: Pearson Addison Wesley, 2005.

FAGGIAN, Hugo Cesar. **Geometria e GPS**. 2019. Dissertação (Mestrado em Mestrado Profissional em Matemática em Rede Nacional) - Instituto de CiênciasMatemáticas e de Computação, Universidade de São Paulo, São Carlos, 2019. doi:10.11606/D.55.2019.tde-17092019-150542. Acesso em: 07 set. 2021.

FAUSTINO, Gleicy Kellen dos Santos; CALAZANS, Hallana Keury Nunes de Sousa;LIMA; Welton Dias de. **Android e a influência do Sistema Operacional Linux**.
2017. Disponível em: <http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/viewFile/829/7

28
> Acesso em: 17 set. 2021.

FIREBASE. **Adicionar o Firebase ao projeto para Android**, 2021. Disponível em: <https://firebase.google.com/docs/android/setup?hl=pt-br>. Acesso em: 17 fev. 2022.

FIREBASE. **Firebase Realtime Database**, 2020b. Disponível em: <https://firebase.google.com/docs/database?authuser=0>. Acesso em: 17 fev. 2022.

FIREBASE. **Ler e gravar dados no Andriod**, 2020c. Disponível em: <https://firebase.google.com/docs/database?authuser=0>. Acesso em: 22 maio. 2022.

GUDWIN, Ricardo R. **Linguagens de Programação**. Universidade Estadual deCampinas, 1997.

HELLMANN, Rafael Alexandre Freiberger. **Aplicativo Android e website interativos para busca de menores preços de produtos com código de barras.** 2016. 90 f. TCC (Graduação)- Curso de Engenharia Elétrica, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

KOTLIN. **Comece com Kotlin**. 2021. Disponível em: <https://kotlinlang.org/docs/gettingstarted.html> Acesso em: 02 fev 2022.

LUCENA, Tiago Franklin Rodrigues; MOTA, Gesiel Leachi Arruda. **Publicidade baseada em geolocalização: possibilidades criativas dos usos dos sistemas locativos**. 2015. Iniciação Científica Cesumar. Temática: Comunicação, Jornalismo,Publicidade e Propaganda. v. 17 n. 2. Maringá, 2015. Disponível em:<https://periodicos.unicesumar.edu.br/index.php/iccesumar/article/view/4239/2665>. Acesso em 09 set 2021.

LUIZ, Gilberto Venâncio. **Consumo de telefone celular: significados e influênciasna vida cotidiana dos adolescentes**. Dissertação de Mestrado em Economia familiar; Estudo da família; Teoria econômica e Educação do consumidor - Universidade Federal de Viçosa, Viçosa, 2008.

MACHADO, Everto Fábio da Silva. **Desenvolvimento de sistemas de geolocalização e rastreamento para a plataforma Android – COMPASS**. 2015. 55 f. Monografia (Especialização) - Curso de Desenvolvimento de Sistemas para a Internet e Dispositivos Móveis, Coordenação de Licenciatura em Informática, Universidade Tecnológica Federal do Paraná, Francisco Beltrão, 2015. Disponível em:<http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6914/1/FB_DESIDM_I_2014_01.pdf >. Acesso em: 08 set. 2021.

MARILIA**. Banco de dados NoSQL: um novo paradigma** – revista SQL

magazine102. 2012. Disponível em: <https://www.devmedia.com.br/banco-de-dados-nosql-um-novoparadigma-revista-sql-magazine-102/25918>. Acesso em: 22 set. 2021.

MORONEY, Laurence. An Introduction to Firebase. In: MORONEY, Laurence. **The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform**. Seattle, EUA: Apress, 2017. Cap. 1. p. 1-24.

OKADA, Sionara Ioco; SOUZA, Eliane Moreira Sá de. **Estratégias de marketing digital na era da busca.** Revista Brasileira de Marketing, vol. 10, núm. 1, abril, 2011,p. 46-72. Universidade Nove de Julho. São Paulo. Disponível em:<http://www.redalyc.org/articulo.oa?id=471747524003>. Acesso em: 27 set. 2021.

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

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: 25 set. 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: 10 out 2021.

SILVA, Débora. **Banco de dados**. 2015. Disponível em: <https://www.estudopratico.com.br/banco-de-dados/>. Acesso em: 08 out. 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-deprogramacao-java-e-numero-1-no-mundo/>. Acesso em: 06 out. 2021.

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

SUDIARTHA, I. K. G; INDRAYANA, N. E. I; SUASNAWA, I. W. **Data Structure Comparison Between MySql Relational Database and Firebase Database NoSql on Mobile Based Tourist Tracking Application.** Journal of Physics: Conference Series. Out. de 2019. Disponível em: <https://iopscience.iop.org/article/10.1088/1742-6596/1569/3/032092/pdf>. Acesso em: 14 fev. 2022.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. 2.ed. Rio de Janeiro: Elsevier, 2014.

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: 01set. 2021.

ZANOTTA, Daniel Capella; CAPPELLETTO, Eliane; MATSUOKA, Marcelo Tomio. **OGPS: unindo ciência e tecnologia em aulas de física**. 2011 Revista Brasileira de Ensino de Física, v. 33, n. 2, 2313. Jul. 2011. Disponível em: <https://www.scielo.br/j/rbef/a/dWNb3PyKWnCsg9wXkqqC9vf/?format=pdf&lang=pt>
. Acesso em: 03 set. 2021.

PUC GOIÁS

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

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Felipe Pereira Brandão
do Curso de Engenharia de Computação, matrícula 2016100330539-0,
telefone: (62) 99636-9831 e-mail felipe.brandao@gmail.com,
na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado O uso de geofencing na busca de produtos e farmácias, 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, 07 de junho de 2022.

Assinatura do autor: Felipe Pereira Brandão

Nome completo do autor: Felipe Pereira Brandão

Assinatura do professor–orientador: [signature]

Nome completo do professor-orientador: Marcelo Antonio Costa de Araújo

APÊNDICE A - Código MapsActivity.kt

```
package com.felipebrand.findpharma.mapas

import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Color
import android.location.Geocoder
import android.location.Location.distanceBetween
import android.os.Bundle
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import com.felipebrand.findpharma.ProductsActivity
import com.felipebrand.findpharma.R
import com.google.android.gms.location.*
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.*
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ValueEventListener
import com.google.firebase.database.ktx.database
import com.google.firebase.ktx.Firebase
import java.io.IOException
import java.io.Serializable
import java.util.*


class MapsActivity : AppCompatActivity(),OnMapReadyCallback {

    // VARIAVEIS GLOBAIS UTILIZADAS NESSA ACTIVITY
    private lateinit var mMap: GoogleMap
    lateinit var databaseRef: DatabaseReference
    var marker: Marker? = null
    private lateinit var circle: Circle
    var fusedLocationProviderClient: FusedLocationProviderClient? = null
    private lateinit var home: LatLng
    private lateinit var buttonSearch: Button
    private lateinit var buttonProds : Button
    private var farmas = mutableListOf<Farmacia>()
    private lateinit var seekBar: SeekBar
    private lateinit var campo_pesquisa: EditText
```

```
    private lateinit var textSize: TextView
    private lateinit var progressView: TextView
    private var markerList = mutableListOf<Marker>()
    private var distance = mutableListOf<Distance>()
    private var distanceAux = mutableListOf<Distance>()
    var distanceList = arrayListOf<Int>()
    var geofenceSize = 200.00
    var zoom = 15.0f
    lateinit var farmaAux: Farmacia
    lateinit var markerLocation: LatLng
    private lateinit var locationCallback: LocationCallback
    private val REQUEST_CODE = 101

    /*FUNÇÃO onCreate É INICIALIZADA AO ABRIR O APLICATIVO, INICIANDO
ALGUMAS VARIÁVEIS E RECEBENDO
    ALGUAMAS INFORMAÇÕES DO BANCO DE DADOS */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)

        //VARIÁVEL CLIENTE RECEBE OS SERVIÇOS DE LOCALIZAÇÃO
        fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this)

        //INICIALIZA O FRAGMENTO DO MAPA
        val supportMapFragment =
            (supportFragmentManager.findFragmentById(R.id.map_fragment) as
SupportMapFragment?)!!
        supportMapFragment.getMapAsync(this)

        //INICIALIZA OS ELEMENTOS DA INTERFACE E ALGUMAS VARIÁVEIS
        seekBar = findViewById(R.id.seekBar)
        textSize = findViewById(R.id.textSize)
        progressView = this.textSize
        buttonSearch = findViewById(R.id.button_search)
        buttonProds = findViewById(R.id.button)
        campo_pesquisa = findViewById(R.id.campo_pesquisa)
        home = LatLng(000.0, 000.0)
        markerLocation = LatLng(000.0, 000.0)

        //  INICIA-SE O BANCO DE DADOS A BUSCAR A PARTIR DO NÓ "PHARMA"
        databaseRef = Firebase.database.reference.child("pharma")

        /* CRIA O EVENTO DE MUDANÇA, SEMPRE QUE O BANCO SOFRE UMA
ALTERAÇÃO, CIRANDO SNAPSHOTS DO
        MOMENTO DO BANCO */
        databaseRef.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {

                for (objSnapshot in snapshot.children) {
```

```
                farmaAux = Farmacia()

                farmaAux.setNome(objSnapshot.child("nome").value as String)
                farmaAux.setLat(objSnapshot.child("latitude").value as Double)
                farmaAux.setLong(objSnapshot.child("longitude").value as Double)
                farmaAux.setEnd(objSnapshot.child("endereco").value as String)
                farmaAux.setId(objSnapshot.value.toString())

                farmas.add(farmaAux)

            }

            //CHAMA PELA PRIMEIRA FEZ A FUNÇÃO PARA MARCAR OS
PONTOS NO MAPA
            mapPoints()
        }
        // FUNÇÃO É CHAMADA EM CASO DE ERRO AO LER O BANCO DE
DADOS
        override fun onCancelled(error: DatabaseError) {
            Toast.makeText(applicationContext, "Erro ao ler os dados",
Toast.LENGTH_SHORT)
                .show()
        }
    })
}

/* FUNÇÃO onResume É CHAMADA SEMPRE QUE O APLICATIVO VOLTAR DE
OUTRA ATIVITADE OU DE FORA DO
* APLICATIVO*/
override fun onResume() {
    super.onResume()

    // VERIFICA SE AS PERMISSÕES FORAM ACEITAS
    if (ActivityCompat.checkSelfPermission(
            this,
            Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(
            this,
            Manifest.permission.ACCESS_COARSE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED
    ) {
        ActivityCompat.requestPermissions(
            this,
            arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
            REQUEST_CODE
        )
        return
    }
```

```
        val task = fusedLocationProviderClient!!.lastLocation // CHAMA A FUNÇÃO
LAST LOCATION
        //PARA GUARDAR A LOCALIZAÇÃO ATUAL DO DISPOSITIVO

        task.addOnSuccessListener { location ->           // EM CASO DE SUCESSO

            //INICIA-SE AS VARIAVEIS DE LOCALIZAÇÃO USANDO O LOCAL AUTAL
            val latLng = LatLng(location.latitude, location.longitude)
            home = latLng
            markerLocation = latLng

            //FAZ A MARCAÇÃO NO MAPA DO LOCAL ATUAL, CRIANDO TAMBÉM A
GEOFENCE
            val markerOptions = MarkerOptions().position(latLng).title("Minha
localização")
                .snippet(getTheAddress(latLng.latitude, latLng.longitude)).draggable(true)
            mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng))
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15f))
            marker = mMap.addMarker(markerOptions)
            marker?.showInfoWindow()
            this.circle = mMap.addCircle(
                CircleOptions().center(latLng).fillColor(Color.argb(20, 0, 255, 255))
                    .strokeWidth(
                        8F
                    ).radius(geofenceSize).visible(true)
            )
        }

        task.addOnFailureListener {
            Log.e("Error location", "Erro ao encontrar localização")
        }

        /*OS CÓDIGOS A SEGUIR MANTEM O APLICATIVO RECEBENDO OS
DADOS DE LOCALIZAÇÃO DO DISPOSITIVO
        DE 15 SEGUNDOS COM UMA PRIORIDADE BALENCEADA, PARA
ECONOMIZAR BATERIA*/
        val locationRequest: LocationRequest = LocationRequest.create()
        locationRequest.interval = 15 * 1000  //Intervalo de busca de atualização em
milisegundos
        locationRequest.priority =
            LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY
        //Para evitar atualizações desnecessárias ele somente atualiza após uma certa
distancia
        locationRequest.smallestDisplacement = 170f
        locationCallback = object : LocationCallback() {
            override fun onLocationResult(locationResult: LocationResult) {
                locationResult
                if (locationResult.locations.isNotEmpty()) {
                    val location =
                        locationResult.lastLocation
```

```
            }
        }
    }
    //REALIZA O UPDATE DA LOCALIZAÇÃO
    fusedLocationProviderClient?.requestLocationUpdates(
        locationRequest,
        locationCallback,
        Looper.myLooper()!!
    )
}

  //METODO onMapReady É CHAMADO QUANDO O FRAGMENTO DO MAPA
ESTÁ PRONTO PARA USO
  override fun onMapReady(googleMap: GoogleMap) {

    //VARIÁVEL RECEBE O CONTROLE DO MAPA PELA BIBLIOTECA
    mMap = googleMap

    //DEFINE AS PREFERÊNCIAS DE ZOOM (5f PARA CONTINENTES, 20f
PARA EDIFÍCIOS)
    mMap.setMinZoomPreference(5.0f)
    mMap.setMaxZoomPreference(20.0f)

    //DEFINE OS CONTROLES DO MAPA, COMO O CONTROLE DE ZOOM
    mMap.isMyLocationEnabled
    mMap.uiSettings.isMyLocationButtonEnabled
    mMap.uiSettings.isZoomControlsEnabled

    /*CLIQUE DO BOTÃO CHAMA UMA NOVA ATIVIDADE, ONDE SERÁ FEITA A
COMPARAÇÃO DO QUE FOI
    PESQUISADO COM O QUE ESTA ARMAZENADO */

    buttonSearch.setOnClickListener { view ->

        val text = campo_pesquisa.getText().toString()

        //CRIA UMA INTENT(Intenção) PARA A NOVA ATIVIDADE A SER
CHAMADA
        val intent = Intent(applicationContext, ProductsActivity::class.java)
        //ENVIA OS DADOS PARA A NOVA ATIVIDADE
        intent.putExtra("buscador", distance as Serializable)
        intent.putExtra("id", text)
        //INICIA A ATIVIDADE
        startActivity(intent)
    }

    /* DEFINE O CONTROLE DESLIZANTE E SUAS FUNÇÕES PARA
MANIPULÇÃO DO RAIO DA GEOFENCE*/
    seekBar.setOnSeekBarChangeListener(object :
```

```
SeekBar.OnSeekBarChangeListener {
        //INICIA VARIAVEIS LOCAIS
        var minRadius: Int = 0
        var zoomOut: Float = 0.0f

        //FUNÇÃO CHAMADA ENQUANTO A BARRA ESTÁ EM PROGRESSO, OU
DESLIZANDO
        override fun onProgressChanged(seekBar: SeekBar, i: Int, b: Boolean) {
            minRadius = i + 200 //Define o raio da Geofence
            val aux = i / 15000 //Define o tamaho do zoom que será aplicado no mapa
            zoomOut = zoom - aux.toFloat() //Cria o zoomOut do mapa caso o zoom
esteja proximo
            circle.radius = minRadius.toDouble()//define o raio da geofence
            if (zoomOut > 12)//anima a camera caso o zoom esteja perto demais
                mMap.animateCamera(CameraUpdateFactory.zoomTo(zoomOut))
            geofenceSize = minRadius.toDouble() //armazena o tamanho do raio
            progressView.text = minRadius.toString() /* altera a visualização para
mostar o
            tamanho da geofence */


        }
        override fun onStartTrackingTouch(seekBar: SeekBar) {
        }

        override fun onStopTrackingTouch(seekBar: SeekBar) {
            /*CHAMA NOVAMENTE A FUNÇÃO PARA CRIAR OS PONTOS NO
MAPA APÓS PARAR DE MOVIMENTAR O
             CONTROLE DESLIZANTE */
            mapPoints()

        }
    })

    //IMPLEMENTA O MARCADOR PERSONALIZADO
    mMap.setOnMapLongClickListener { latLng ->
        geofenceMarker(latLng)
        mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng))
        markerLocation = latLng
        mapPoints()

    }
    //EM CASO DE CLICK EM UM MARCADOR DE FARMÁCIA
    mMap.setOnMarkerClickListener { clicked ->
        //define a visibilade do botão para ver os produtos
        if(clicked.title != marker?.title) {
            buttonProds.visibility = View.VISIBLE
        }else{
            buttonProds.visibility = View.INVISIBLE
        }
```

```
            //cria uma lista passando a distância
            for (i in farmas.indices){
                if(clicked.title.equals(farmas[i].getNome())) {
                    for(j in 0..5){
                        distanceAux.add(j, distance[i])
                    }
                }
            }
            val auxClicked = clicked.title
            buttonProds.setOnClickListener { view ->

                //CRIA UMA INTENT(Intenção) PARA A NOVA ATIVIDADE A SER
CHAMADA
                val intent = Intent(applicationContext, ProductsActivity::class.java)
                //ENVIA OS DADOS PARA A NOVA ATIVIDADE
                intent.putExtra("marker", auxClicked)
                intent.putExtra("buscador", distanceAux as Serializable)
                //INICIA A ATIVIDADE
                startActivity(intent)
            }
            false

        }

        //IMPLEMENTA O BOTÃO PARA RETONRAR À POSIÇÃO ATUAL DO
DISPOSITIVO
        mMap.setOnMyLocationButtonClickListener {
            geofenceMarker(home)
            markerLocation = home
            return@setOnMyLocationButtonClickListener false

        }

    }

    /* FUNÇÃO QUE CRIA AS GEOFENCES NOS MARCADORES
PERSNALIZADOS E NA LOCALIZAÇÃO ATUAL*/
    fun geofenceMarker(latLng: LatLng) {       //recebe uma localização como
parametro
        circle.remove()     //remove geofences ja criadas
        marker?.remove()    //remove os marcadores para colocar os novos
        if (latLng == home) {
            marker =
                mMap.addMarker(MarkerOptions().position(latLng).title("Minha
Localização"))!!
        } else
            marker = mMap.addMarker(
                MarkerOptions().position(latLng).title("Marcador Personalizado")
            )!!
        circle = mMap.addCircle(
```

```
            CircleOptions().center(latLng)
                .fillColor(Color.argb(20, 0, 255, 255))
                .strokeWidth(8F).radius(geofenceSize).visible(true)
            // adicona as geofences recebendo o raio vindos do controle deslizante
        )
    }

    /* ESSA FUNÇÃO UTILIZA A BIBLIOTECA Geocoder PARA RECEBER O
ENDEREÇO DO MARCADOR*/
    private fun getTheAddress(latitude: Double, longitude: Double): String? {
        var retVal = ""
        val geocoder = Geocoder(this, Locale.getDefault())
        try {
            val addresses = geocoder.getFromLocation(latitude, longitude, 1)
            retVal = addresses[0].getAddressLine(0)

        } catch (e: IOException) {
            e.printStackTrace()
        }
        return retVal
    }
    /*FUNÇÃO CRIADA PARA MARCAR AS FARMÁCIAS ENCONTRADAS NO
MAPA*/
    fun mapPoints(){
        val results = FloatArray(3) //vetor utilizado no comando distanceBetween
        lateinit var auxDistance: Distance



        if (markerList.isNotEmpty()) {
            for (i in markerList.indices)
                markerList[i].remove()

            markerList.clear() // remove todos os marcadores caso já tenha no mapa

        }
        if (distance.isNotEmpty())
            distance.clear()        //limpa a lista com os dados antigos para escrever os
novos

        for (i in farmas.indices) {

            distanceBetween(
                markerLocation.latitude,
                markerLocation.longitude,
                farmas[i].getLat(),
                farmas[i].getLong(),
                results
            )
            distanceList.add(i,results[0].toInt())      /*Função faz o calculo da distnacia
```

```
entre
        o marcador das farmácias e o marcador do usuário salvando em um vetor em
metros*/


        if (results[0].toDouble() <= geofenceSize) {
            //ADICIONA OS MARCADORES NO MAPA PERSONALIZANDO-OS
            markerList.add(
                mMap.addMarker(
                    MarkerOptions().position(
                        LatLng(
                            farmas[i].getLat(),
                            farmas[i].getLong()
                        )
                    )
                        .title(farmas[i].getNome().toString())
                        .icon(BitmapDescriptorFactory
                            .defaultMarker(BitmapDescriptorFactory.HUE_CYAN))
                        .snippet(farmas[i].getEnd())

                )!!
            )
            //GUARDA NA CLASSE ALGUMAS INFORMAÇÕES
            auxDistance = Distance()
            auxDistance.setDistancia_loja(distanceList[i])
            auxDistance.setNome_loja(farmas[i].getNome().toString())
            distance.add(auxDistance)
        }

    }

}

//FUNÇÃO QUE RECONHECE SE AS PAERMISSÕES
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String?>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    when (requestCode) {
        REQUEST_CODE -> if (grantResults.size > 0 && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
            return
        }

    }
}

}
```

APÊNDICE B - Código ProductsActivity.kt

```
package com.felipebrand.findpharma

import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.DividerItemDecoration
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.felipebrand.findpharma.mapas.Distance
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ValueEventListener
import com.google.firebase.database.ktx.database
import com.google.firebase.ktx.Firebase

class ProductsActivity : AppCompatActivity() {
    //INSTACIA DAS VARIÁVEIS GLOBAIS E DE INTERFACE
    private lateinit var userRecyclerView: RecyclerView
    private lateinit var myAdapter: MyAdapter
    private lateinit var text_find: TextView

    lateinit var mDatabase: DatabaseReference

    private var busca = arrayListOf<Busca>()
    var aux = Busca()

    //MÉTODO CHAMADO NO MOMENTO DA CRIAÇÃO DA ATIVIDADE
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        //INICIA OS COMPONENTES DA INTERFACE E O RecyclerView
        setContentView(R.layout.activity_products)
        userRecyclerView = findViewById(R.id.lista_prods)
        text_find = findViewById(R.id.textFind)

        //  INICIA O BANCO DE DADOS
        mDatabase = Firebase.database.reference.child("pharma")

        //VARIÁVEL DADOS RECEBE O QUE FOI ENVIADO DA ACTIVITY
MapsActivity
        val dados: Bundle? = intent.extras
        val search = dados?.getString("id")
        val markerClick = dados?.getString("marker")
```

```
        val distance = dados?.getSerializable("buscador") as ArrayList<Distance>

        //DEFINE COMO SERÁ O TIPO DE IMPRESSÃO DA LISTA NO RecyclerView
        userRecyclerView.layoutManager = LinearLayoutManager(this)
        userRecyclerView.setHasFixedSize(true)
        userRecyclerView.addItemDecoration(DividerItemDecoration(this,
            LinearLayout.VERTICAL))
        myAdapter = MyAdapter(busca, distance)
        userRecyclerView.adapter = myAdapter

        /* CRIA O EVENTO DE MUDANÇA, SEMPRE QUE O BANCO SOFRE UMA
ALTERAÇÃO, CIRANDO SNAPSHOTS DO
            MOMENTO DO BANCO */
        mDatabase.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if(busca.isNotEmpty()){
                    busca.clear()   //limpa a busca
                }
                for (objSnapshot in snapshot.children) {
                    for (postSnapshot in objSnapshot.child("produtos").children) {
                        aux = Busca()
                        aux.setNome(objSnapshot.child("nome").value as String)
                        aux.setPopular(postSnapshot.child("nome_popular").value as String)
                        aux.setCarac(postSnapshot.child("carac").value as String)
                        aux.setValor(postSnapshot.child("valor").value as Double)
                        //  if (markerClick != null) {
                        if(markerClick == aux.getNome())
                            busca.add(aux)

                        // } else {
                        if (search == aux.getPopular()) {     /*FAZ A BUSCA DO PRODUTO E
INSERE NA
                            LISTA APENAS OS PRODUTOS PESQUISADOS*/
                            busca.add(aux)
                        }
                        // }
                    }
                }
                if(busca.isEmpty()){
                    text_find.visibility = View.VISIBLE // CASO NENHUM PRODUTO FOR
ENCONTRADO
                }else{
                    myAdapter = MyAdapter(busca, distance) // ENVIA PARA O
ADAPTADOR A LISTA
                }

            }
            // FUNÇÃO É CHAMADA EM CASO DE ERRO AO LER O BANCO DE
DADOS
            override fun onCancelled(error: DatabaseError) {
```

```
                Toast.makeText(
                    applicationContext,
                    "Erro ao ler os dados.",
                    Toast.LENGTH_SHORT
                ).show()
            }

        })
    }
}
```

APÊNDICE C - Código MyAdapter.kt

```
package com.felipebrand.findpharma

import android.annotation.SuppressLint
import android.icu.text.DecimalFormat
import android.os.Build
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.annotation.NonNull
import androidx.annotation.RequiresApi
import androidx.recyclerview.widget.RecyclerView
import com.felipebrand.findpharma.mapas.Distance

/*CLASSE ADAPTADOR NECESSÁRIO PARA CRIAR LISTAS VERTICAIS*/


open class MyAdapter(
    private var produtos: ArrayList<Busca>,
    private var distance: ArrayList<Distance>
) : RecyclerView.Adapter<MyAdapter.MyViewHolder>() {

    @RequiresApi(Build.VERSION_CODES.N)
    val dec = DecimalFormat("#,###.00")
    @RequiresApi(Build.VERSION_CODES.N)
    val dec2 = DecimalFormat("###,#")

//CRIA A LISTA VERTICAL UTILZANDO O LAYOUT QUE SERÁ REPLICADO
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
        val itemLista: View =
LayoutInflater.from(parent.context).inflate(R.layout.product_item,
            parent, false)
        return MyViewHolder(itemLista)
    }
    @RequiresApi(Build.VERSION_CODES.N)
    @SuppressLint("SetTextI18n")

    /*RECEBE OS ELEMENTOS DAS LISTAS GUARDANDO EM CADA POSIÇÃO
QUE É INCREMENTADO SEMPRE QUE UM NOVO
     ITEM É INSERIDO */

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {

        val prodAux: Busca = produtos[position]

        holder.nome_loja.text = prodAux.getNome()
```

```
//        if(lista_prod[position].getNome() == lista_prod[position+1].getNome()){
//            holder.distancia.text = "<"+ dec2.format(distance[0].getDistancia_loja())+"
m"
//        }else{
            holder.distancia.text = "<"+
dec2.format(distance[position].getDistancia_loja())+" m"
//        }
        holder.carac_prod.text = prodAux.getCarac()
        holder.preco_prod.text = "R$" + dec2.format(prodAux.getValor()).replace(".",
            ",")
        holder.nome_prod.text = prodAux.getPopular()
    }

    //FUNÇÃO PARA CONTROLE DA QUANTIDADE DE ITENS NA LISTA
    override fun getItemCount(): Int {
        return produtos.size
    }

    //CLASSE QUE INTERLIGA OS ITENS NO LAYOUT QUE SERÁ REPLICADO
    class MyViewHolder(@NonNull itemView: View) :
RecyclerView.ViewHolder(itemView) {
        var nome_loja: TextView = itemView.findViewById(R.id.show_name)
        var carac_prod: TextView =  itemView.findViewById(R.id.show_carac)
        var distancia: TextView = itemView.findViewById(R.id.show_distance)
        var preco_prod: TextView = itemView.findViewById(R.id.show_price)
        var nome_prod: TextView = itemView.findViewById(R.id.show_prod)

    }
}
```

APÊNDICE D - Código MainActivity

```
package com.felipebrand.findpharma

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.felipebrand.findpharma.mapas.MapsActivity

//CLASSE APENAS PARA MOSTRAR UMA TELA DE APRESNETAÇÃO DO NOME DO APLICATIVO
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //DEFINE O TEMA PESONALIZADO
        setTheme(R.style.Theme_FindPharma)
        setContentView(R.layout.activity_main)
        Thread.sleep(1000) //AGUARDA 1 SEGUNDO NA TELA PERSONALIZADA
        tela1()
    }

    private fun tela1() {
        //CHAMA UMA INTENT DO MapsActivity
        val intent = Intent(this, MapsActivity::class.java)
        startActivity(intent)// INICIA O MapsActivity
    }
}
```

APÊNDICE E - Código Farmacia.kt

```
package com.felipebrand.findpharma.mapas

import java.io.Serializable

class Farmacia : Serializable {
    private var nome: String? = null
    private var lat = 0.0
    private var longi = 0.0
    private var end: String? = null
    private var Id: String? = null


    fun getNome(): String? {
        return nome
    }

    fun setNome(nome: String) {
        this.nome = nome
    }

    fun getLat(): Double {
        return lat
    }

    fun setLat(lat: Double) {
        this.lat = lat
    }

    fun getLong(): Double {
        return longi
    }

    fun setLong(longi: Double) {
        this.longi = longi
    }

    fun getEnd(): String? {
        return end
    }

    fun setEnd(end: String) {
        this.end = end
    }

    fun getId(): String? {
        return Id
    }
```

```
    fun setId(id: String) {
        this.Id = id
    }
}
```

APÊNDICE F - Código Busca.kt

```
package com.felipebrand.findpharma

import java.io.Serializable

class Busca:Serializable {
    private var nome: String? = null
    private var nome_popular:String? = null
    private var carac:String? = null
    private var valor = 0.0
    private var distanciaLoja = 0

    fun getNome(): String? {
        return nome
    }

    fun setNome(nome: String) {
        this.nome = nome
    }
    fun getPopular():String?{
        return nome_popular
    }
    fun setPopular(popular:String){
        this.nome_popular = popular
    }


    fun getCarac():String?{
        return carac
    }
    fun setCarac(carac:String){
        this.carac = carac
    }
    fun getValor():Double{
        return valor
    }
    fun setValor(valor:Double){
        this.valor = valor
    }

    fun getDistancia_loja():Int{
        return distanciaLoja
    }

    fun setDistancia_loja(dloja: Int) {
        this.distanciaLoja = dloja
    }
}
```

APÊNDICE G - Código Distance.kt

```
package com.felipebrand.findpharma.mapas

import java.io.Serializable

class Distance: Serializable {

    private var distanciaLoja = 0

    fun getDistancia_loja():Int{
        return distanciaLoja
    }

    fun setDistancia_loja(dloja: Int) {
        this.distanciaLoja = dloja
    }

}
```

APÊNDICE H - Códigos da Interface

Pasta Drawable – background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient android:endColor="@color/deepPink"
            android:startColor="@color/white"
        android:angle="90"></gradient>
</shape>
```

Pasta Drawable – splash_welcome.xml

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

    <item android:drawable="@drawable/background"/>
    <item>
        <bitmap
            android:gravity="fill"
            android:src="@drawable/imagem_inicial"/>

    </item>

</layer-list>
```

Pasta Layout - activity_main.xml

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



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

Pasta Layout - activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".mapas.MapsActivity">

    <fragment
        android:id="@+id/map_fragment"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        map:layout_constraintBottom_toTopOf="@+id/seekBar"
        map:layout_constraintEnd_toEndOf="parent"
        map:layout_constraintHorizontal_bias="0.0"
        map:layout_constraintStart_toStartOf="parent"
        map:layout_constraintTop_toTopOf="parent"
        map:layout_constraintVertical_bias="0.0"
        tools:context=".activity.MapsActivity" />

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/deepPink"
        android:max="480"
        android:padding="16dp"
        android:paddingStart="32dp"
        android:progress="0"
        android:progressTint="@color/white"
        android:thumbTint="@color/white"
        map:layout_constraintBottom_toBottomOf="parent"
        map:layout_constraintEnd_toStartOf="@+id/textSize"
        map:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textSize"
        android:layout_width="100dp"
        android:layout_height="0dp"
        android:background="@color/deepPink"
        android:gravity="center"
        android:text="200"
        android:textColor="@color/white"
        android:textSize="20sp"
        android:textStyle="bold"
        map:layout_constraintBottom_toBottomOf="parent"
```

```
    map:layout_constraintEnd_toEndOf="parent"
    map:layout_constraintTop_toTopOf="@+id/seekBar"
    map:layout_constraintVertical_bias="0.0" />

<TextView
    android:layout_width="29dp"
    android:layout_height="48dp"
    android:layout_marginEnd="4dp"
    android:background="@color/deepPink"
    android:gravity="center"
    android:text="m"
    android:textColor="@color/white"

    android:textSize="20sp"
    android:textStyle="bold"
    map:layout_constraintBottom_toBottomOf="parent"
    map:layout_constraintEnd_toEndOf="parent"
    map:layout_constraintTop_toTopOf="@+id/seekBar"
    map:layout_constraintVertical_bias="0.0" />

<Button
    android:id="@+id/button_search"
    android:layout_width="109dp"
    android:layout_height="45dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="4dp"
    android:color="@color/deepPink"
    android:contentDescription="@string/app_name"
    android:text="Pesquisar"
    android:textSize="10dp"
    android:visibility="visible"
    map:layout_constraintEnd_toEndOf="@+id/map_fragment"
    map:layout_constraintTop_toBottomOf="@+id/campo_pesquisa" />

<EditText
    android:id="@+id/campo_pesquisa"
    android:layout_width="match_parent"
    android:layout_height="54dp"
    android:layout_marginTop="4dp"
    android:background="@color/purple_500"
    android:ems="10"
    android:hint="Pesquisar por produtos..."
    android:textColorHint="@color/grey"
    android:inputType="text"
    android:textColor="@color/white"
    android:textSize="30sp"
    map:layout_constraintEnd_toStartOf="@+id/button_search"
    map:layout_constraintHorizontal_bias="0.0"
    map:layout_constraintStart_toStartOf="parent"
    map:layout_constraintTop_toTopOf="@+id/map_fragment" />
```

```
    <Button
        android:id="@+id/button"
        android:layout_width="152dp"
        android:layout_height="47dp"
        android:layout_marginStart="128dp"
        android:layout_marginTop="560dp"
        android:text="@string/ver_produtos"
        android:visibility="invisible"
        map:layout_constraintBottom_toBottomOf="@+id/map_fragment"
        map:layout_constraintEnd_toEndOf="parent"
        map:layout_constraintHorizontal_bias="0.0"
        map:layout_constraintStart_toStartOf="parent"
        map:layout_constraintTop_toBottomOf="@+id/campo_pesquisa"
        map:layout_constraintVertical_bias="0.0" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Pasta Layout - activity_products.xml

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/lista_prods"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        tools:listitem="@layout/product_item" />

    <TextView
        android:id="@+id/textFind"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nenhum produto encontrado"
        android:textSize="28sp"
        android:textStyle="bold"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/lista_prods"
        app:layout_constraintEnd_toEndOf="@+id/lista_prods"
        app:layout_constraintStart_toStartOf="@+id/lista_prods"
        app:layout_constraintTop_toTopOf="@+id/lista_prods"
        tools:ignore="UnknownId" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Pasta Layout - product_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardElevation="8dp"
    app:cardCornerRadius="8dp"
    android:layout_margin="16dp"
    android:orientation="vertical"
    >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0"
            android:orientation="vertical">
            <TextView
                android:id="@+id/show_prod"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:text="Remedio"
                android:textColor="@color/black"
                android:textSize="20sp"
                android:textStyle="bold" />
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal">

                <TextView
                    android:id="@+id/show_carac"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="8dp"
                    android:layout_weight="1"
                    android:text="Especificação"
                    android:textColor="@color/black"
                    android:textSize="18sp" />
                <TextView
                    android:id="@+id/show_price"
                    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:text="Preço"
        android:textSize="22sp"
        android:textColor="@color/black"
        android:textStyle="bold" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/show_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_weight="1"
        android:text="Nome da Loja"
        android:textSize="18sp"
        android:textColor="@color/black"/>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

    android:orientation="horizontal">

    <TextView
        android:id="@+id/show_adress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_weight="1"
        android:text="Endereço"
        android:textSize="18sp"
        android:textColor="@color/black"/>
    <TextView
        android:id="@+id/show_distance"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:text="Distancia"
        android:textSize="14sp"
        android:textColor="@color/black"/>


</LinearLayout>
```

```
        </LinearLayout>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

Pasta Values - colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="grey">#FFB3B3B3</color>
    <color name="deepPink">       #FF1493</color>

</resources>
```

Pasta themes - themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.FindPharma"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- 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>

    <style name="TemaInicial"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <item name="android:statusBarColor">@color/black</item>
        <item name="android:windowBackground">@drawable/splash_welcome</item>


    </style>
</resources>
```

APÊNDICE I - Código AndroidManifest

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

    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/TemaInicial">
        <activity
            android:name=".ProductInFarma"
            android:exported="false" />

        <uses-library android:name="com.google.android.maps" />

        <activity
            android:name=".ProductsActivity"
            android:exported="false" />
        <activity
            android:name=".mapas.MapsActivity"
            android:exported="true" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="${GOOGLE_MAPS_API_KEY}" />
    </application>

</manifest>
```

APÊNDICE J - Código build.gradle (Project: FindPharma)

```
buildscript {
    dependencies {
        classpath 'com.google.gms:google-services:4.3.10'
    }
}// Top-level build file where you can add configuration options common to all sub-
projects/modules.
plugins {
    id 'com.android.application' version '7.1.3' apply false
    id 'com.android.library' version '7.1.3' apply false
    id 'org.jetbrains.kotlin.android' version '1.6.21' apply false
}

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

APÊNDICE K - Código build.gradle (Module: FindPharma.app)

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'com.google.gms.google-services'
    id 'com.google.secrets_gradle_plugin' version '0.5'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.felipebrand.findpharma"
        minSdk 21
        targetSdk 32
        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:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.6.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'com.google.firebase:firebase-database-ktx:20.0.5'
    implementation 'com.google.android.gms:play-services-maps:18.0.2'
    implementation 'com.google.android.gms:play-services-location:19.0.1'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```