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

GEOFENCING APLICADO EM REDE DE SUPERMERCADOS COM USO DE PLATAFORMA MOBILE

RUAN CARLOS PEREIRA DO PRADO

GOIÂNIA
2023

RUAN CARLOS PEREIRA DO PRADO

GEOFENCING APLICADO NA REDE DE SUPERMERCADO EM PLATAFORMA MOBILE

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

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

GOIÂNIA
2023

RUAN CARLOS PEREIRA DO PRADO

GEOFENCING APLICADO NA REDE DE SUPERMERCADO EM PLATAFORMA MOBILE

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. MSc. Ludmilla Reis Pinheiro dos Santos

Coordenadora de Trabalho de Conclusão de Curso

Banca examinadora:

_____

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

_____

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

_____

Prof. PhD. Nilson Cardoso Amaral

GOIÂNIA

2023

RESUMO

No intuito de atender os comerciantes e principalmente os usuários das redes de supermercado, o presente trabalho apresenta um aplicativo para dispositivos móveis que possibilita a substituição do jornal de ofertas dos supermercados. Com a utilização de um sistema georreferenciado, onde o aplicativo reconhece a aproximação do usuário na determinada região ou seção, do supermercado, e de maneira automática, envia para o cliente as promoções e preços sinalizados pelos supermercados, alocado através de informações do banco de dados do estabelecimento. Deste modo, o aplicativo busca facilitar a vida do cliente que utiliza o supermercado físico, e ainda economizar na produção de papeis impressos no estabelecimento, e dessa forma inclusive uma maior proteção ao meio ambiente.

Palavras-Chave: *Android; geofence; FireBase;* supermercado

ABSTRACT

To serve merchants and especially users of supermarket chains, the present work presents an application for mobile devices that makes it possible to replace the supermarket offers newspaper. Using a georeferenced system, where the application recognizes the user's approach to a particular region or section of the supermarket, and automatically sends the customer the promotions and prices signaled by the supermarkets, allocated through information from the database. of the establishment. In this way, the application seeks to make life easier for the customer who uses the physical supermarket, and also saves on the production of printed papers in the establishment, and in this way, even greater protection to the environment.

Keywords: *Android; geofence; FireBase; supermarkets.*

LISTA DE FIGURAS

Figura 1 - Infográfico com passos sobre o funcionamento do GPS.....18
Figura 2 - Linhas latitudinais ou paralelos.....20
Figura 3 - Linhas longitudinais conhecidas como Meridianos.....20
Figura 4 - Serviços disponibilizados pelo *FireBase* no comando da Google.....27
Figura 5 - Tela de desenvolvimento do *Android Studio*.....34
Figura 6 – Print de um trecho do código AndroidManifest.xml 35
Figura 7 – Print de um trecho do código de verificação de permissão 36
Figura 8 – Print da tela de requisição de permissão 36
Figura 9 – Print do componente *Bottom Navigation*..... 37
Figura 10 – Print da tela Inicio (home) 38
Figura 11 – Print da tela Folheto 39
Figura 12 – Print da tela Promoções com usuário fora da cerca virtual..... 40
Figura 13 – Print da implementação da classe *onLocationChanged*..... 41
Figura 14 – Print da notificação 41
Figura 15 – Print da tela promoções com usuário na cerca virtual 42
Figura 16 – Print da tela Contato..... 43
Figura 17 – Print da tela Entrar 44
Figura 18 – Print do banco de dados *RealTime FireBase* 45
Figura 19 – Print do banco de dados *FireBase Storage* 46
Figura 20 – Print de um trecho do código para leitura no banco de dados 47
Figura 21 – Print trecho do código da função “setupRecyclewView” 48

LISTA DE ABREVIATURAS

Me(a)	Mestre(a)
M.E.E	Mestre em engenharia elétrica
MSc	*Master of Science*
Prof.	Professor
Dr.	Doutor
PhD.	*Philosophy Doctor*

LISTA DE SIGLAS

GPS	*Global Positioning System*
IDE	*Integrated Development Environment*
AGPS	*Assisted Global Positioning System*
NAVSTAR	*Navigation Satellite with Time And Ranging*
GSM	*Global System for Mobile*
DCS	*Digital Cellular System*
PCS	*Personal Communication System*
RFID	*Radio frequency identification*
IBM	*International Business Machines Corporation*
SQL	*Structured Query Language*
JSON	*JavaScript Object Notation*
NoSQL	*Not Only SQL*
ML	*Machine learning*
SDK	*Software Development Kit*
HTTPS	*Hyper Text Transfer Protocol Secure*
IU	Interface do Usuário
CDN	*Content Delivery Network*
NDK	*Kit de Desenvolvimento Nativo*

SUMÁRIO

1 INTRODUÇÃO 12
1.2 Objetivos 12
1.2.1 Objetivos Gerais 12
1.2.2 Objetivos Específicos 12
1.2 Justificativa 13
1.3 Metodologia 13
1.4 Descrição dos Capítulos 14

2 ESTADO DA ARTE 15
2.1 Geolocalização 15
2.1.1 GPS 16
2.1.2 Radiofrequência 20
2.1.3 Wi-Fi 20
2.1.4 AGPS 21
2.2 Geofencing 21
2.3 Banco de Dados 23
2.3.1 Banco de Dados Relacional 23
2.3.2 Banco de Dados Não Relacional 24
2.3.3 MongoDB 24
2.3.4 FireBase 25
2.3.4.1 Criar aplicativos melhores 27
2.3.4.1.1 Cloud Firestore 27
2.3.4.1.2 Kit de ML 27
2.3.4.1.3 *Cloud Functions* 27
2.3.4.1.4 Autenticação 28
2.3.4.1.5 Hospedagem 28
2.3.4.1.6 *Cloud Storages* 28
2.3.4.1.7 Banco de dados em tempo real 28
2.3.4.2 Melhorar a qualidade do aplicativo 29
2.3.4.2.1 *Crashlytics* 29
2.3.4.2.2 Monitoramento de desempenho 29
2.3.4.2.3 *Test Labs* 29

2.3.4.3 Faça seu negócio crescer 30
2.3.4.3.1 Mensagens no aplicativo 30
2.3.4.3.2 *Google Analytics* 30
2.3.4.3.3 Previsões 30
2.3.4.3.4 Teste A / B 31
2.3.4.3.5 *Cloud Messaging* (FCM) 31
2.3.4.3.6 Configuração remota 31
2.3.4.3.7 Links dinâmicos 31
2.3.4.3.8 Indexação de aplicativos 32
2.4 Android Studio 32

3 DESENVOLVIMENTO DA SOLUÇÃO 35
3.1 Implementações das permissões 35
3.2 Inicialização da aplicação 35
3.3 Telas da aplicação e suas regras 37
3.3.1 Início 37
3.3.2 Folheto 38
3.3.3 Promoções 39
3.3.4 Contato 42
3.3.5 Entrar 43
3.4 Banco de dados 44

4 CONSIDERAÇÕES FINAIS E PERSPECTIVAS FUTURAS 49
4.1 Perspectiva para trabalhos futuros 51

5.REFERÊNCIAS 52
ANEXO I 55
Apêndice 56-102

1 INTRODUÇÃO

O presente trabalho de conclusão de curso tem por finalidade, a apresentação de um aplicativo móvel para Android, que recebe promoções e preços quando o dispositivo móvel estiver próximo ou dentro da cerca virtual conhecida como *geofence*. Esta *geofence* foi implementada para enviar informações assim que o usuário adentrar nas dependências do supermercado e adjacências. Essas informações consistem em promoções diárias, mensais ou mesmo atemporais definidas pelo próprio estabelecimento. O usuário aproveita-se dos preços baixos e descontos exclusivos além de outras informações.

O aplicativo conta com um banco de dados, o qual possui informações de todos os produtos elencados no supermercado tais como, frutas, produtos de limpeza, utensílios e demais itens. Além de contar com imagens e referências dos produtos em uma tabela com as promoções do dia. A aplicação utiliza-se da linguagem Java, juntamente da *Integrated Development Environment* (IDE) *Android Studio* como base para o desenvolvimento do aplicativo, o banco de dados conta com a tecnologia da *Google*, o *Google FireBase*.

1.1 Objetivos

Esta seção tem por finalidade apresentar e descrever os objetivos gerais e específicos deste trabalho.

1.1.1 Objetivos Gerais

Desenvolver um aplicativo *Android* utilizando a tecnologia *geofencing,* para localizar clientes nas dependências de um supermercado, tomando como base uma cerca virtual vinculada à posição do supermercado e ao cruzar o espaço delimitado, os clientes recebem instantaneamente uma notificação no aplicativo com promoções diárias ou mensais dentre outras que o supermercado determinar.

1.1.2 Objetivos Específicos

- Aplicar *Geofencing* a automação comercial,

- Implementar um aplicativo para *Android* utilizando o *Geofencing*,
- Alimentar um banco de dados com os produtos do supermercado para serem utilizados pelo aplicativo,
- Desenvolver o sistema com o intuito de facilitar as compras no supermercado,
- Desenvolver o sistema para substituir os jornais de ofertas tradicionais dos supermercados.

1.2 Justificativa

O presente trabalho justifica-se por possibilitar a integração de *Geofencing* a automação comercial juntamente com as tecnologias presentes nos celulares Android atuais.

O aplicativo em questão, traz para o cliente da rede de supermercado uma maior facilidade e simplicidade para conhecer os preços dos produtos e suas características, ao invés de buscar no antigo jornal de ofertas, que geralmente é grande e desajeitado além de representar um desperdício do ponto de vista ambiental, devido ao descarte normalmente inadequado deste panfleto, e desta forma barateando para o supermercadista, evitando a confecção dos jornais que são de alto custo. E assim contribuindo para uma melhora generalizada tanto do ponto de vista do cliente quanto do proprietário do estabelecimento. Com o aplicativo instalado no *mobile*, o cliente adentra as propriedades do supermercado e automaticamente recebe uma notificação no celular. Sem qualquer dificuldade o cliente acessa as promoções de forma rápida e portátil sem prejudicar o passeio pelo supermercado. Dessa forma, a aplicação tem o intuito de facilitar a vida do cliente que estaria presencialmente no mercado e gosta de acompanhar as promoções durante as compras de forma a fazer suas melhores escolhas de produtos certos**.**

1.3 Metodologia

Inicialmente foi realizado uma revisão bibliográfica criteriosa baseada em artigos científicos, monografias, revistas, livros, e vários outros meios de informação confiáveis. Simultaneamente, também foi realizado o desenvolvimento da base de dados utilizando informações disponíveis do estabelecimento, para assim construir o

banco de dados não relacional, que por sua vez será implementado utilizando o *Google FireBase*, e nele terá fotos de produtos, códigos, preços, nome de produtos, dentre outras informações. Após finalizar essa etapa, será construído o aplicativo, no qual será desenvolvido através da IDE *Android Studio*.

1.4 Descrição dos capítulos

O capítulo 1 apresenta o objetivo do trabalho, juntamente com a metodologia que foi utilizada neste presente trabalho.

O capítulo 2 demonstra detalhadamente a fundamentação teórica, apresentando as ferramentas que foram usadas neste trabalho, como o GPS, Google FireBase e Android Studio.

O capítulo 3 traz desenvolvimento da solução, onde é apresentado o aplicativo funcional e a explicação detalhada, com *prints* do código e da tela.

O capítulo 4 apresenta as considerações finais e as perspectivas para futuras inspirações deste trabalho.

2 ESTADO DA ARTE

Este capítulo apresenta a fundamentação teórica necessária para a concepção e elaboração deste presente trabalho, visando a orientação e compreensão a respeito dos conceitos e definições utilizados no desenvolvimento do aplicativo *Android*, no qual se baseia o objetivo deste trabalho.

2.1 Geolocalização

É um recurso utilizado para localizar uma determinada posição geográfica de um dispositivo, baseado em um sistema de coordenadas espalhadas pelo globo terrestre. Essas coordenadas geográficas, podem ser obtidas por meio de diferentes tipos de sinais, como *Global Positioning System* (GPS), Radiofrequência, Wi-Fi ou *Assisted Global Positioning System (A-GPS)* (BRANDÃO, 2019).

Uma forma bastante conhecida para fazer a representação cartográfica é o sistema de Coordenadas Geográficas. São utilizadas para representar e localizar qualquer ponto da superfície terrestre. É composto por linhas imaginarias, denominadas Latitudes e Longitudes (PENA, 2020).

A distância em graus de qualquer ponto da Terra em relação à linha do equador é conhecida como Latitude ou paralelo. O globo terrestre tem como principais paralelos: o círculo polar ártico, o círculo polar antártico, o trópico de câncer e o trópico de capricórnio. As latitudes podem variar de 0º a 180º, sendo contadas a partir da linha do Equador, marco 0º, responsável por dividir o planeta em dois hemisférios Norte (Boreal ou Setentrional) e Sul (Austral ou Meridional) (PENA, 2020).

A distância, em graus, de qualquer ponto da Terra em relação ao meridiano de *Greenwich* é conhecida como longitude ou meridianos. Os meridianos variam entre 0ª e 180ª, para leste ou oeste, sendo medidas a partir do meridiano de *Greenwich* (PENA, 2020).

As longitudes e latitudes são traçadas no globo a cada 15º, no entanto, qualquer ponto da superfície terrestre possui latitude e longitude específicas. No âmbito das representações gráficas, o único ponto de referência utilizado são as coordenadas geográficas traçadas a partir dessas linhas (PENA, 2020).

De acordo com o Professor Giovanini (2014), normalmente os pontos geográficos são apresentados não somente em graus e sim no formato de graus,

minutos e segundos, sendo assim cada local do planeta terá uma única coordenada, como por exemplo a representação: 30º 10’ 13,57”;

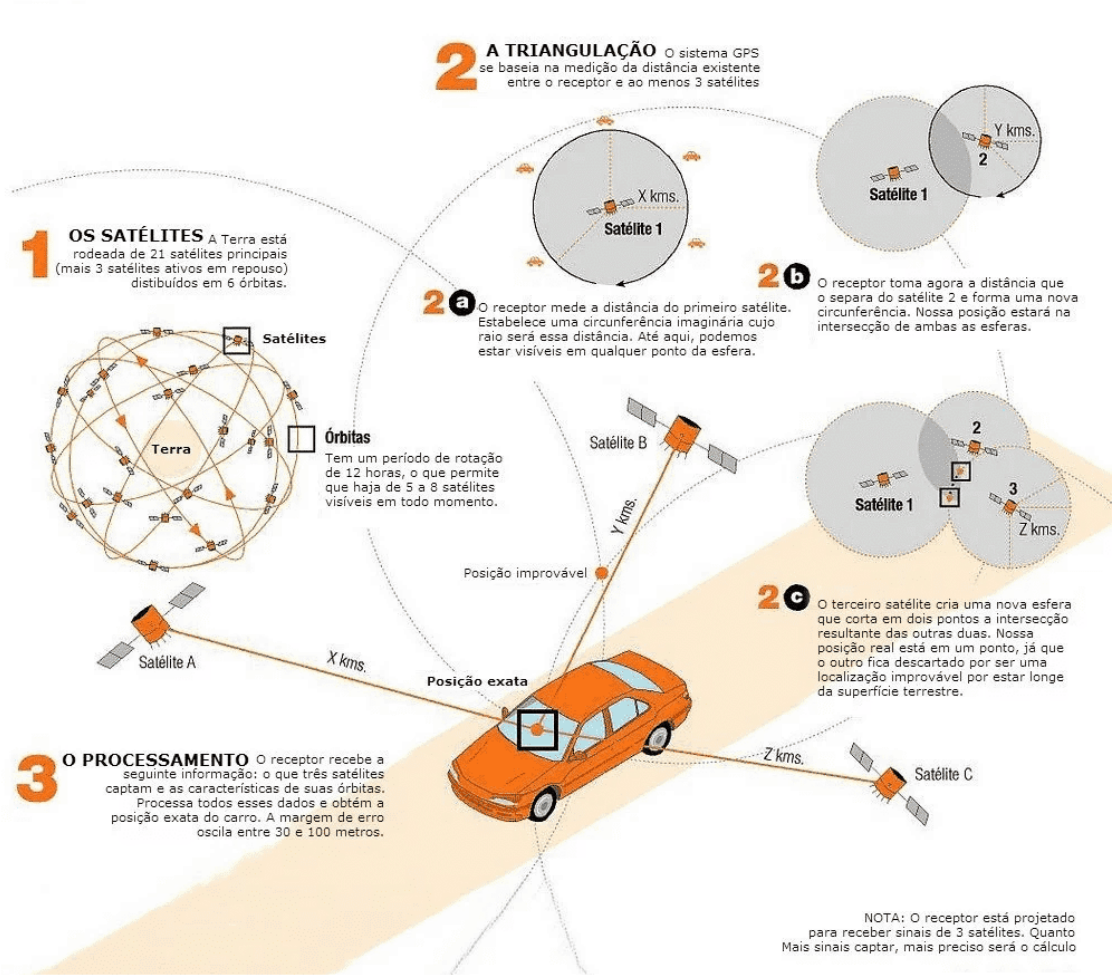
Onde:

- º é o símbolo utilizado para graus;
- ‘ é o símbolo utilizado para minutos e;
- “ é o símbolo utilizado para segundos.

2.1.1 GPS

Possivelmente é o dispositivo com um método tecnológico mais popular em utilização para se obter as coordenadas em questão. Ele funciona através de dados fornecidos por satélites que estão posicionados na órbita do planeta Terra. Para encontrar a posição de um objeto através do GPS, é necessário que o dispositivo capte o sinal de pelo menos três satélites, pois assim permite a triangulação de dados, após o processamento destes dados, revela com grande precisão a posição do objeto. A Figura 01 é um infográfico que traz a explicação do funcionamento de um sistema de rastreamento veicular utilizando o GPS (BRANDÃO, 2019).

Figura 1 – Infográfico com passos sobre o funcionamento do GPS



Fonte: Brandão, 2019

O sistema de GPS da atualidade funciona com uma constelação de 32 satélites NAVSTAR (*Navigation Satellite with Time And Ranging*), sendo 24 ativos e 8 reservas, prontos para utilização caso ocorra alguma falha nos satélites ativos, que orbitam a Terra com um ciclo aproximado de duas voltas por dia, de tal modo que em qualquer ponto da superfície terrestre, haja pelo menos 4 satélites visíveis e nunca estejam no mesmo plano geométrico (coplanares), condições essenciais para determinar a posição de maneira única (FAGGIAN, 2019).

Os satélites são monitorados por bases fixas na superfície e se comunicam entre si, fazendo a verificação constantemente de sua trajetória e posição com máxima precisão, efetuando ajustes nas informações se necessário. Os satélites emitem simultaneamente um sinal eletrônico de GPS, e esses sinais viajam à velocidade da luz, e esses sinais estão codificados com a assinatura do satélite, sua posição e horário de emissão (FAGGIAN, 2019).

De acordo com Santana (2019), devido a sua acessibilidade e sua popularização, o GPS foi implementado também nas plataformas *mobile*, um grande exemplo é o aplicativo Google Maps. O sistema de localização dos *smartphones* funciona de três modos distintos, sendo eles:

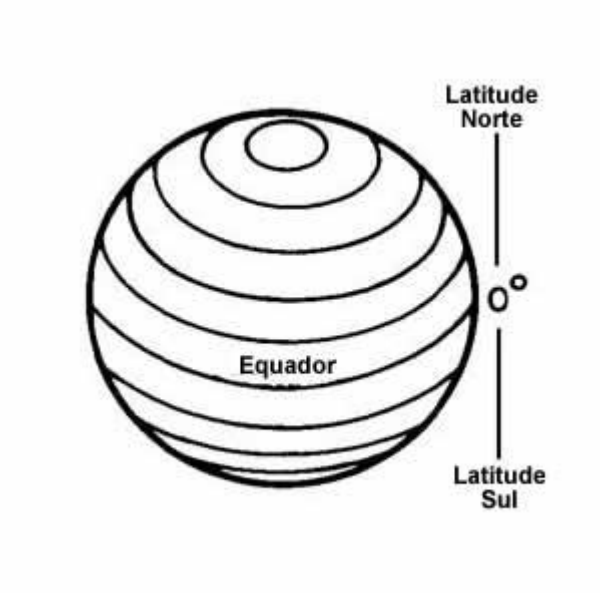
- Alta precisão: utiliza de informações de localização através da rede móvel e/ou internet com informações do receptor GPS do dispositivo para determinar a localização;
- Economia de bateria: faz uso apenas de informações de localização vindas da rede móvel e/ou internet e assim determinar a localização;
- Somente no dispositivo: usa-se apenas as informações vinda do receptor do GPS do smartphone para determinar a localização.

A escolha do modo apropriado, é geralmente feita pelo próprio sistema operacional do smartphone. Ele leva em conta a disponibilidade de pontos de acesso à internet ou rede de telefonia. Para algumas aplicações não é necessária uma grande precisão e, portanto, uma estimativa já é o suficiente, no entanto, algumas podem exigir informações mais precisas. O nível exato de precisão é raramente utilizado como parâmetro de definição do modo apropriado, embora influencie significativamente na qualidade do serviço prestado (MICHAHELLES E WATZDORF, 2010).

O GPS utiliza-se o sistema de coordenadas geográficas, esse sistema faz uso de linhas imaginárias que circundam o globo terrestre, popularmente conhecidas como paralelos e meridianos (TEMBHEKAR; SAKHARE, 2021).

O paralelo principal do globo terrestre é a linha do Equador, que possui latitude 0º e divide o globo terrestre em dois hemisférios, sendo eles Norte e Sul. Assim sendo, latitude é a distância, em graus, de qualquer ponto da superfície terrestre em relação à linha do Equador. A latitude poder ser Norte (N) ou Sul (S) e vai de 0º até 90º, como mostra a Figura 2 (LÁZARO, 2015).

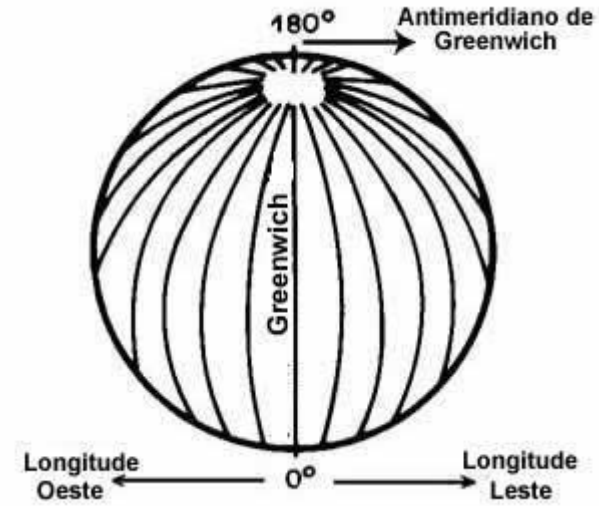
Figura 2 - Linhas latitudinais ou paralelos



Fonte: Adaptado de Lázaro, 2015

O meridiano de *Greenwich*, por convenção, foi estabelecido como meridiano principal. Esse meridiano (0º) e o seu antimeridiano (180º) dividem o globo terrestre em dois hemisférios, sendo eles leste (oriental) e oeste (ocidental). Assim sendo, longitude é a distância, em graus, de qualquer ponto da Terra em relação ao meridiano de *Greenwich*. A longitude pode ser Leste (L) ou Oeste (O) e vai de 0º a 180º, como mostrando na Figura 3 (LÁZARO, 2015).

Figura 3 - Linhas longitudinais conhecidas como Meridianos



Fonte: Adaptado de Lázaro, 2015

A interseção do paralelo com o meridiano indica a coordenada geográfica de um ponto sobre a superfície terrestre. E a partir disso, obtém a latitude e a longitude, com os quais pode-se localizar com precisão, algo na superfície terrestre, seja num continente, numa ilha, ou num oceano. Em tempo, o nome do meridiano de *Greenwich* deriva da localidade na região metropolitana de Londres, onde fica o Observatório Real de *Greenwich* (LÁZARO, 2015).

2.1.2 Radiofrequência

Radiofrequência ou GSM (*Global System for Mobile*), é um tipo de geolocalização que funciona por meio de ondas de rádio, utilizando dados fornecidos por torres de operadoras de telefonia móvel. O sinal é captado, faz a triangulação e assim é possível obter a posição do objeto. Este método tem uma grande área de cobertura e permite localizar qualquer dispositivo que esteja ligado e com sinal de operadora, mesmo com a opção de GPS desabilitada (BRANDÃO, 2019).

O GSM é atualmente utilizado em diversos países. Ele começou a ser popularizado na Europa e em seguida se difundiu pelo mundo. Esta tecnologia pode ser separada em 3 sistemas, GSM900, DCS1800 (*Digital Cellular System*) e 2PCS1900 (*Personal Communication System*), sendo o número após a sigla a frequência com a qual o sistema trabalha. A operadora de telefonia móvel TIM SUL, por exemplo, faz uso especificamente das frequências 900 e 1800 MHz (CORDEIRO, 2011).

A rede GSM é formada por várias entidades, sendo sua arquitetura muito flexível, o que torna extremamente viável e vantajosa para as operadoras dos sistemas móveis de celulares. Pode ser dividida em três partes principais, sendo estação móvel, o subsistema estação base e o subsistema de rede (CORDEIRO, 2011).

2.1.3 Wi-Fi

O Wi-Fi é um método indoor, ou seja, é aquele realizado em um local fechado. A geolocalização via Wi-Fi funciona como qualquer conexão sem fio, fazendo uso da intensidade do sinal para determinar a localização do dispositivo. Semelhante à

radiofrequência, este método também não necessita que o GPS esteva ativo no dispositivo (BRANDÃO, 2019).

A localização da posição através do Wi-Fi funciona da seguinte forma: o dispositivo analisa os pontos de acesso Wi-Fi ao alcance do próprio dispositivo e cria uma lista, que inclui a intensidade do sinal de cada um. E assim o dispositivo contacta servidores que contêm uma lista de pontos de acesso Wi-Fi de todo o mundo que inclui as suas posições geográficas. Estas bases de dados incluem uma lista dos nomes das redes Wi-Fi, os endereços MAC únicos. Após uma comparação do conteúdo da lista das redes Wi-Fi, que estão perto dos dispositivos, com a lista dos pontos de acesso conhecidos, o serviço de localização consegue saber a sua localização aproximada. A precisão é comparável à do que se consegue através do GPS (TROIA, 2021).

2.1.4 AGPS

Conhecido também como GPS assistido, este é um método de geolocalização que funciona como uma fusão entre os métodos de GPS e de radiofrequência, ou seja, ele utiliza tanto os dados obtidos via satélite, quanto os de operadoras de telefonia móvel (BRANDÃO, 2019).

2.2 *Geofencing*

O *geofencing* é um serviço baseado em localização, onde um aplicativo ou outro software usa GPS, RFID (*Radio frequency identification*), Wi-Fi ou dados de celular para acionar uma ação pré-definida quando um dispositivo móvel ou etiqueta RFID que entra na cerca virtual configurada em um local (WHITE, 2017).

A *geofence* possui diversos tipos de funções, e tudo depende de como ela foi configurada, ela pode solicitar notificações *push* móveis, acionar mensagens de texto ou alertas, enviar anúncios direcionados nas mídias sociais, permitir o rastreamento de frotas de veículos, desabilitar certas tecnologias ou fornecer dados de marketing baseado na localização (WHITE, 2017).

Para usar o *geofencing,* primeiramente o desenvolvedor deve estabelecer um limite virtual em torno de um local. Esta “geocerca virtual” irá disparar uma resposta

quando um dispositivo autorizado entra ou sai dessa área pré-estabelecida pelo desenvolvedor (WHITE, 2017).

A *geofence* ou geocerca pode ser configurada por usuário finais, usando recursos de *geofencing* em seus aplicativos móveis. Esses aplicativos permitem ao usuário escolher um endereço para criar a sua geocerca, e assim que atravessada ela aciona um alerta específico ou uma notificação tipo *push* (WHITE, 2017).

A geocerca é mais comumente definida dentro do código de um aplicativo móvel. Especificamente, os usuários precisam optar e aceitar os serviços de localização para que a geocerca funcione corretamente. Ela é programada no aplicativo pelo dono do estabelecimento, permitindo que os usuários escolham se desejam ou não permitir o acesso ao local por meio do aplicativo. (CIO/EUA, 2018).

Assim que uma área geográfica é definida, as oportunidades para de uso para as empresas são aparentemente infinitas, especialmente na área de marketing. Saber que os clientes estão por perto permite que as empresas personalizem ofertas exclusivas com base em eventos locais ou feriados (CIO/EUA, 2018).

Existem alguns cuidados com *geofencing* que precisam ser levados a sério, especialmente quando se trata de privacidade com marketing. Em 2016, o estado americano de Massachusetts foi um dos primeiros a promulgar uma lei de proteção ao consumidor que se opunha ao uso de publicidade baseada em localização. É amplamente reconhecido na indústria que os dados de localização precisos são confidenciais, portanto, qualquer prática que se baseie nesses dados deve considerar como o geofencing está sendo implementado e como os usuários estão sendo notificados (CIO/EUA, 2018).

Além do marketing para as empresas, a *geofencing* tem aplicação no monitoramento de frotas e equipes externas. Associado à tecnologia de telemática, o *geofencing* pode ser útil tanto no acompanhamento do deslocamento e na execução das atividades, quanto no envio de mensagens e alertas para os dispositivos dos colaboradores, além de possibilitar a automatização do sistema de relógio ponto (BRANDÃO, 2020).

Ainda pode-se utilizá-la para fazer a automação de casas ou empresas. Há uma gama de situações em que há possibilidades de configurar o uso da *geofencing.* Um exemplo de automação residencial, supondo que uma pessoa more em uma região muito fria e está se deslocando para sua casa. E essa pessoa esteja chegando a uma certa distância pré-configurada de sua casa, o aquecedor ou ar-condicionado se

ligaria automaticamente, e assim que chegar na sua casa terá um clima perfeito para a sua chegada. Além disso, pode-se configurar perfis diferentes em seu celular, dependo do local em que esteja, por exemplo, quando estiver em casa, pode dispensar a necessidade de senha para o desbloqueio do aparelho (BRANDÃO, 2020).

2.3 Banco de Dados

Um banco de dados é um repositório sistêmico de informações. Elas podem ser, por exemplo, dados de clientes de um cómercio, dados internos de uma empresa, e muitos outros dados (ROVEDA, 2021).

Portanto um banco de dados é uma estrutura de software que permite que dados sejam armazenados, organizados, protegidos, atualizados, acrescentados, excluídos e acessados sempre que necessário ou que for programado (ALECRIM, 2018).

Os bancos de dados armazenam e dão acesso rápido às informações, além disso é por meio dele que os analistas de suporte cliente/usuário têm acesso às bases de dados de uma empresa e podem pegar informações rapidamente. Outro ponto extremamente importante é a segurança da informação. Antigamente os dados ficavam armazenados em vários sistemas ou arquivos, e com o banco esses dados ficam concentrados em um único lugar protegido, para evitar que eles sejam danificados ou roubados por invasores (REDAÇÃO IMPACTA, 2016).

Ainda pode-se usar o banco de dados para automatizar uma série de procedimentos. Sem o banco seria necessário verificar as transações financeiras manualmente, com o banco por exemplo, os responsáveis podem trabalhar com bases de dados e parametrizar comandos para que elas gerem relatórios automáticos (REDAÇÃO IMPACTA, 2016).

2.3.1 Banco de Dados Relacional

Um banco de dados relacional é uma coleção de elementos de dados organizados em um conjunto de tabelas formalmente descritas, a partir das quais as informações podem ser acessadas ou remontadas de muitas formas diferentes, sem a necessidade de reorganizar as tabelas do banco de dados. O banco de dados

relacional foi inventado em 1970 por Edgar Frank Codd, na IBM (*International Business Machines Corporation*). (TECHTARGET, 2021)

O programa de usuário padrão e a interface de aplicativo para um banco relacional é a linguagem SQL (*Structured Query Language*). Os comandos dessa linguagem são utilizados para consultas interativas, para coleta de relatórios (TECHTARGET, 2021).

2.3.2 Banco de Dados Não Relacional

Um banco de dados não relacional é um banco de dados que não usa o esquema de tabela de linhas e colunas como no método anterior. Em vez disso, esse modelo usa o método de armazenamento otimizado para os requisitos específicos do tipo de dados que está sendo armazenado. Por exemplo, os dados podem ser armazenados como pares chave/valor simples, como documentos JSON (*JavaScript Object Notation)* ou como um gráfico que consiste em bordas e vértices (TEJADA, 2021).

Esse método não usa a linguagem SQL, e sim outras linguagens de programação e componentes para consultar os dados, portanto, faz uso do termo “NoSQL” (*Not Only SQL*). Na prática, o termo significa banco de dados não relacionais, mesmo que muitos desses bancos de dados deem suporte a consultas compatíveis com SQL (TEJADA, 2021).

2.3.3 MongoDB

É um banco de dados *opensource*, de alta performance e flexível, sendo considerado o principal banco de dados NoSQL. O MongoDB é orientado a documentos, ou seja, os dados são armazenados como documentos, ao contrário de bancos de dados de modelo relacional, onde é trabalhado com registros em linhas e colunas. Os documentos podem ser descritos como dados no formato de chave-valor, no caso, utilizando o formato JSON (GUEDES, 2020).

O MongoDB foi desenvolvido para oferecer escalabilidade, desempenho e alta disponibilidade, desde a implantação de um único servidor até grandes arquiteturas complexas de vários centros de dados. A replicação nativa do MongoDB e a tolerância

automática a falhas oferecem confiabilidade e flexibilidade operacional em toda a empresa (GUEDES, 2020).

Para fazer uso do MongoDB é necessário baixá-lo, escolhendo uma versão dependendo do sistema operacional. Através do aplicativo de *Shell* do MongoDB, pode-se criar o banco de dados, documentos e coleções. Por padrão, o *shell* do Mongo conecta automaticamente ao banco de dados “test” e para mudar para outro banco, basta utilizar o comando “use nomeDoBanco”. Caso o banco mencionado não existir, o MongoDB o criará automaticamente quando forem incluídos dados nele (MEDEIROS, 2014).

De acordo com Marylene Guedes (2020), o MongoDB possui algumas características que o tornam uma das melhores opções para incorporar como banco de dados no desenvolvimento de aplicações:

- Sintaxe para consultas: permite consultas simples e as mais complexas, podendo obter todos os tipos de informações;
- Indexação: pode-se criar índices, igual nos modelos relacionais;
- Escalabilidade horizontal: Aumenta seu tamanho de acordo com o uso de armazenamentos, implica na divisão do conjunto de dados e a carga de vários servidores adicionais para aumentar a capacidade;
- Permite executar consultas executando código *JavaScript*;

2.3.4 FireBase

O Google *FireBase* é uma plataforma digital utilizada para facilitar o desenvolvimento de aplicativos web ou móveis, de uma forma efetiva, rápida e simples. Com as suas diversas funções gratuitas, é utilizado como uma técnica de Marketing Digital, com a finalidade de aumentar a base de usuários e gerar maiores benefícios econômicos. O seu principal objetivo é melhorar o rendimento dos apps mediante a implementação de diversas funcionalidades que farão do aplicativo um instrumento muito mais maleável, seguro e de fácil acesso para os usuários (CONTENT, 2019).

Utilizando o *FireBase* é possível oferecer experiências de aplicativos mais ricas para o usuário, otimizando a performance e a experiência da plataforma. Além disso, o *FireBase* é versátil podendo ser implementado em vários sistemas, sejam eles o *Android*, *iOS* e *Web* (SILVA, 2021).

No ano de 2011, antes da ferramenta ser conhecida como *FireBase*, era uma *startup* chamada Envolve. Como Envolve, forneceu aos desenvolvedores uma API que permitiu a integração da funcionalidade de bate-papo online em seu site. Os desenvolvedores estavam usando o Envolve para sincronizar dados do aplicativo, como o estado do jogo em tempo real, entre seus usuários. Após um tempo acarretou aos fundadores da Envolve, James Tamplin e Andrew Lee, a separar o sistema de bate-papo da arquitetura em tempo real. Em abril de 2012, o *FireBase* foi criado como uma empresa separada que fornecia *Backend-as-a-Service* (serviço disponibilizado onde toda a estrutura do *backend* como: configuração de servidor, integração com banco de dados, sistema de *push notification* e outros serviços, estão completamente prontos para se integrar com o seu aplicativo.) com funcionalidade em tempo real. Depois de ter sido adquirido pelo Google em 2014, o *FireBase* evoluiu rapidamente para o gigante multifuncional de uma plataforma móvel e web que é hoje, como mostrado na Figura 4, as funcionalidades que surgiu após a compra da google.

Figura 4 – Serviços disponibilizados pelo *FireBase* no comando da Google

Fonte: SINGH, 2018

Como apresentado na Figura 4 anterior, o *FireBase* demonstra vários serviços e facilidades para o desenvolvedor. . Dentro do FireBase existe grupos de aplicações para ajudar o desenvolvedor, sendo elas: “criar aplicativos melhores”, “melhorar a qualidade do aplicativo” e “fazer seu negócio crescer” (SINGH, 2018).

2.3.4.1 Criar aplicativos melhores

O *FireBase* permite criar aplicativos mais poderoso, seguros e escalonáveis, usando uma infraestrutura de classe mundial usando algumas aplicações próprias (SINGH, 2018).

2.3.4.1.1 Cloud Firestore

É um banco de dados flexível e escalonável para desenvolvimento móvel, web e servidor do *FireBase* e Google *Cloud Platform*. É um banco de dados de documentos que faz uso do NoSQL que permite armazenar, sincronizar e consultar dados facilmente para seus aplicativos móveis e da web (SINGH, 2018).

2.3.4.1.2 Kit de ML

É um SDK (*Software Development Kit* - conjunto de ferramentas que possibilita aos programadores a criação de novas aplicações) móvel que traz a experiência de aprendizado de máquina do Google para aplicativos mobile, *Android* e *iOS*, em um pacote poderoso e fácil de se usar. Seja novo ou experiente no aprendizado de máquina, pode-se implementar a funcionalidade de que precisa em apenas algumas linhas de código, fácil e rapidamente. Não é necessário ter um conhecimento profundo de redes neurais ou otimização de modelo para começar. Por outro lado, para um desenvolvedor de ML (*Machine learning*) experiente, o Kit de ML fornece APIs convenientes que ajudam a usar seus modelos personalizados do *TensorFlow Lite* em seus aplicativos móveis (SINGH, 2018).

2.3.4.1.3 *Cloud Functions*

Permite a execução automática do código de *back-end*, em resposta a eventos acionados por recursos do *FireBase* e solicitações HTTPS (*Hyper Text Transfer Protocol Secure*). O código é armazenado na nuvem do Google e executado em um ambiente gerenciado. Não há necessidade de gerenciar e dimensionar os próprios servidores. Esta funcionalidade está disponível para Android, iOS, C ++, Unity e plataforma web (SINGH, 2018).

2.3.4.1.4 Autenticação

Fornece serviços de *back-end*, SDKs fáceis de usar e bibliotecas de IU (Interface do Usuário) prontas para autenticar usuários em seu aplicativo. Ele suporta autenticação usando senhas, números de telefone, provedores de identidade federados populares como Google, Facebook, Twitter e entre outros (SINGH, 2018).

2.3.4.1.5 Hospedagem

É a hospedagem de conteúdo da web de nível de produção para desenvolvedores. Com um único comando, é possível implementar rapidamente aplicativos da web e servir conteúdo estático e dinâmico a uma CDN (*Content Delivery Network*) global, isso somente para a plataforma web (SINGH, 2018).

2.3.4.1.6 *Cloud Storages*

Serviço de armazenamento de objetos desenvolvido para escala do Google. Os SDKs do *FireBase* para armazenamento em nuvem adicionam segurança do Google a uploads e downloads de arquivos para seus aplicativos *FireBase*, independentemente da qualidade da rede. Pode-se usar os SDKs para armazenar imagens, áudio, vídeo ou outro conteúdo gerado pelo usuário. No servidor, pode-se usar o Google Cloud Storage para acessar os mesmos arquivos. Estão disponíveis nas versões Android, iOS, C ++, Unity e plataforma web (SINGH, 2018).

2.3.4.1.7 Banco de dados em tempo real

É um banco de dados NoSQL hospedado na nuvem que permite armazenar e sincronizar entre seus usuários em tempo real. O *Real-time Database* é realmente apenas um grande objeto JSON que os desenvolvedores podem gerenciar em tempo real (SINGH, 2018).

2.3.4.2 Melhorar a qualidade do aplicativo

O *FireBase* oferece *insights* sobre o desempenho e a estabilidade do aplicativo, para que seja possível canalizar seus recursos e tempo de maneira mais eficaz, usando as seguintes funções do *FireBase* (SINGH, 2018).

2.3.4.2.1 *Crashlytics*

É um relator de falhas, leve e em tempo real que ajuda a rastrear, priorizar e corrigir problemas de estabilidade que prejudicam a qualidade do seu aplicativo. A utilização do *Crashlytics* economiza tempo na solução de problemas, agrupando as falhas de maneira inteligente e destacando as circunstâncias que levaram a elas, ajudando o desenvolvedor a corrigi-las e lançar um aplicativo com maior qualidade. Está disponível apenas para plataforma mobile (SINGH, 2018).

2.3.4.2.2 Monitoramento de desempenho

É um serviço que ajuda o desenvolvedor a obter informações sobre as características de desempenho de seus aplicativos iOS e Android. Use o SDK do Monitoramento de desempenho para coletar dados de desempenho do seu aplicativo e, em seguida, revise e análise esses dados no console do *FireBase*. O Monitoramento de desempenho ajuda a entender onde e quando o desempenho do aplicativo que possa ser melhorado, para que possa usar essas informações para corrigir problemas de desempenho (SINGH, 2018).

2.3.4.2.3 *Test Labs*

É uma infraestrutura de teste de aplicativos, baseada em nuvem. Fornece muitos dispositivos moveis para ajudar a testar os aplicativos (SINGH, 2018).

2.3.4.3 Faça seu negócio crescer

O *FireBase* facilita atingir milhões de usuários, simplificando o envolvimento e a retenção de usuários usando as funções a seguir (SINGH, 2018).

2.3.4.3.1 Mensagens no aplicativo

Ajuda a se conectar com os usuários que estão usando ativamente o aplicativo, enviando-lhes mensagens direcionadas e contextuais que os estimulam a concluir ações importantes no aplicativo, como por exemplo, superar um nível de jogo, comprar um item ou assinar um conteúdo (SINGH, 2018).

2.3.4.3.2 *Google Analytics*

É um aplicativo gratuito que fornece informações sobre o uso do aplicativo e o envolvimento do usuário. No coração do *FireBase* está o Google *Analytics*, uma solução analítica gratuita e ilimitada. O *Analytics* se integra aos recursos do *FireBase* e fornece relatórios ilimitados para até 500 eventos distintos que é possível definir usando o SDK do próprio *FireBase*. Os relatórios de análise ajudam a entender claramente como os usuários se comportam, o que torna mais fácil a tomada de decisões (SINGH, 2018).

2.3.4.3.3 Previsões

Aplica o aprendizado de máquina aos seus dados analíticos para criar segmentos de usuários dinâmicos com base no comportamento previsto dos usuários no aplicativo. Essas previsões estão automaticamente disponíveis para uso com o *FireBase Remote Config*, o Editor de Notificações, *FireBase In-App Messaging* e A / B *Testing*. Também pode-se vincular os dados do *Predictions* do aplicativo ao *BigQuery,* para obter exportações diárias que podem ser analisadas posteriormente ou enviadas para ferramentas de terceiros para análise. (SINGH, 2018).

2.3.4.3.4 Teste A / B

Facilita a otimização da experiência do aplicativo, tornando mais fácil a execução, análise e dimensionar experimentos de produto e marketing. Entrega para o desenvolvedor o poder de testar mudanças na interface do usuário, recursos ou campanhas de engajamento do seu aplicativo para ver se eles realmente mexem com suas principais métricas antes de implementá-las amplamente. Esta funcionalidade não está disponível na versão web (SINGH, 2018).

2.3.4.3.5 Cloud Messaging (FCM)

O *Firebase* oferece uma conexão confiável e de baixo consumo de bateria entre o servidor e os dispositivos, proporcionando uma experiência de comunicação perfeita. Além disso, ele permite o envio e recebimento de mensagens e notificações em plataformas como iOS, Android e web, tudo isso sem custo adicional. Com o *Firebase*, é possível garantir uma comunicação eficiente e sem interrupções nos aplicativos (SINGH, 2018).

2.3.4.3.6 Configuração remota

Permite publicar atualizações para os usuários imediatamente. Seja a alteração do esquema de cores de uma tela, o *layout* de uma seção específica no aplicativo ou mostrar opções promocionais/sazonais, isso é totalmente possível usando os parâmetros do lado do servidor, sem a necessidade de publicar uma nova versão (SINGH, 2018).

2.3.4.3.7 Links dinâmicos

Com os links dinâmicos, os usuários do aplicativo obtêm a melhor experiência disponível para a plataforma em que abrem seu link. Se um usuário abre um link dinâmico no iOS ou Android, ele pode ser levado diretamente para o conteúdo vinculado em seu aplicativo nativo. Se um usuário abrir o mesmo *Dynamic Link* em um navegador de desktop, ele poderá ser levado ao conteúdo equivalente em seu site (SINGH, 2018).

2.3.4.3.8 Indexação de aplicativos

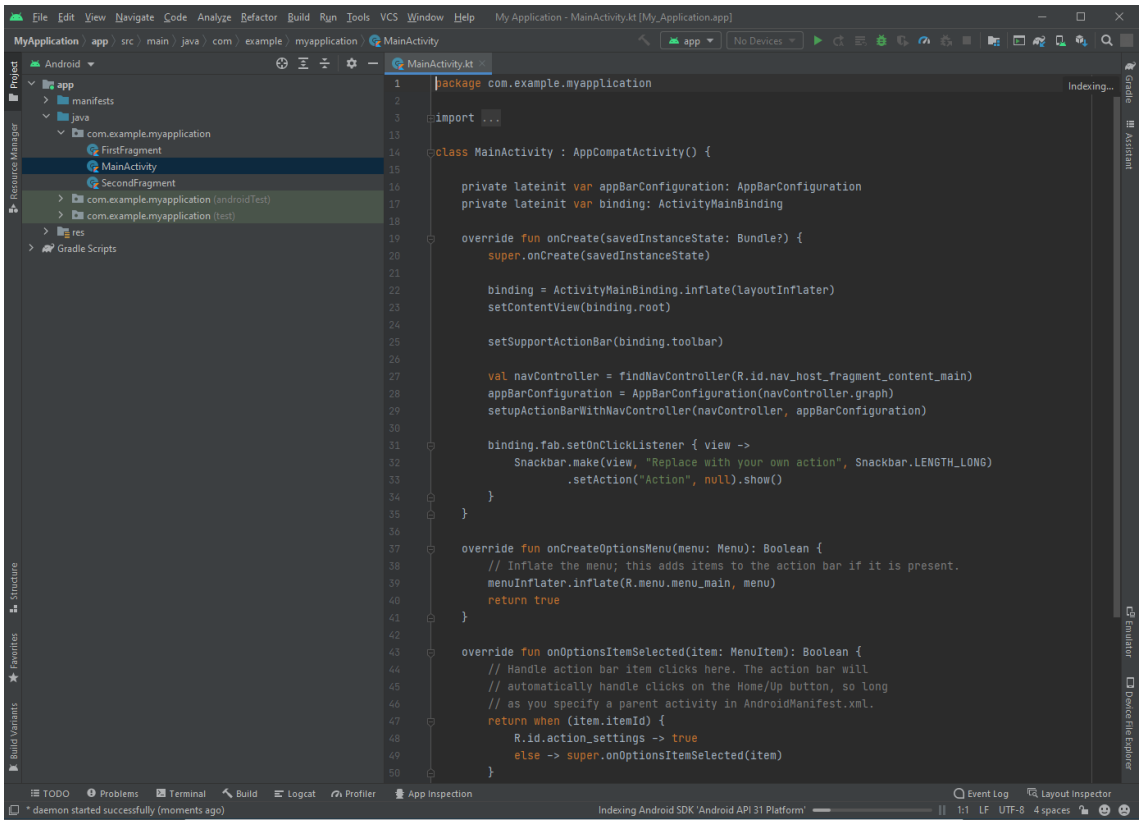
Coloca o aplicativo na Pesquisa Google. Se os usuários tiverem o aplicativo instalado, eles podem iniciá-lo e ir diretamente para o conteúdo que procuram. A Indexação de aplicativos reengaja os usuários de seu aplicativo, ajudando-os a encontrar conteúdo público e pessoal diretamente em seus dispositivos, oferecendo até mesmo preenchimento automático de consultas para ajudá-los a encontrar o que precisam com mais rapidez. Caso os usuários ainda não tenham o aplicativo, as consultas relevantes acionam um cartão de instalação para o seu aplicativo nos resultados da pesquisa. Disponível apenas nas versões mobile (SINGH, 2018).

2.4 Android Studio

O *Android Studio* é uma IDE que foi criada pela Google. Esse ambiente de desenvolvimento possui várias funcionalidades, tais como preenchimento automático de comandos, auto indentação, navegador de pasta, simulação de dispositivo, dentre outras funcionalidades. Além disto, ele possui uma compatibilidade de remodelação com o *Android Wear*, um relógio com sistema operacional Android, portanto, é extremamente viável a utilização dessa IDE para facilitar a vida do desenvolvedor (HELLMANN, 2016).

O *Android Studio* oferece um ambiente de desenvolvimento robusto, com compatibilidade de simulação a vários dispositivos móveis, e é uma IDE com muitas ferramentas extras para facilitar ainda mais a vida do programador (HELLMANN, 2016). Na Figura 5 é apresentada uma interface de desenvolvimento do *Android Studio.*

Figura 5 – Tela de desenvolvimento do Android Studio



Fonte: Elaborada pelo Autor

De acordo com o guia de desenvolvimento disponibilizado pelos próprios desenvolvedores da IDE, O *Android Studio* é o ambiente de desenvolvimento integrado oficial para o desenvolvimento de apps Android e é baseado no *IntelliJ IDEA*. Além do editor de código e das ferramentas de desenvolvedor avançadas do próprio *IntelliJ*, o *Android Studio* oferece ainda mais recursos para aumentar sua produtividade na criação de apps Android, tais como:

- Um sistema de compilação flexível baseado em *Gradle*;
- Um emulador rápido com inúmeros recursos;
- Um ambiente unificado que possibilita o desenvolvimento para todos os dispositivos Android;
- A aplicação de mudanças para enviar mudanças de código e recursos ao aplicativo em execução sem reiniciar o app;
- Modelos de código e integração com GitHub para ajudar a criar recursos comuns de apps e importar exemplos de código;
- Frameworks e ferramentas de teste cheios de possibilidades;

- Ferramentas de *lint* para detectar problemas de desempenho, usabilidade, compatibilidade com versões, entre outros;
- Compatibilidade com C++ e NDK (Kit de Desenvolvimento Nativo);
- Compatibilidade integrada com o Google Cloud Platform, facilitando a integração do *Google Cloud Messaging* e do *App Engine*.

Este capítulo apresentou o estado da arte, mostrando as ferramentas que estão disponíveis nos mercados e foram foco de estudo para realizar as comparações e escolher as melhores ferramentas para serem usadas no presente trabalho.

3 DESENVOLVIMENTO DA SOLUÇÃO

Este capítulo apresenta o desenvolvimento da solução que foi implementada na realização desse Trabalho de Conclusão de Curso II, de forma a expor as ferramentas que foram utilizadas, além de conceitos tratados anteriormente para a construção do aplicativo.

3.1 Implementações das permissões

Para se fazer o uso da *geofence* é solicitado ao usuário na primeira vez que ele entra no sistema a permissão para que o aplicativo acesse a sua localização, possibilitando-o executar suas funcionalidades, como pegar a localização exata do usuário.

O aplicativo irá solicitar permissão para ter acesso à localização aproximada e à localização precisa do usuário, sendo respectivamente "*ACCESS_FINE_LOCATION*" e "*ACCESS_COARSE_LOCATION*". Para as permissões de execução ou especial que o aplicativo irá precisar, a partir do *Android* 6.0, elas devem ser inseridas dentro do arquivo "*AndroidManifest.xml*", localizado na pasta raiz de origem do projeto, como mostra a Figura 6 abaixo. (DEVELOPERS, 2023).

Figura 6 – Print de um trecho do código AndroidManifest.xml

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

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

Fonte: Elaborada pelo autor

3.2 Inicialização da aplicação

Ao inicializar o aplicativo, o arquivo *"MainActivity.java*" é executado como inicializador para criar os *fragments*, telas do sistema. Nesse processo de criação de telas, é feita uma verificação se o usuário tem as permissões de localização aceita. A

verificação é feita através da função “*checkSelfPermission*”, que faz a verificação de permissões.

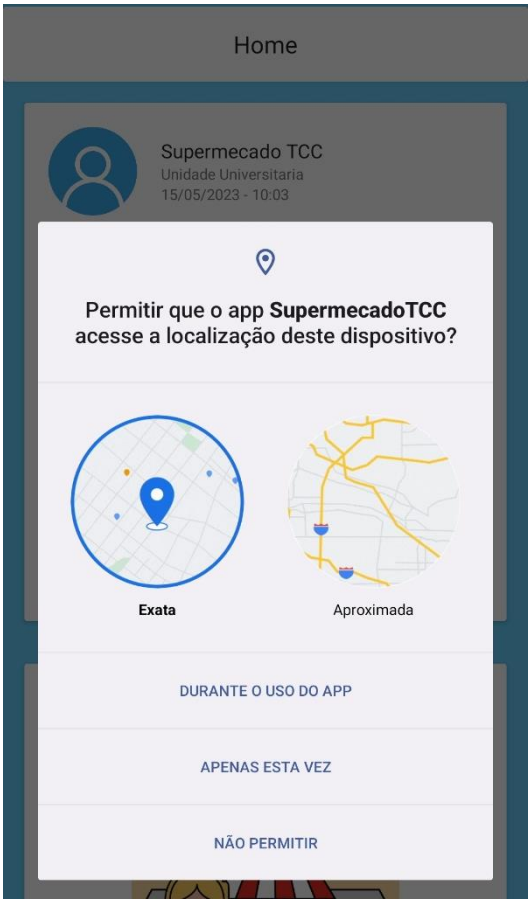
Se o usuário não possui a permissão, é pedido para ele aceitar para prosseguir com o aplicativo, como é mostrado na Figura 7 via código e Figura 8 na interface.

Figura 7 – Print de um trecho do código de verificação de permissão

```
// Verifique as permissões de localização
if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    // Se as permissões não foram concedidas, solicite-as ao usuário
    ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_PERMISSION_REQUEST_CODE);
} else {
    // As permissões já foram concedidas, inicie o serviço de localização
    startLocationService();
}
```

Fonte: Elaborada pelo autor

Figura 8 – Print da tela de requisição de permissão



Fonte: Elaborada pelo autor

3.3 Telas da aplicação e suas regras

O aplicativo conta com 5 telas, que foram feitas utilizando os *fragments*. Um *Fragment* representa um comportamento ou uma parte da interface do usuário em uma *FragmentActivity*. Ele pode ser considerado como uma seção modular de uma atividade, possuindo seu próprio ciclo de vida e recebendo eventos de entrada independentes. Além disso, os *fragments* podem ser adicionados ou removidos dinamicamente durante a execução da atividade, oferecendo flexibilidade na construção da interface. (DEVELOPERS, 2023).

A navegação entre as telas é feita através do "*Navigation Component*" e a "*bottom navigation*".

O *Navigation Component* é um componente do *Android Jetpack* que simplifica a implementação da navegação em um aplicativo Android. Ele fornece uma estrutura robusta para criar fluxos de navegação entre os destinos do aplicativo. O Navigation Component ainda faz a simplificação da implementação na navegação, promove uma arquitetura mais modular e melhora a legibilidade e manutenibilidade do código do aplicativo. (DEVELOPERS, 2023).

A *Bottom Navigation* é um componente de interface do usuário do Android que fornece uma barra de navegação na parte inferior da tela, como é mostrado na Figura 9 a barra de navegação do aplicativo. (DEVELOPERS, 2023).

Figura 9 – Print do componente *Bottom Navigation*

Fonte: Elaborada pelo autor

Ao serem combinados, o *Navigation Component* e a *Bottom Navigation,* fica mais fácil a implementação da navegação entre telas de em um aplicativo Android, e fornece uma estrutura clara e consistente ao criar o aplicativo.

3.3.1 Início

A tela "Inicio" foi desenvolvida para ser uma tela de avisos, mensagem, dentre outras informações. Como por exemplo, dia de promoção, algum evento.

A tela é composta pelo título Home, seguido de postagem feitas pelo estabelecimento. Na figura 10 é possível verificar a aparência dela.

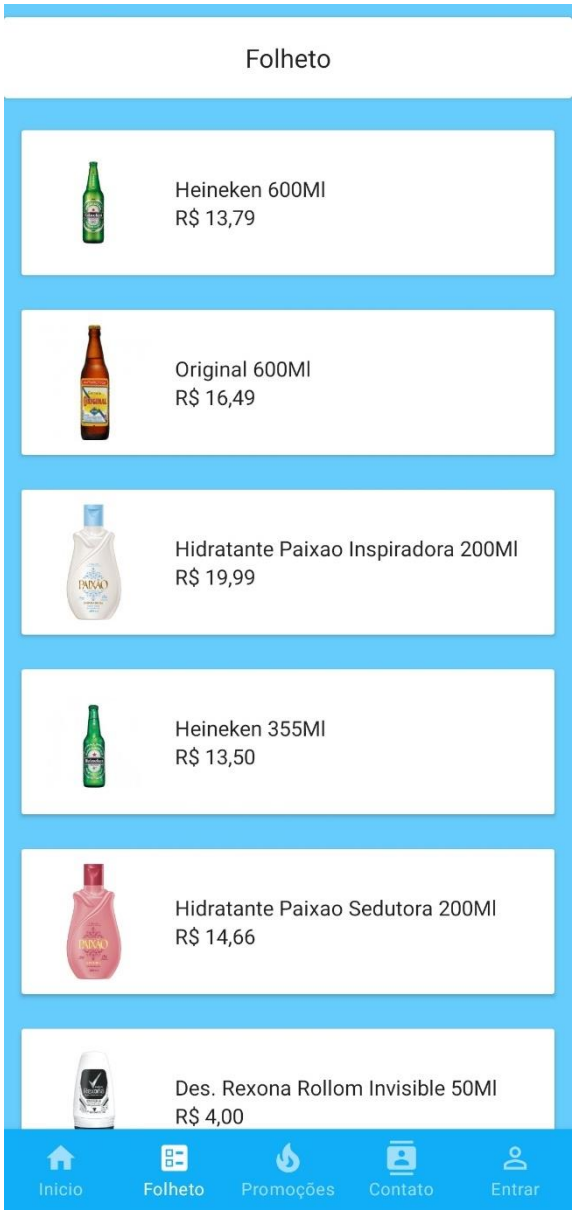
Figura 10 – Print da tela Inicio (home)

Fonte: Elaborada pelo autor

3.3.2 Folheto

A tela Folheto foi desenvolvida para apresentar ao usuário as promoções do estabelecimento, nela apresenta a imagem, preço e descrição do item em promoção, como mostrado na Figura 11.

Figura 11 – Print da tela Folheto



Fonte: Elaborada pelo autor

3.3.3 Promoções

A tela Promoções é onde está concentrado a *geofence*. Nela há dois casos de exibição, quando o usuário está na cerca virtual e quando ele está fora. Para o caso do usuário está fora dessa cerca virtual a tela irá aparecer a mensagem “Não há promoções, você está longe do supermercado”, como mostra a Figura 12.

Figura 12 – Print da tela Promoções com usuário fora da cerca virtual

Fonte: Elaborada pelo autor

Utilizando as coordenadas geográficas de latitude “-16.622617” e longitude “-49.290399”, de um local em Goiânia utilizado como a sede do supermercado, como o centro da cerca virtual, e delimitando a cerca com um raio de 5 metros, é feito a comparação para verificar se o cliente está na cerca virtual. Usando a classe “*onLocationChanged*” para pegar a localização do usuário e definir se ele está ou não na cerca virtual, como mostra a Figura 13.

Figura 13 – Print da implementação da classe *onLocationChanged*

```
public void onLocationChanged(Location location) {
    // Verificar se o usuário está dentro da cerca virtual
    float distance = location.distanceTo(getFenceCenterLocation());
    if (distance <= fenceRadius) {// O usuário está dentro da cerca virtual
        //Inicialização da textview
        TextView textView = getView().findViewById(R.id.textViewCerca);
        //Marca a textView como Invisivel para exibir somente a lista de produtos
        textView.setVisibility(View.GONE);
        //Busca dados no banco para exibir na tela
        buscaDados();
        //Para de buscar os dados no banco
        onPause();
    }
    else{ // O usuário não está dentro da cerca virtual
        RecyclerView recyclerView = getView().findViewById(R.id.recyclerView_Promocoes); //Inicializa a lista
        recyclerView.setVisibility(View.GONE); //Seta como invisivel a lista para nao ser exibida
        TextView textView = getView().findViewById(R.id.textViewCerca); //Inicializa ao cardview
        textView.setText("Não há promoções, você esta longe do supermecado"); //Seta o texto no cardview
    }

}
```

Fonte: Elaborada pelo autor

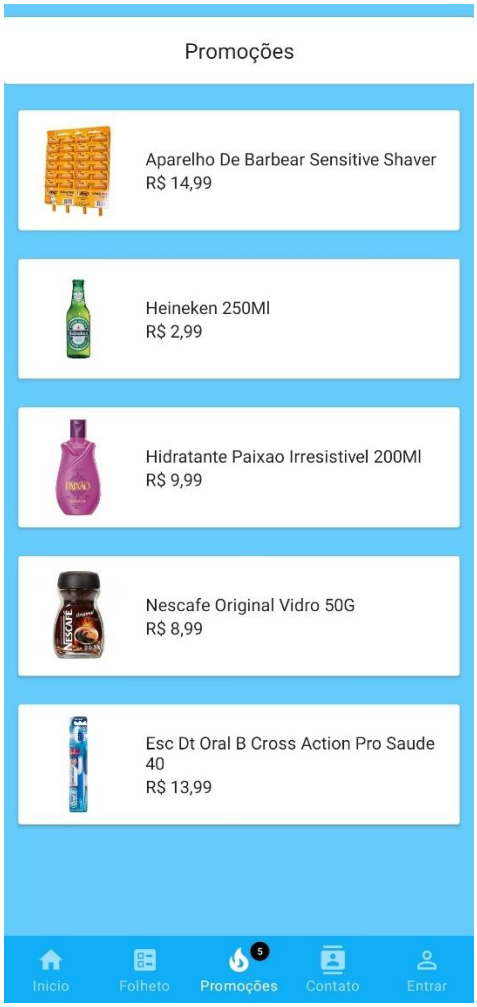
Se o usuário estiver dentro da cerca virtual, ele irá receber uma notificação avisando das promoções exclusivas, como é mostrado na Figura 14.

Figura 14 – Print da notificação

Fonte: Elaborada pelo autor

Além disso, ao clicar na notificação ou abrir o aplicativo na tela de promoções, irá carregar ofertas exclusivas do supermercado, como mostra a Figura 15.

Figura 15 – Print da tela promoções com usuário na cerca virtual



Fonte: Elaborada pelo autor

3.3.4 Contato

A tela Contato foi desenvolvida para que o usuário, através do aplicativo, saiba as informações do supermercado, como por exemplo, número de contato, horário de funcionamento e endereço. Na figura 16 é apresentado a tela contato com os detalhes do supermercado.

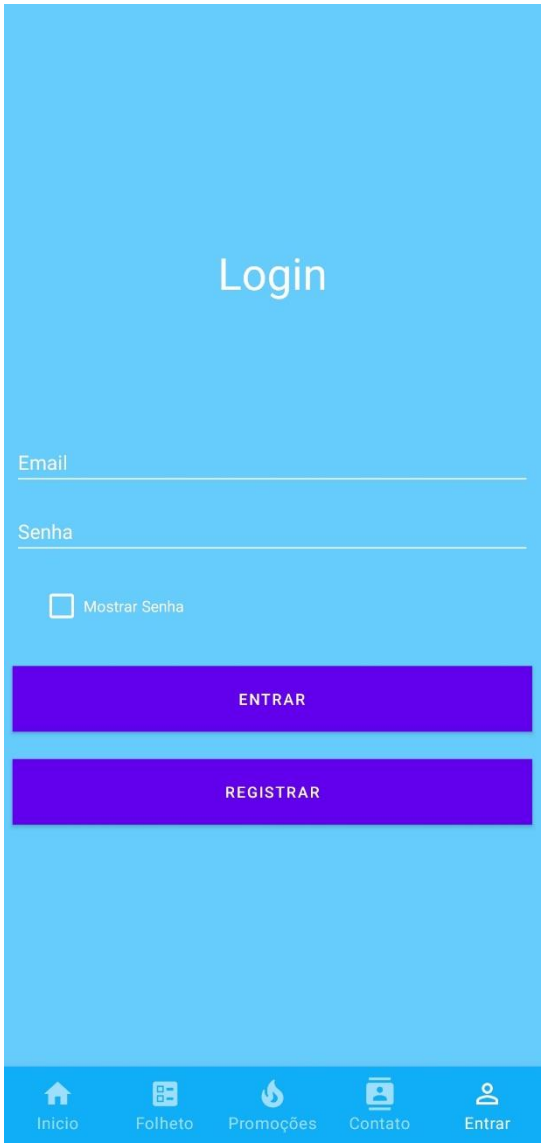
Figura 16 – Print da tela Contato

Fonte: Elaborada pelo autor

3.3.5 Entrar

A tela Entrar foi desenvolvida para que o usuário cliente e estabelecimento, através do aplicativo, possa acessar algumas informações mais sigilosas, como no caso do usuário para ver dados e estabelecimento informar promoções, posts e dentre outros. Na Figura 17 é possível ver a tela inicial de entrar, onde pede e-mail e senha.

Figura 17 – Print da tela Entrar



Fonte: Elaborada pelo autor

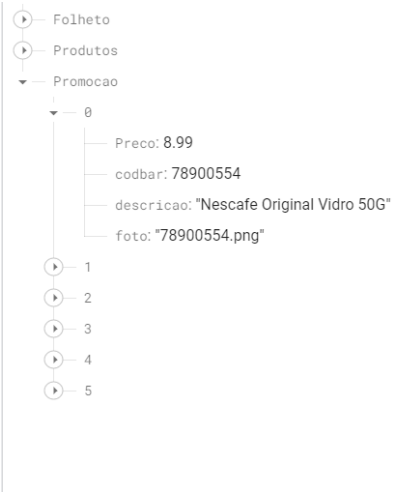
3.4 Banco de dados

Neste aplicativo foi utilizado o banco de dados da *Google*, o *FireBase*. Na sua implementação de *RealTime DataBase* para armazenar a base de dados escritas, como a descrição e preços dos produtos. Para as imagens dos produtos, foi utilizado o *Storage* para armazenar as fotos.

Como o banco de dados *FireBase* é no formato não-relacional, os dados são salvos como uma árvore. No Banco foram divididos em 3 nós principais, sendo eles o “Folheto”, onde é indicado quais produtos estão na promoção do estabelecimento, os “Produtos” que consta todos os produtos que o supermercado possui, para esse

exemplo foi cadastrado no banco apenas 500 produtos. E o nó “Promocao”, que ficam as ofertas exclusivas da *Geofence.* Dentro dos nós principais, tem os nós secundários que por padrão ele enumerou começando no 0, e dentro desses nós secundários é possível ver as informações de cada produto individualmente, como é mostrado na Figura 18.

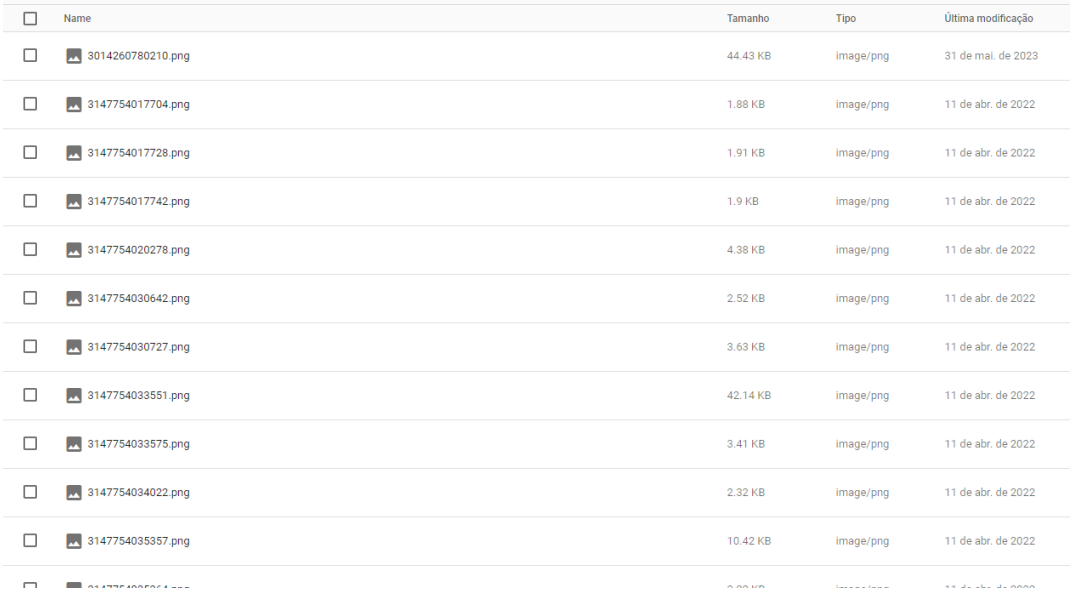
Figura 18 – Print do banco de dados *RealTime Firebase*



Fonte: Elaborada pelo autor

O nó “foto” dentro do *RealTime* é um campo com o nome da foto que está armazenada no *Storage*, da própria *google*. O *FireBase Storage* permite armazenar e servir arquivos, como imagens, vídeos, áudios e outros recursos estáticos, na nuvem. Na Figura 19 é possível ver as imagens armazenadas dentro do *FireBase Storage*.

Figura 19 – Print do banco de dados *FireBase Storage*



Name	Tamanho	Tipo	Última modificação
3014260780210.png	44.43 KB	image/png	31 de mai. de 2023
3147754017704.png	1.88 KB	image/png	11 de abr. de 2022
3147754017728.png	1.91 KB	image/png	11 de abr. de 2022
3147754017742.png	1.9 KB	image/png	11 de abr. de 2022
3147754020278.png	4.38 KB	image/png	11 de abr. de 2022
3147754030642.png	2.52 KB	image/png	11 de abr. de 2022
3147754030727.png	3.63 KB	image/png	11 de abr. de 2022
3147754033551.png	42.14 KB	image/png	11 de abr. de 2022
3147754033575.png	3.41 KB	image/png	11 de abr. de 2022
3147754034022.png	2.32 KB	image/png	11 de abr. de 2022
3147754035357.png	10.42 KB	image/png	11 de abr. de 2022

Fonte: Elaborada pelo autor

E para acessar esses dados pelo aplicativo é necessário fazer a vinculação da api do google com o aplicativo, através do arquivo na Figura 20 mostra um trecho do código de como é feito a leitura no banco *FireBase RealTime* e no *Storage.*

Figura 20 – Print de um trecho do código para leitura no banco de dados

```
private void buscaDados() {
    // Obtém uma referência para o nó "Promocao" no banco de dados
    DatabaseReference reference = database.getReference().child("Promocao");

    // Limita a lista para os primeiros 5 registros e adiciona um ouvinte de valor único
    reference.limitToFirst(5).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            // Cria uma lista para armazenar os objetos FolhetoItem
            List<FolhetoItem> PromocaoItems = new ArrayList<>();

            // Itera sobre os filhos do snapshot (registros obtidos do banco de dados)
            for (DataSnapshot data : snapshot.getChildren()) {
                // Extrai os valores da descrição, preço e caminho da foto usando getValue()
                String descricao = data.child("descricao").getValue(String.class);
                Double preco = data.child("Preco").getValue(Double.class);
                String fotoPath = data.child("foto").getValue(String.class);

                // Cria uma referência de armazenamento para o caminho da foto
                StorageReference imageRef = FirebaseStorage.getInstance().getReference().child(fotoPath);

                // Obtém a URL de download da imagem e trata o sucesso do download
                imageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        // Cria um objeto FolhetoItem com os valores obtidos e a URL da imagem
                        FolhetoItem promocaoItem = new FolhetoItem(descricao, uri, preco);

                        // Adiciona o objeto FolhetoItem à lista PromocaoItems
                        PromocaoItems.add(promocaoItem);

                        // Verifica se todos os itens foram processados e chama setupRecyclerView()
                        if (PromocaoItems.size() == snapshot.getChildrenCount()) {
                            setupRecyclerView(PromocaoItems);
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        // Lidar com falha no download da imagem
                    }
                });
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            // Lidar com erro de cancelamento da operação
        }
    });
}
```

Fonte: Elaborada pelo autor

A leitura é feita e colocada em uma lista com a imagem, preço e descrição. Com a lista preenchida é chamada a função “setupRecyclewView” para fazer a inicialização da tela, e preencher a tela com os dados da lista. Na Figura 21 mostra um trecho do código da função “setupRecyclewView”.

Figura 21 – Print trecho do código da função “setupRecyclewView”.

```
private void setupRecyclerView(List<FolhetoItem> promocaoItems) {
    RecyclerView recyclerView = getView().findViewById(R.id.recyclerView_Promocoes);
    RecycleView_Folheto adapter = new RecycleView_Folheto(promocaoItems);
    recyclerView.setLayoutManager(new LinearLayoutManager(requireContext()));
    recyclerView.setAdapter(adapter);
}
```

Fonte: Elaborada pelo autor

A ordem de exibição na tela depende do fato tempo de download das imagens pelo Storage as imagens, devido ao volume de dados que é puxado ao mesmo tempo. E a cada execução de tela é feito um novo download de informações do banco.

4 CONSIDERAÇÕES FINAIS E PERSPECTIVAS FUTURAS

O trabalho desenvolvido teve por finalidade o desenvolvimento de um aplicativo voltado para a rede de supermercados utilizando a *geofence*, no qual propõe a substituição do panfleto de ofertas.

Durante o processo de desenvolvimento do aplicativo e aprendizado do banco de dados da Google, fica evidente a facilidade de trabalhar com o *FireBase* e principalmente as funcionalidades que ele apresenta em sua versão gratuita. O aplicativo foi desenvolvido utilizando o *RealTime Database*, que na prática não é possível encontrar diferença para o banco de dados convencional, onde todas funções estão presentes e simples de usar.

O Android Studio é o ambiente oficial de desenvolvimento de aplicativos Android. E ele é muito completo tanto para códigos quanto integrações, utilizando-o juntamente com o Google *FireBase* as integrações de banco de dados foram feitas com apenas um botão. Deixando muito fácil trabalhar nele e implementar o aplicativo. O Android Studio apresenta funcionalidades de testes com logs de erro muito bem desenvolvida e documentações na internet de forma fácil.

A linguagem Java e xml do Android utilizadas nesse projeto foram bem tranquilas, com documentações fáceis de achar e principalmente a estrutura da programação orientada a objeto do Java se assemelhando bastante com a estrutura de programação de outras linguagens, como por exemplo o C#. A implementação do Java para o *backend* e o xml para o *frontend* foi pensada pelo fato de ser simples a implementação e integração com o Android Studio.

Com o desenvolvimento deste trabalho foi adquirido um vasto conhecimento na linguagem Java e xml para Android, e sobre os tipos de bancos não-relacional utilizando o *FireBase* para toda essa parte de implementação. O projeto acima apresenta um grande potencial para ser continuado adaptando as funcionalidades de segurança, acompanhamento real do cliente dentro do aplicativo e principalmente no histórico de quais corredores o cliente mais anda, quais são os produtos que mais vendem e o que menos vendem. Os objetivos propostos inicialmente para esse projeto foram todos cumpridos.

Encontra se em anexo 1 termo de autorização de publicação de produção acadêmica na página 55, e a partir do apêndice A, pagina 56, o código comentado do programa desenvolvido separado por arquivo.

4.1 Perspectiva para trabalhos futuros

Propõe-se como perspectiva para trabalhos futuros:

- Implementar a *Geonfece* de forma mais ampla e exata para utilizar o aplicativo como GPS dentro do supermercado
- Implementar um sistema de reconhecimentos de produtos dentro do aplicativo através da câmera e posição do usuário
- Implementar para restaurante, como a precisão melhorada o cliente chegar e receber cardápio e ir acompanhando os gastos feitos na noite

5.REFERÊNCIAS

FAGGIAN, Hugo César. **Geometria e GPS**. 2019. 60 f. Dissertação (Mestrado) - Curso de Matemática em Rede Nacional, Instituto de Ciências Matemáticas e de Computação ICMCUSP, Universidade de São Paulo (USP), São Carlos, 2019. Disponível em: <https://www.teses.usp.br/teses/disponiveis/55/55136/tde-17092019-150542/publico/HugoCesarFraggian_revisada.pdf>. Acesso em: 09 set. 2022.

BRANDÃO, Bruna. **Como funciona a Geolocalização na prática?** Para que serve? Maplink, 2019. Disponível em: <https://maplink.global/blog/como-funciona-geolocalizacao/>. Acesso em: 09 set. 2022.

SANTANA, John Kennedy Ribeiro de; FARIAS, Paulo Lucas Cândido de; XAVIER, Joaquim Pedro de Santana; FIGUEIREDO, Victor Pina. **Precisão de GPS em smartphones**: Uma ferramenta para pesquisas acadêmicas e trabalhos de campo. Revista de Geografia – PPGEO - UFJF, [S.L.], v. 9, n. 2, p. 255-267, abr. 2023.

MICHAHELLES, F.; WATZDORF. **Accuracy of positioning data on smartphones**. LocWeb '10: Proceedings of the 3rd International Workshop on Location and the Web p. 1-4, 2010. Disponível em: <https://dl.acm.org/doi/10.1145/1899662.1899664 >. Acesso: em 23 set. 2022.

LÁZARO, Anselmo. **Coordenadas geográficas** - Latitude, Longitude e GPS. 14 maio 2015. Disponível em: <https://educacao.uol.com.br/disciplinas/geografia/ coordenadas-geograficas-latitude-longitude-e-gps.htm>. Acesso em: 03 out. 2022.

WHITE, Sarah K. **What is geofencing?** Putting location to work. CIO Magazine, IDG Communications, Inc. 01 nov. 2017. Disponível em: <https://www.cio.com/article/ 2383123/geofencing-explained.html>. Acesso em: 10 out. 2022.

PENA, Rodolfo F. Alves**. Latitudes e Longitudes**. 19 maio 2020. Disponível em: <https://mundoeducacao.uol.com.br/geografia/latitudes-longitudes.htm>. Acesso em: 10 out. 2022.

ROVEDA, Ugo**. Banco de dados:** o que é, para que serve, tipos e como criar. 01 fev. 2021. Disponível em: < https://kenzie.com.br/blog/banco-de-dados/> Acesso em: 15 out. 2022.

ALECRIM, Emerso. **Bancos de dados são mais importantes nas nossas vidas do que a gente imagina**. Julho 2018. Disponível em: <https://tecnoblog.net/245120/banco-de-dados-importancia/> Acesso em: 15 out. 2022.

TECHTARGET. **Banco de dados relacional**. Julho 2021. Disponível em: <https://www.computerweekly.com/br/quiz/Banco-de-dados-relacional> Acesso em 15 out. 2022.

TEJADA, Zoier. **Dados não relacionais e NoSQL**. 11 agosto 2021. Disponível em: <https://docs.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data> Acesso em 17 out. 2022.

DEVELOPERS. **Conheça o Android Studio**. 2021. Disponível em: <https://developer.android.com/studio/intro?hl=pt-br>. Acesso em: 20 out 2021.

CIO/EUA. **3 coisas que você precisa saber sobre geofencing**. 02 setembro 2018. Disponível em: < https://cio.com.br/tendencias/3-coisas-que-voce-precisa-saber-sobre-geofencing/> Acesso em: 23 fev. 2023.

BRANDÃO, Bruna. **O que é geofencing?** Para que serve? Como funciona na prática? 18 março 2020. Disponível em: < https://maplink.global/blog/o-que-e-geofencing/> Acesso em 23 fev. 2023.

REDAÇÃO IMPACTA. **Por que devo usar um banco de dados?** 12 dezembro 2016. Disponível em: < https://www.impacta.com.br/blog/por-que-devo-usar-um-banco-de-dados/> Acesso em: 23 fev. 2023.

GUEDES, Marylene. **O que é MongoDB?** Setembro 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mongodb> Acesso em: 30 maio 2023.

MEDEIROS, Higor. **Introdução ao MongoDB**. 2014. Disponível em: <https://www.devmedia.com.br/introducao-ao-mongodb/30792> Acesso em: 03 nov. 2021.

CONTENT, Rock**. Conheça FireBase:** a ferramenta de desenvolvimento e análise de aplicativos mobile. 21 agosto 2019. Disponível em: <https://rockcontent.com/br/blog/firebase/> Acesso em: 03 mar. 2023.

SILVA, Eduardo. **FireBase:** o que é e quando usar no desenvolvimento mobile? 06 maio 2021. Disponível em: <https://blog.geekhunter.com.br/firebase-o-que-e-e-quando-usar-no-desenvolvimento-mobile/> Acesso em 03 mar. 2023.

SINGH, Vrijraj. ***Introduction to Firebase***. 12 dezembro 2018. Disponível em: <https://medium.com/codingurukul/introduction-to-firebase-f9f6ccc8a785> Acesso em: 05 abr. 2023.

GIOVANINI, Adenilson. **Latitude e longitude**: o que são e como calcular! 2014. Disponível em: <https://adenilsongiovanini.com.br/blog/latitude-e-longitude-o-que-sao-e-como-calcular/> Acesso em: 19 abr. 2023.

CORDEIRO, Allan Rangel. **Localização geográfica através de aparelho celular**. 2011. 9f. TCC (Graduação) - Universidade Federal do Paraná. Disponível em: <https://www.eletrica.ufpr.br/ufpr2/tccs/39.pdf> Acesso em: 04 abr. 2023.

TROIA, Pedro. **Como funciona a localização através de Wi-fi**. Janeiro 2021. Disponível em: <https://www.pcguia.pt/2021/01/como-funciona-a-localizacao-atraves-de-wi-fi/> Acesso em: 04 jun. 2023.

TIAGO. **Android Studio**: O Que É E Como Desenvolver Apps Nele. 20 julho 2020. Disponível em: <https://mundodevops.com/blog/android-studio/> Acesso em: 04 jun. 2023.

REMESSA, Online. **FireBase**: descubra para que serve, como funciona e como usar. 01 nov. 2021. Disponível em:<https://www.remessaonline.com.br/blog/firebase-descubra-para-que-serve-como-funciona-e-como-usar/> Acesso em 04 jun. 2023.

Anexo I

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 nº 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Ruan Lobo P. do Prado
do Curso de Engenharia da Computação, matrícula 2017.2.0033.0066-0,
telefone: (62) 9 9522 7789 e-mail rpguan@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 Codificação aplicada na rede de Supervisão no datapoint mobile, 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, 28 de janeiro de 2023.

Assinatura do autor: Ruan Lobo P. do Prado

Nome completo do autor: Ruan Lobo Pereira do Prado

Assinatura do professor–orientador:

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

Apêndices

Apêndice A – AndroidManifest.xml

```
<!-- O código abaixo é um arquivo de manifesto XML do Android, que descreve as configurações e permissões do aplicativo-->
<!-- Declara as permissões necessárias para acessar a localização do dispositivo -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
<uses-permission android:name="android.permission.VIBRATE" />

<!-- Declaração da aplicação -->
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.SupermecadoTCC"
    tools:targetApi="31">

    <!-- Declara a atividade principal do aplicativo -->
    <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>
</application>
```

Apêndice B – contato.java

```
package com.example.supermecadotcc.fragments;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.example.supermecadotcc.R;

public class contato extends Fragment {
    // Parâmetros de inicialização do fragmento
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // Tipos dos parâmetros
    private String mParam1;
    private String mParam2;

    public contato() {
        // Construtor público vazio necessário
    }

    /**
     * Criar uma nova instância do fragmento usando os parâmetros fornecidos.
     * @param param1 Parâmetro 1.
     * @param param2 Parâmetro 2.
     * @return Uma nova instância do fragmento contato.
     */
    public static contato newInstance(String param1, String param2) {
        contato fragment = new contato();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Infla o layout para este fragmento
        return inflater.inflate(R.layout.fragment_contato, container, false);
    }
}
```

Apêndice C – folheto.java

```
package com.example.supermecadotcc.fragments;
import android.net.Uri;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.example.supermecadotcc.FolhetoItem;
import com.example.supermecadotcc.RecycleView_Folheto;
import com.example.supermecadotcc.R;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
```

```
import com.google.firebase.storage.StorageReference;
import java.util.ArrayList;
import java.util.List;

public class folheto extends Fragment {
    // Parâmetros de inicialização do fragmento
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
    // Tipos dos parâmetros
    private String mParam1;
    private String mParam2;

    private FirebaseDatabase database;
    private FirebaseStorage storage;
    private ValueEventListener valueEventListener;

    public folheto() {
        // Construtor público vazio necessário
    }

    /**
     * Criar uma nova instância do fragmento usando os parâmetros fornecidos.
     * @param param1 Parâmetro 1.
     * @param param2 Parâmetro 2.
     * @return Uma nova instância do fragmento contato.
     */
    public static folheto newInstance(String param1, String param2) {
        folheto fragment = new folheto();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
```

```
        mParam2 = getArguments().getString(ARG_PARAM2);
    }

    // Inicializa o banco de dados Firebase
    database = FirebaseDatabase.getInstance();
    storage = FirebaseStorage.getInstance();

    // Define o ouvinte para recuperar os dados do Firebase
    ouvinte();
}

private void ouvinte() {
    DatabaseReference reference = database.getReference().child("Folheto");
    reference.limitToFirst(20).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            List<FolhetoItem> folhetoItems = new ArrayList<>();

            for (DataSnapshot data : snapshot.getChildren()) {
                String descricao = data.child("descricao").getValue(String.class);
                Double preco = data.child("Preco").getValue(Double.class);
                String fotoPath = data.child("foto").getValue(String.class);

                StorageReference imageRef = storage.getReference().child(fotoPath);
                imageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        FolhetoItem folhetoItem = new FolhetoItem(descricao, uri, preco);
                        folhetoItems.add(folhetoItem);

                        if (folhetoItems.size() == snapshot.getChildrenCount()) {
                            setupRecyclerView(folhetoItems);
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        // Lida com a falha no download da imagem
                    }
                });
```

```
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            // Lida com o cancelamento do evento de banco de dados
        }
    });
}

private void setupRecyclerView(List<FolhetoItem> folhetoItems) {
    RecyclerView recyclerView = getView().findViewById(R.id.recyclerView);
    RecycleView_Folheto adapter = new RecycleView_Folheto(folhetoItems);
    recyclerView.setLayoutManager(new LinearLayoutManager(requireContext()));
    recyclerView.setAdapter(adapter);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Infla o layout para este fragmento
    return inflater.inflate(R.layout.fragment_folheto, container, false);
}
}
```

Apêndice D – home.java

```
package com.example.supermecadotcc.fragments;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.example.supermecadotcc.R;

public class home extends Fragment {

    // Parâmetros de inicialização do fragmento
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
```

```
// Tipos dos parâmetros
private String mParam1;
private String mParam2;

public home() {
    // Construtor público vazio necessário
}

/**
 * Criar uma nova instância do fragmento usando os parâmetros fornecidos.
 * @param param1 Parâmetro 1.
 * @param param2 Parâmetro 2.
 * @return Uma nova instância do fragmento contato.
 */
public static home newInstance(String param1, String param2) {
    home fragment = new home();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Verifica se há argumentos passados para o fragmento
    if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Infla o layout para este fragmento
    return inflater.inflate(R.layout.fragment_home2, container, false);
```

```
    }
}
```

Apêndice E – login.java

```
package com.example.supermecadotcc.fragments;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.example.supermecadotcc.R;

public class login extends Fragment {

    // Parâmetros de inicialização do fragmento
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // Tipos dos parâmetros
    private String mParam1;
    private String mParam2;

    public login() {
        // Construtor público vazio necessário
    }

    /**
     * Criar uma nova instância do fragmento usando os parâmetros fornecidos.
     * @param param1 Parâmetro 1.
     * @param param2 Parâmetro 2.
     * @return Uma nova instância do fragmento contato.
     */
    public static login newInstance(String param1, String param2) {
        login fragment = new login();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
```

```
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Verifica se há argumentos passados para o fragmento
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Infla o layout para este fragmento
        return inflater.inflate(R.layout.fragment_login, container, false);
    }
}
```

Apêndice F – promoções.java

```
package com.example.supermecadotcc.fragments;
import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.example.supermecadotcc.FolhetoItem;
import com.example.supermecadotcc.R;
import com.example.supermecadotcc.RecycleView_Folheto;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import java.util.ArrayList;
import java.util.List;

public class promocoes extends Fragment implements LocationListener{

    //Parâmetros de inicialização do fragmento
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // Tipos dos parâmetros
    private String mParam1;
    private String mParam2;


    private FirebaseDatabase database;
    private FirebaseStorage storage;
    private ValueEventListener valueEventListener;
    private LocationManager locationManager;
    private double fenceLatitude = -16.622617;
    private double fenceLongitude = -49.290399;
    private float fenceRadius = 5; // raio em metros da cerca virtual
    private static final int REQUEST_LOCATION_PERMISSION = 123;
```

```
public promocoes() {
    // Construtor público vazio necessário
}

/**
 * Criar uma nova instância do fragmento usando os parâmetros fornecidos.
 * @param param1 Parâmetro 1.
 * @param param2 Parâmetro 2.
 * @return Uma nova instância do fragmento contato.
 */
public static promocoes newInstance(String param1, String param2) {
    promocoes fragment = new promocoes();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
    }
    database = FirebaseDatabase.getInstance();
    FirebaseStorage storage = FirebaseStorage.getInstance();
    StorageReference storageRef = storage.getReference();
}

private void buscaDados() {
    DatabaseReference reference = database.getReference().child("Promocao");
    //Limito a lista para evitar lentidões e crio uma lista
```

```
        reference.limitToFirst(5).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                List<FolhetoItem> PromocaoItems = new ArrayList<>();

                for (DataSnapshot data : snapshot.getChildren()) {
                    String descricao = data.child("descricao").getValue(String.class);
                    Double preco = data.child("Preco").getValue(Double.class);
                    String fotoPath = data.child("foto").getValue(String.class);

                    StorageReference imageRef =
FirebaseStorage.getInstance().getReference().child(fotoPath);
                    imageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                        @Override
                        public void onSuccess(Uri uri) {
                            FolhetoItem promocaoItem = new FolhetoItem(descricao, uri, preco);
                            PromocaoItems.add(promocaoItem);

                            if (PromocaoItems.size() == snapshot.getChildrenCount()) {
                                setupRecyclerView(PromocaoItems);
                            }
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            // Lidar com falha no download da imagem
                        }
                    });
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
```

```
        });
    }

    private void setupRecyclerView(List<FolhetoItem> promocaoItems) {
        RecyclerView recyclerView = getView().findViewById(R.id.recyclerView_Promocoes);
        RecycleView_Folheto adapter = new RecycleView_Folheto(promocaoItems);
        recyclerView.setLayoutManager(new LinearLayoutManager(requireContext()));
        recyclerView.setAdapter(adapter);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Infla o layout para este fragmento
        View view = inflater.inflate(R.layout.fragment_promocoes, container, false);
        locationManager = (LocationManager)
requireActivity().getSystemService(Context.LOCATION_SERVICE);

        return view;
    }

    @Override
    public void onResume() {
        super.onResume();
        // Registrar o LocationListener para receber atualizações de localização
        if (ActivityCompat.checkSelfPermission(requireContext(),
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0L, (float) 0,
(LocationListener) this);
        } else {
            // Solicitar permissão de localização, se necessário
            ActivityCompat.requestPermissions(requireActivity(), new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION_PERMISSION);
        }
    }
```

```
    @Override
    public void onPause() {
        super.onPause();
        // Parar de receber atualizações de localização
        locationManager.removeUpdates((LocationListener) this);
    }



    public void onLocationChanged(Location location) {
        // Verificar se o usuário está dentro da cerca virtual
        float distance = location.distanceTo(getFenceCenterLocation());
        if (distance <= fenceRadius) {// O usuário está dentro da cerca virtual
            //Inicialização da textview
            TextView textView = getView().findViewById(R.id.textViewCerca);
            //Marca a textView como Invisivel para exibir somente a lista de produtos
            textView.setVisibility(View.GONE);
            //Busca dados no banco para exibir na tela
            buscaDados();
            //Para de buscar os dados no banco
            onPause();
        }
        else{ // O usuário não está dentro da cerca virtual
            RecyclerView recyclerView = getView().findViewById(R.id.recyclerView_Promocoes);
//Inicializa a lista
            recyclerView.setVisibility(View.GONE); //Seta como invisivel a lista para nao ser exibida
            TextView textView = getView().findViewById(R.id.textViewCerca); //Inicializa ao cardview
            textView.setText("Não há promoções, você esta longe do supermecado"); //Seta o texto no
cardview
        }


    }


    private Location getFenceCenterLocation() {
        Location centerLocation = new Location("center");
        centerLocation.setLatitude(fenceLatitude);
```

```
        centerLocation.setLongitude(fenceLongitude);
        return centerLocation;
    }
}
```

Apêndice G – FolhetoItem.java

```
package com.example.supermecadotcc;
import android.net.Uri;

public class FolhetoItem {
    private String descricao;
    private Uri fotoUri;
    private Double preco;

    public FolhetoItem(String descricao, Uri fotoUri, Double preco) {
        this.descricao = descricao;
        this.fotoUri = fotoUri;
        this.preco = preco;
    }
    public String getDescricao() {
        return descricao;
    }
    public Uri getFotoUri() {
        return fotoUri;
    }
    public Double getPreco() {
        return preco;
    }
}
```

Apêndice H – MainActivity.java

```
package com.example.supermecadotcc;
// Importações das classes necessárias
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import androidx.core.content.ContextCompat;
import androidx.navigation.NavController;
import androidx.navigation.NavDestination;
import androidx.navigation.Navigation;
import androidx.navigation.fragment.NavHostFragment;
import androidx.navigation.ui.NavigationUI;
import android.Manifest;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import com.example.supermecadotcc.databinding.ActivityMainBinding;
import com.google.android.material.badge.BadgeDrawable;

// Declaração da classe MainActivity, que estende AppCompatActivity
public class MainActivity extends AppCompatActivity {
    // Constante para o código de solicitação de permissão de localização
    private static final int LOCATION_PERMISSION_REQUEST_CODE = 1001;
```

```
    // Declaração de variáveis de classe
    private ActivityMainBinding binding;
    private NavHostFragment navHostFragment;
    private NavController navController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Chamada ao método onCreate da classe pai para executar as inicializações necessárias
        super.onCreate(savedInstanceState);

        // Infla o layout da atividade usando a classe ActivityMainBinding
        binding = ActivityMainBinding.inflate(getLayoutInflater());

        // Define o layout da atividade como o layout raiz do objeto binding
        setContentView(binding.getRoot());

        // Verifique as permissões de localização
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
            // Se as permissões não foram concedidas, solicite-as ao usuário
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
LOCATION_PERMISSION_REQUEST_CODE);
        } else {
            // As permissões já foram concedidas, inicie o serviço de localização
            startLocationService();
        }

        // Inicializa a navegação na atividade
        initNavigation();
    }

    // Método para inicializar a navegação na atividade
    private void initNavigation() {
        // Obtém o Fragment do tipo NavHostFragment a partir do gerenciador de fragmentos
```

```
        navHostFragment = (NavHostFragment)
getSupportFragmentManager().findFragmentById(R.id.nav_host_fragment);

        // Obtém o NavController a partir do NavHostFragment
        navController = navHostFragment.getNavController();

        // Configura o NavController para trabalhar em conjunto com a bottom navigation do layout
        NavigationUI.setupWithNavController(binding.bottomNavigation, navController);

        // Adiciona um listener para quando a destinação atual do NavController for alterada
        navController.addOnDestinationChangedListener((navController, destination, bundle) -> {
            // Verifica se a destinação atual é a "menu_promocao"
            if (destination.getId() == R.id.menu_promocao) {
                // Obtém o BadgeDrawable da bottom navigation relacionado à "menu_promocao"
                BadgeDrawable badgeDrawable =
binding.bottomNavigation.getBadge(R.id.menu_promocao);
                // Verifica se o BadgeDrawable não é nulo
                if (badgeDrawable != null) {
                    // Oculta o badgeDrawable
                    badgeDrawable.setVisible(false);
                    // Limpa o número do badgeDrawable
                    badgeDrawable.clearNumber();
                }
            }
        });
    }

    // Método para inicializar o "badge" (insígnia) de promoção no layout
    private void initBadge(int quantidade) {
        BadgeDrawable badge = binding.bottomNavigation.getOrCreateBadge(R.id.menu_promocao);
        badge.setVisible(true);
        badge.setNumber(quantidade);

        badge.setBackgroundColor(ContextCompat.getColor(this, R.color.black));
        badge.setBadgeTextColor(ContextCompat.getColor(this, R.color.white));
```

```
        badge.setVerticalOffset(13);
        badge.setHorizontalOffset(-10);
    }

    // Método para iniciar o serviço de localização
    private void startLocationService() {
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        LocationListener locationListener = new LocationListener() {
            // Implementação dos métodos de LocationListener

            @Override
            public void onLocationChanged(Location location) {
                // Aqui você recebe as atualizações de localização do dispositivo
                Location centerLocation = new Location("center");
                centerLocation.setLatitude(-16.622617); // Latitude central da cerca virtual
                centerLocation.setLongitude(-49.290399); // Longitude central da cerca virtual

                float distance = location.distanceTo(centerLocation); // Distância em metros

                if (distance <= 5) {
                    // A localização está dentro da cerca virtual
                    // Chama função para criar notificação
                    createNotification();
                } else {
                    // A localização está fora da cerca virtual
                }
            }

            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {
            }

            @Override
```

```
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onProviderDisabled(String provider) {
        }
    };

    // Solicite as atualizações de localização
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
locationListener);
  }

  // Constante para o ID da notificação
  private static final int NOTIFICATION_ID = 1;
  // Constante para o ID do canal de notificação
  private static final String CHANNEL_ID = "location_notifications";

  // Método para criar e exibir a notificação
  private void createNotification() {
    // Construir a notificação
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.login)
        .setContentTitle("Promoção Exclusiva")
        .setContentText("Entre no aplicativo para ver as promoções exclusiva")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    // Criação do canal de notificação (apenas para Android 8.0 e versões superiores)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence channelName = "Canal de Notificação";
```

```
            int importance = NotificationManager.IMPORTANCE_DEFAULT;
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, channelName,
importance);

            // Registrar o canal de notificação no sistema
            NotificationManager notificationManager = getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);
        }

        // Exibir a notificação
        NotificationManagerCompat notificationManagerCompat =
NotificationManagerCompat.from(this);
        notificationManagerCompat.notify(NOTIFICATION_ID, builder.build());
    }
}
```

Apêndice I - RecycleView_Folheto

```
package com.example.supermecadotcc;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import android.net.Uri;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import com.bumptech.glide.Glide;
import java.text.DecimalFormat;
import java.util.List;

public class RecycleView_Folheto extends RecyclerView.Adapter<RecycleView_Folheto.ViewHolder>
{
    private List<FolhetoItem> folhetoItems;
```

```
    public RecycleView_Folheto(List<FolhetoItem> folhetoItems) {
        this.folhetoItems = folhetoItems;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // Inflar o layout do item do card do folheto
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_card_folheto,
parent, false);
        // Retornar uma instância de ViewHolder contendo a view inflada
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        // Obter o item do folheto na posição especificada
        FolhetoItem item = folhetoItems.get(position);

        // Configurar os dados do item nos componentes da view
        holder.descricaoTextView.setText(item.getDescricao());

        // Formatando o preço com a formatação "R$ #0.00"
        Double dataPreco = item.getPreco();
        DecimalFormat decimalFormat = new DecimalFormat("R$ #0.00");
        String precoFormatado = decimalFormat.format(dataPreco);
        holder.precoTextView.setText(precoFormatado);

        // Carregar a imagem usando Glide a partir da URI fornecida
        Glide.with(holder.itemView.getContext())
                .load(item.getFotoUri())
                .into(holder.fotoImageView);
    }

    @Override
```

```
    public int getItemCount() {
        // Retorna o número de itens na lista
        return folhetoItems.size();
    }

    // Classe ViewHolder que representa os componentes da view do item do card do folheto
    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView descricaoTextView;
        TextView precoTextView;
        ImageView fotoImageView;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            // Inicializar os componentes da view a partir dos IDs
            descricaoTextView = itemView.findViewById(R.id.textViewDescricao);
            precoTextView = itemView.findViewById(R.id.textViewPreco);
            fotoImageView = itemView.findViewById(R.id.fotoImageView);
        }
    }
}
```

Apêndice J - RecycleView_Promocao

```
package com.example.supermecadotcc;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.bumptech.glide.Glide;
import java.text.DecimalFormat;
import java.util.List;
```

```
public class RecycleView_Promocao extends
RecyclerView.Adapter<RecycleView_Promocao.ViewHolder> {
    private List<FolhetoItem> folhetoItems;

    public RecycleView_Promocao(List<FolhetoItem> folhetoItems) {
        this.folhetoItems = folhetoItems;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // Inflar o layout do item do card do folheto
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_card_folheto,
parent, false);
        // Retornar uma instância de ViewHolder contendo a view inflada
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        // Obter o item do folheto na posição especificada
        FolhetoItem item = folhetoItems.get(position);

        // Configurar os dados do item nos componentes da view
        holder.descricaoTextView.setText(item.getDescricao());

        // Formatando o preço com a formatação "R$ #0.00"
        Double dataPreco = item.getPreco();
        DecimalFormat decimalFormat = new DecimalFormat("R$ #0.00");
        String precoFormatado = decimalFormat.format(dataPreco);
        holder.precoTextView.setText(precoFormatado);

        // Carregar a imagem usando Glide a partir da URI fornecida
        Glide.with(holder.itemView.getContext())
```

```
                .load(item.getFotoUri())
                .into(holder.fotoImageView);
    }

    @Override
    public int getItemCount() {
        // Retorna o número de itens na lista
        return folhetoItems.size();
    }

    // Classe ViewHolder que representa os componentes da view do item do card do folheto
    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView descricaoTextView;
        TextView precoTextView;
        ImageView fotoImageView;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            // Inicializar os componentes da view a partir dos IDs
            descricaoTextView = itemView.findViewById(R.id.textViewDescricao);
            precoTextView = itemView.findViewById(R.id.textViewPreco);
            fotoImageView = itemView.findViewById(R.id.fotoImageView);
        }
    }
}
```

Apêndice K – BuildConfig.java

```
/**
 * Automatically generated file. DO NOT MODIFY
 */
package com.example.supermecadotcc;

public final class BuildConfig {
```

```
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String APPLICATION_ID = "com.example.supermecadotcc";
  public static final String BUILD_TYPE = "debug";
  public static final int VERSION_CODE = 1;
  public static final String VERSION_NAME = "1.0";
}
```

Apêndice L – activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Declaração do XML e configuração da codificação do arquivo -->

<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">
    <!-- Abertura do elemento raiz, um ConstraintLayout, que é um layout que permite criar regras de posicionamento dos elementos.
    Também estão definidos os namespaces android, app e tools para uso de atributos específicos desses namespaces -->

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:name="androidx.navigation.fragment.NavHostFragment"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toTopOf="@+id/bottom_navigation"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_main" />
```

```
    <!-- Um FragmentContainerView é usado para hospedar fragmentos dentro do layout. Neste caso, é usado para hospedar o NavHostFragment,
    que é responsável por gerenciar a navegação entre os fragmentos.
    Os atributos definidos controlam o posicionamento e o comportamento do fragmento -->

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_navigation"
        style="@style/Widget.App.BottomNavigationView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#10AEF7"
        app:labelVisibilityMode="labeled"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:menu="@menu/bottom_navigation_menu" />
    <!-- Um BottomNavigationView é um componente de interface do usuário que fornece uma barra de navegação na parte inferior da tela.
    Os atributos definidos controlam o estilo, o tamanho, a aparência e o comportamento do BottomNavigationView -->
</androidx.constraintlayout.widget.ConstraintLayout>
<!-- Fechamento do elemento raiz ConstraintLayout -->
```

Apêndice M – fragmente_contato.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#66CCFB"
    android:orientation="vertical"
    tools:context=".fragments.contato">
    <ScrollView
        android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

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

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="16dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:textAlignment="center"
            android:text="Contatos"
            style="@android:style/TextAppearance.Holo.Large"/>
    </LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp">

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

```
android:orientation="vertical"
android:padding="16dp">


<ImageView
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:src="@drawable/pucgoias"
    android:layout_marginTop="10dp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:text="SUPERMECADO TCC - LOJA 01"
    style="@android:style/TextAppearance.Holo.Large"
    android:padding="10dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="20dp"
    android:text="Av. Universitária 1.440, Setor Universitário"
    style="@android:style/TextAppearance.Holo.Small"
    android:layout_marginLeft="10dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="20dp"
    android:text="CEP: 74605-010"
    style="@android:style/TextAppearance.Holo.Small"
    android:layout_marginLeft="10dp"
    />

<TextView
```

```
            android:layout_width="wrap_content"
            android:layout_height="20dp"
            android:text="Goiânia, Goiás"
            style="@android:style/TextAppearance.Holo.Small"
            android:layout_marginLeft="10dp"
            />
    </LinearLayout>


</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="16dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="Telefones:"
            style="@android:style/TextAppearance.Holo.Large"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="(62) 3210-3210"
            style="@android:style/TextAppearance.Holo.Medium"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
```

```
            android:text="(62) 3210-3211"
            style="@android:style/TextAppearance.Holo.Medium"/>
    </LinearLayout>


</androidx.cardview.widget.CardView>

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical"
                android:padding="16dp">
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:text="Horário de funcionamento"
                    style="@android:style/TextAppearance.Holo.Large"/>
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:text="De Seguna a Sábado"
                    style="@android:style/TextAppearance.Holo.Medium"/>
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:text="Das 08H00 às 20h00"
                    style="@android:style/TextAppearance.Holo.Medium"/>

                <TextView
```

```
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginTop="16dp"
                android:text="Domingo e Feriados"
                style="@android:style/TextAppearance.Holo.Medium"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Das 08H00 às 12h00"
                style="@android:style/TextAppearance.Holo.Medium"/>
        </LinearLayout>


    </androidx.cardview.widget.CardView>

  </LinearLayout>
 </ScrollView>
</LinearLayout>
```

Apêndice N – fragmente_folheto

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#66CCFB"
    android:orientation="vertical"
    tools:context=".fragments.folheto">

    <!-- TODO: Update blank fragment layout -->
    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="10dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:padding="16dp">
            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:textAlignment="center"
                android:text="Folheto"
                style="@android:style/TextAppearance.Holo.Large"/>
        </LinearLayout>
    </androidx.cardview.widget.CardView>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="10dp" />


</LinearLayout>
```

Apêndice O – fragmente_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#66CCFB"
    android:orientation="vertical"
```

```
tools:context=".fragments.home">

<!-- TODO: Update blank fragment layout -->

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

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

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical"
                android:padding="16dp">
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:textAlignment="center"
                    android:text="Home"
                    style="@android:style/TextAppearance.Holo.Large"/>
            </LinearLayout>
        </androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:layout_margin="16dp">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

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

        <ImageView
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:src="@drawable/login" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginStart="16dp"
            android:layout_gravity="center">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Supermecado TCC"
                style="@android:style/TextAppearance.Holo.Medium"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
                android:text="Unidade Universitaria"
                style="@android:style/TextAppearance.Holo.Small"
                />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="15/05/2023 - 10:03"
                style="@android:style/TextAppearance.Holo.Small"
                />
        </LinearLayout>

    </LinearLayout>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:src="@drawable/de55d23a865b4a80daa4afabb0406628"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp"
        android:text="Inauguração do Supermecado TCC no dia 10/06/2023, estamos aguardando
todos vocês."
        style="@android:style/TextAppearance.Holo.Medium"/>
  </LinearLayout>


</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp">

    <LinearLayout
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp">

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

    <ImageView
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:src="@drawable/login" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:orientation="vertical">

        <TextView
            style="@android:style/TextAppearance.Holo.Medium"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Supermecado TCC" />

        <TextView
            style="@android:style/TextAppearance.Holo.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Unidade Universitaria" />
```

```
            <TextView
                style="@android:style/TextAppearance.Holo.Small"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="01/05/2023 - 15:41" />
        </LinearLayout>

    </LinearLayout>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:src="@drawable/ontrucaomercado"/>

    <TextView
        style="@android:style/TextAppearance.Holo.Medium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp"
        android:text="Supermecado em fase final de construção, estamos aguardando vocês." />
  </LinearLayout>
 </androidx.cardview.widget.CardView>
  </LinearLayout>
 </ScrollView>
</LinearLayout>
```

Apêndice P – fragmente_login.xml

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

```
android:gravity="center"
android:background="#66CCFB"
tools:context=".fragments.login">

<!-- TODO: Update blank fragment layout -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Login"
    android:textColor="@color/white"
    android:textSize="40sp"
    android:layout_marginBottom="120sp"
    android:textAlignment="center"
    />

<EditText
    android:id="@+id/edt_email"
    android:layout_width="380dp"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:textColorHint="@color/white"
    android:backgroundTint="@color/white"
    />

<EditText
    android:id="@+id/edt_senha"
    android:layout_width="380dp"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/white"
    android:hint="Senha"
    android:inputType="textPassword"
    android:textColorHint="@color/white"
    android:layout_marginTop="10dp"
```

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

<CheckBox
    android:id="@+id/ckb_mostrar_senha"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginRight="70dp"
    android:buttonTint="@color/white"
    android:text="Mostrar Senha"
    android:textColor="@color/white" />

<Button
    android:layout_width="380dp"
    android:layout_height="wrap_content"
    android:background="@color/white"
    android:textColor="@color/white"
    android:text="Entrar"
    android:layout_marginTop="20dp"
    android:id="@+id/btn_login"
    />

<Button
    android:layout_width="380dp"
    android:layout_height="wrap_content"
    android:background="@color/white"
    android:textColor="@color/white"
    android:text="Registrar"
    android:layout_marginTop="20dp"
    android:id="@+id/btn_registrar"
    />
```

```
</LinearLayout>
```

Apêndice Q – fragmente_promocoes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#66CCFB"
    android:orientation="vertical"
    tools:context=".fragments.promocoes">

    <!-- TODO: Update blank fragment layout -->
    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:padding="16dp">
            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:textAlignment="center"
                android:text="Promoções"
                style="@android:style/TextAppearance.Holo.Large"/>
        </LinearLayout>
    </androidx.cardview.widget.CardView>

    <androidx.recyclerview.widget.RecyclerView
```

```
        android:id="@+id/recyclerView_Promocoes"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="10dp" />

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="200dp"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:padding="16dp">
            <TextView
                android:id="@+id/textViewCerca"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:textAlignment="center"
                style="@android:style/TextAppearance.Holo.Small"/>
        </LinearLayout>
    </androidx.cardview.widget.CardView>


</LinearLayout>
```

Apêndice R – list_item_card_folheto

```
<?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"
android:layout_margin="12dp">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp">
    <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:orientation="horizontal">

            <ImageView
                android:id="@+id/fotoImageView"
                android:layout_width="80dp"
                android:layout_height="80dp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginStart="16dp"
            android:layout_gravity="center">

            <TextView
                android:id="@+id/textViewDescricao"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                style="@android:style/TextAppearance.Holo.Medium"/>
```

```
            <TextView
                android:id="@+id/textViewPreco"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                style="@android:style/TextAppearance.Holo.Medium"/>

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

Apêndice S – build.gradle (project)

```
buildscript {
    dependencies {
        classpath 'com.google.gms:google-services:4.3.10'
    }
}
```

// Configuração do script de construção do projeto, onde as dependências necessárias são especificadas. Neste caso, está sendo adicionada a dependência 'com.google.gms:google-services:4.3.10'.

// Arquivo de construção no nível superior onde você pode adicionar opções de configuração comuns a todos os subprojetos/módulos.

```
plugins {
    id 'com.android.application' version '7.3.0' apply false
    id 'com.android.library' version '7.3.0' apply false
}
```

// Plugins utilizados no projeto. Neste caso, estão sendo utilizados os plugins 'com.android.application' e 'com.android.library' com a versão '7.3.0'. O 'apply false' indica que a aplicação dos plugins está desativada.

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

// Definição de uma tarefa chamada 'clean' que é do tipo 'Delete'. Essa tarefa é responsável por excluir o diretório de construção do projeto (buildDir) do projeto raiz (rootProject).

Apêndice T – build.grade (Module)

```
plugins {
    id 'com.android.application' // Plugin para construção de aplicativo Android
    id 'com.google.gms.google-services' // Plugin para serviços do Google Play
}

android {
    compileSdk 32 // Versão do SDK de compilação

    defaultConfig {
        applicationId "com.example.supermecadotcc" // ID do pacote da aplicação
        minSdk 28 // Versão mínima do SDK suportada
        targetSdk 32 // Versão do SDK de destino
        versionCode 1 // Código da versão do aplicativo
        versionName "1.0" // Nome da versão do aplicativo

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner" // Runner de testes
    }

    buildTypes {
        release {
            minifyEnabled false // Desativa a minificação do código no modo de lançamento
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro' //
Arquivos de configuração do ProGuard
        }
    }

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8 // Compatibilidade de versão do Java
        targetCompatibility JavaVersion.VERSION_1_8 // Compatibilidade de versão do Java
    }

    namespace 'com.example.supermecadotcc' // Namespace da aplicação
```

```
    buildFeatures {
        viewBinding true // Ativa o recurso de ViewBinding
    }

    viewBinding {
        enabled = true // Ativa o ViewBinding
    }
}

dependencies {
    implementation 'androidx.appcompat:appcompat:1.5.1' // Biblioteca de suporte para o AndroidX
    implementation 'com.google.android.material:material:1.6.1' // Biblioteca de componentes de Material Design
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4' // Biblioteca de layout com restrições
    implementation 'androidx.navigation:navigation-fragment:2.5.2' // Biblioteca de navegação - Fragmentos
    implementation 'androidx.navigation:navigation-ui:2.5.2' // Biblioteca de navegação - Interface do usuário
    implementation 'androidx.legacy:legacy-support-v4:1.0.0' // Biblioteca de suporte legado
    implementation 'androidx.recyclerview:recyclerview:1.2.1' // Biblioteca de RecyclerView
    implementation 'com.google.firebase:firebase-database:20.0.4' // Biblioteca do Firebase - Banco de dados
    implementation 'com.google.firebase:firebase-storage:20.0.1' // Biblioteca do Firebase - Armazenamento
    implementation 'com.github.bumptech.glide:glide:4.12.0' // Biblioteca de carregamento de imagens
    implementation 'com.google.firebase:firebase-messaging:23.1.2' // Biblioteca do Firebase - Mensagens
    implementation 'com.google.firebase:firebase-messaging-ktx:23.1.2' // Biblioteca do Firebase - Mensagens (Kotlin)
    implementation 'com.google.firebase:firebase-inappmessaging-display-ktx:20.3.2' // Biblioteca do Firebase - Mensagens In-App (Kotlin)
    implementation 'com.google.firebase:firebase-inappmessaging-display:20.3.2' // Biblioteca do Firebase - Mensagens In-App
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0' // Processador de anotações do Glide

    testImplementation 'junit:junit:4.13.2' // Biblioteca de testes unitários
```

```
    androidTestImplementation 'androidx.test.ext:junit:1.1.3' // Biblioteca de testes de instrumentação
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0' // Biblioteca de testes de
UI
}
```