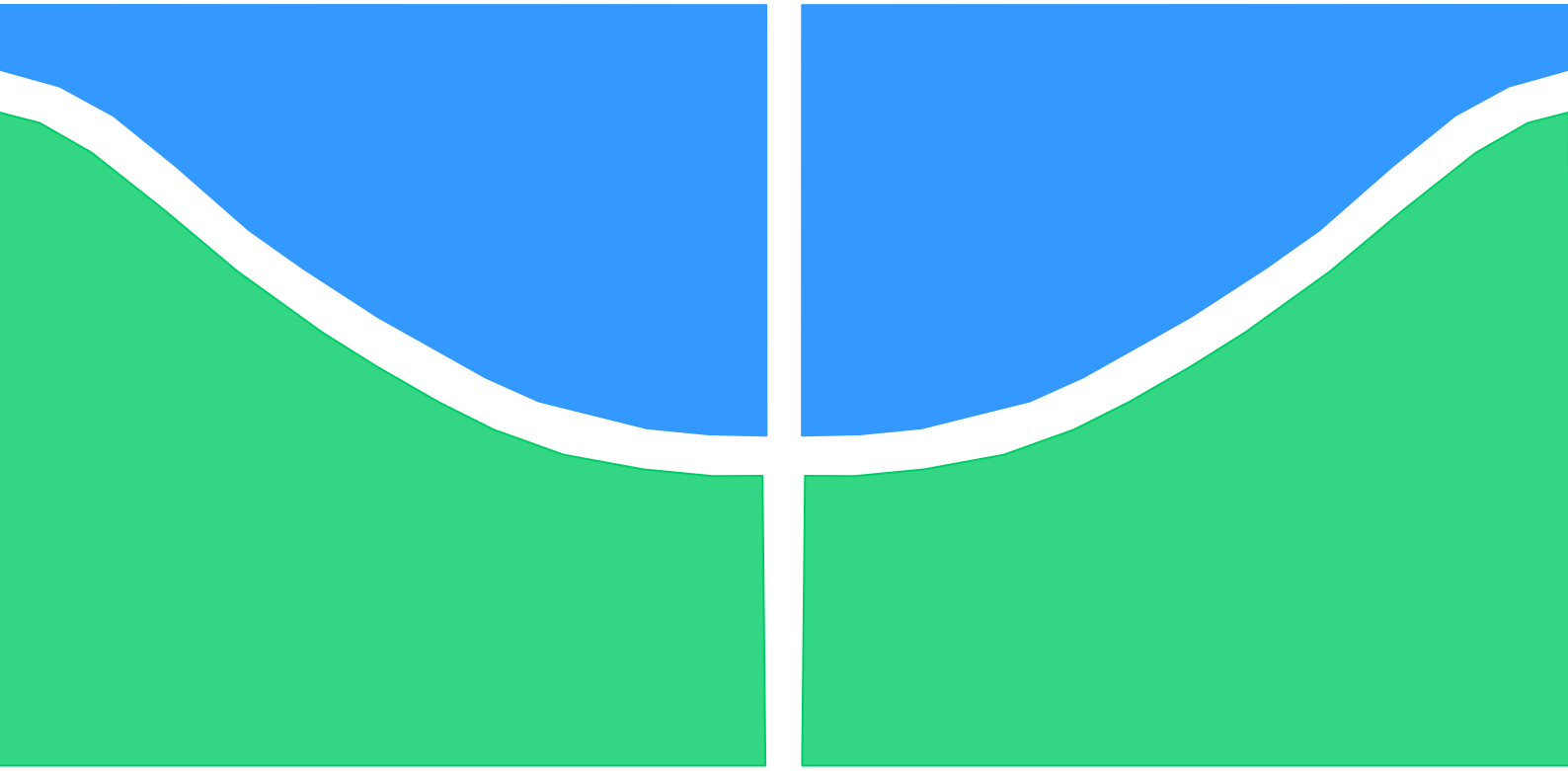

TRABALHO DE GRADUAÇÃO

GEOMARKETING BASEADO EM BEACONS

Isaac Aleixo Rodrigues Batista
João Paulo Fernandes Bezerra

Brasília, 26 de novembro de 2019

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA

Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

GEOMARKETING BASEADO EM BEACONS

Isaac Aleixo Rodrigues Batista
João Paulo Fernandes Bezerra

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Profª. Cláudia Jacy Barenco Abbas, UnB/ ENE
(Orientadora) _____

Prof. Georges Daniel Amvame Nze, UnB/ ENE _____

Prof. Alexandre Nery Solon, UnB/ ENE _____

Dedicatórias

Dedico este relatório a Deus, por sempre estar presente em minha vida. À minha namorada e futura esposa Tamires, pelos conselhos e companheirismo. À minha família, pela ajuda nas decisões difíceis.

João Paulo Fernandes Bezerra

Agradeço a todos os familiares e amigos que sempre me apoiaram durante minha jornada universitária.

Isaac Aleixo Rodrigues Batista

Agradecimentos

Agradeço a minha família em especial a minha mãe por sempre me apoiarem ao longo dessa jornada. Agradeço a minha orientadora Claudia Barenco Abbas por toda a dedicação e apoio prestados na realização desse projeto, assim como aos demais professores da UNB por todas a lições valiosas que me passaram.

Isaac Aleixo Rodrigues Batista

Agradeço a Deus por me iluminar e abençoar meu caminho, por me orientar e me guiar durante toda a graduação.
Aos meus pais e meus irmãos, que estiveram comigo por todo o caminho e me ajudaram em todas as decisões referentes à universidade.
À minha namorada, que me aconselhou e esteve comigo durante todos esses anos, sua presença em minha vida é essencial. Obrigado por todo o companheirismo nos momentos fáceis e difíceis.
Agradeço a professora Cláudia Jacy Barenco Abbas pela excelente orientação nesse projeto e pelo apoio ao realizar o artigo. Aos professores da UnB, que me orientaram e instruíram, fazendo de mim uma pessoa melhor.

João Paulo Fernandes Bezerra

RESUMO

A inovação no campo de tecnologias sem fio, como Bluetooth Low Energy (BLE), e o crescente mercado de Beacons, contribuiu para o desenvolvimento de uma solução que torna as compras mais agradáveis e eficientes para o cliente com o monitoramento em tempo real, facilitando a gestão do negócio.
O objetivo desse estudo é apresentar uma solução BLE (Bluetooth Low Energy) utilizando tecnologia Beacon aplicado ao Geomarketing. Neste trabalho desenvolve-se um aplicativo que faz uso da tecnologia BLE trazendo uma solução inovadora da maneira de interação Vendedor e Comprador.

Palavras-chave: Geomarketing; Beacons; Bluetooth Low Energy.

ABSTRACT

The breakthrough in the field of wireless technologies, such as Bluetooth Low Energy (BLE), and the growing market of Beacons, contributed to the development of a solution that makes shopping more enjoyable and efficient for the customer with the real time monitoring facilitating the business management.
The objective of this study is to present a BLE (Bluetooth Low Energy) solution using beacon technology applied to Geomarketing. In this work was developed an application that makes use of BLE technology bringing an innovative solution of the way Seller and Buyer interacts.

Keywords: Geomarketing; Beacons; Bluetooth Low Energy.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

API - Application Programming Interface
BLE – Bluetooth Low Energy
IDE – Integrated Development Environment
TTL – Time to Live
SDK – Software Development Kit
RFID – Radio Frequency Identification
NFC – Near Field Communication
Códigos QR – Códigos Quick Response
WiFi - Wireless Fidelity
REST - Representational State Transfer.
RF – Radio Frequency
IA – Artificial Intelligence
XML - Extensible Markup Language

1 INTRODUÇÃO

1.1 Introdução

Com o enorme avanço no campo das tecnologias de comunicação sem fio, como Bluetooth Low Energy, a disponibilidade de grande volume de dados sobre os clientes e o mercado crescente de Beacons, faz-se necessário o desenvolvimento de um sistema que possa ser usado em qualquer ambiente, para tornar as compras mais agradáveis e eficientes para o cliente, além de realizar monitoramento em tempo real, que torna mais fácil o gerenciamento da venda pelo vendedor.

A tecnologia de Beacons e o protocolo Bluetooth Low Energy tornaram o processo de coleta de dados sobre os clientes mais fácil e mais rápido, reduzindo custos, melhorando os serviços e realizando ofertas personalizadas com base nas preferências do cliente.

Foi desenvolvido um sistema de Geomarketing baseado em Bluetooth Low Energy (BLE) utilizando Beacons. Este sistema baseado em BLE consiste em dois componentes principais: Aplicativo de smartphone e beacons (BLE). O aplicativo para smartphone detecta a localização dos clientes utilizando a tecnologia BLE, e o aplicativo servidor envia promoções personalizadas, como anúncios móveis para smartphones dos clientes. Este trabalho visa demonstrar que o sistema de Geomarketing é capaz de detectar a localização do cliente, transmitir os dados via Bluetooth e exibir promoções personalizadas em um smartphone.

Este trabalho está organizado da seguinte forma: no capítulo 2 temos os Trabalhos Relacionados; no capítulo 3 temos a fundamentação teórica das tecnologias usadas neste estudo; no capítulo 4 descrevemos o desenvolvimento da solução e no capítulo 5 tem-se a Conclusão.

1.2 Objetivos

A proposta do projeto é a criação de uma solução BLE com víeis de aplicação em Geomarketing, para essa solução foi escolhido o desenvolvimento de uma aplicação para smartphones Android.

A escolha do Bluetooth Low Energy deu-se por sua extrema importância na transmissão de beacons. Os beacons são utilizados no BLE, por ter uma melhor economia de energia e por possuir transferência periódica de dados. Essas características fazem o BLE ser ideal para transmissão de Beacons e propagandas. Logo, o Bluetooth Low Energy foi a escolha precisa para este projeto de Geomarketing baseado em Beacons.

O aplicativo desenvolvido neste projeto tem grande importância no mercado de geomarketing, é uma solução relativamente nova para o mercado e possui inúmeras atribuições. As funcionalidades presentes em nossa aplicação não servem somente para vendas de produtos, com mudanças no aplicativo a funcionalidade pode se tornar útil em diversos campos como, trânsito, comércio, soluções domésticas, anúncios do governo e outras áreas de aplicações internas e externas, dentro de um alcance possível ao BLE.

O projeto é de grande importância para a tecnologia de Beacons e é uma inovação sobre como fazer marketing. Mostra disto é a publicação do artigo presente no Anexo I submetido à conferência ICMarktech'19 - Maia, Portugal e aceito para publicação na revista Springer, série SIST - Smart Innovation, Systems and Technologies em 2020.

O que nos motivou a desenvolver este projeto foi a inovação das tecnologias sem fio, em especial o Bluetooth Low Energy e o crescimento no uso de mensagens com beacon. Isso contribuiu para o desenvolvimento de nossa solução, que torna as compras mais agradáveis e eficientes para o cliente, com o monitoramento em tempo real, facilitando a gestão do negócio.

Nossa ideia é um marketing de pessoa a pessoa de maneira a inovar a comunicação e relação Comprador – Vendedor. O indivíduo que deseja vender algum produto, pode anunciar em nosso aplicativo de maneira que todas as pessoas em um alcance de Bluetooth Low Energy poderão visualizar a oferta e serão potenciais consumidores.

2 TRABALHOS RELACIONADOS

2.1 Smartphones e Serviços BLE

Impulsionado pelo rápido crescimento do mercado de sensores e beacons que oferecem conectividade baseada em Bluetooth Low Energy (BLE), o artigo “Smartphones & BLE Services: Empirical Insights” [8], estuda as características de desempenho da interface BLE em vários smartphones Android, e o consequente impacto de um serviço baseado em BLE proposto: localização indoor contínua.

Primeiro são utilizadas medidas extensivas nos estudos com vários dispositivos Android para estabelecer se o BLE nos smartphones atuais tem pouco consumo de energia quanto esperado, e estabelecer que o uso contínuo de uma interface BLE é melhor utilizado quando escolhido um intervalo moderado de varredura e um ciclo de trabalho baixo. Então são exploradas as implicações de tais restrições, nos parâmetros de uma pilha de BLE do smartphone, sobre a precisão de um sistema de técnicas de localização. Mostra-se que, enquanto indoor com base em RF a localização pode ser altamente precisa (80% das estimativas têm erros inferior ou igual a 4 metros) para usuários estacionários somente se a densidade de beacons é alta, a combinação de (grande intervalo de ciclo de trabalho baixo) faz com que o erro de localização se degrade significativamente para usuários em movimento.

Estes resultados fornecem informações práticas sobre os casos de uso e limitações para futuros serviços móveis baseados em BLE.

2.2 Beacon Marketing

O marketing de beacon é um canal de marketing de proximidade usado pelas empresas para interagir e se envolver com os consumidores em locais estratégicos (ver figura 2.1). Essa comunicação é acionada por um pequeno dispositivo de hardware conhecido como beacon Bluetooth. Esses dispositivos transmitem notificações avançadas acionadas por localização para smartphones nas proximidades.

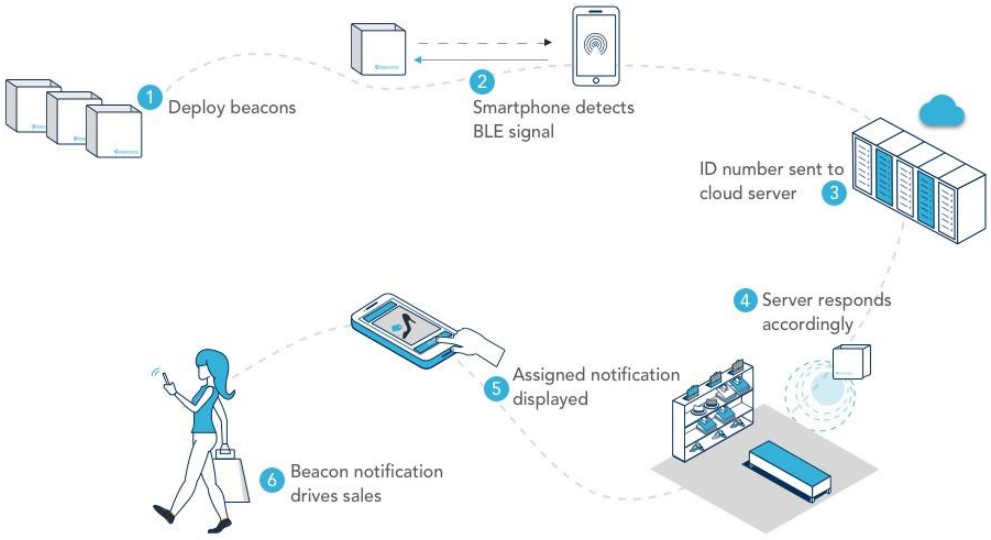

Fig.2.1 – Como funciona o Beacon Marketing [12]

Os beacons implantados em lojas, aeroportos e outros locais comerciais transmitem continuamente sinais de Bluetooth Low Energy em seu alcance. O intervalo de beacon é calibrado conforme o caso de uso. Varia de 10m a 300m. Aplicativos que utilizam beacons são capazes de escanear esses sinais de baixa energia. Depois que esses aplicativos detectam o sinal de beacon, ele encontra o ID anexado ao

sinal. O smartphone se refere ao servidor em nuvem usando o ID e busca a ação vinculada a ele. Essa ação pode notificar os usuários sobre uma oferta, um formulário de feedback ou o site da empresa. Essas notificações são ricas em aplicativos que abrem uma campanha, um cartão de descontos, um formulário ou um site.

No artigo "Developing a geomarketing solution" [7], uma solução inovadora de geomarketing é proposta. Esta solução consiste em três componentes: aplicativo no smartphone, o aplicativo do servidor e os beacons. O aplicativo para smartphone detecta a localização do cliente usando beacons e as informações são enviadas para o aplicativo do servidor para processamento, a fim de gerar o cenário adequado. Além disso, utiliza ferramentas de mineração de dados para entender as relações entre os produtos que as pessoas compram, para fornecer mensagens de marketing personalizadas e adaptar a loja às suas necessidades.

A solução implementada pela Beaconstac, caso mostrado acima na figura 2.1 possui uma grande desvantagem que é a não aplicação outdoor. Ela foi realizada apenas para marketing indoor. Já a aplicação desenvolvida neste trabalho tem o intuito de suprir demandas indoor e outdoor, com o alcance máximo sendo o mesmo alcance do Bluetooth Low Energy.

2.3 Sistema Interativo de Localização em Tempo Real Baseado em Rede BLE e Beacon

Nos últimos anos, os beacons Bluetooth vêm ganhando popularidade na implementação de uma variedade de serviços inovadores baseados em localização. No entanto, a tecnologia broadcast de beacon para transmissão só pode fornecer informações unidirecionais. Caso os usuários de smartphones desejem responder às mensagens do beacon, eles devem depender de suas próprias conexões de Internet móvel para enviar as informações de volta ao sistema backend. No entanto, os serviços de Internet móvel podem não estar sempre disponíveis ou são muito caros.

No artigo [11], sobre localização em tempo real baseado em BLE, é proposto desenvolver um sistema de localização em tempo real baseado apenas na tecnologia Bluetooth low energy (BLE) para suportar as comunicações interativas combinando as opções de topologia de broadcast e mesh para estender a aplicabilidade das soluções beacon. Especificamente, transformando o smartphone em um dispositivo beacon e aumentar os dispositivos de beacon com a capacidade de formar uma rede mesh. O resultado da implementação mostra que os dispositivos de beacon podem detectar a presença de usuários específicos em locais específicos e, em seguida, o estado de presença pode ser enviado para o servidor de aplicativos por meio da retransmissão dos dispositivos de beacon. Além disso, o servidor de aplicação pode enviar mensagens personalizadas aos usuários, novamente através da retransmissão aos dispositivos de beacon. Com a capacidade de retransmitir mensagens entre dispositivos de beacon, seria conveniente que os desenvolvedores implementassem uma variedade de aplicações como o rastreamento de clientes VIP no aeroporto ou rastreamento de um idoso com doença de Alzheimer dentro de um bairro.

2.4 Estimote

A Estimote é uma empresa de tecnologia que cria uma plataforma de análise e engajamento baseada em sensor. Eles desenvolveram um sistema operacional para locais físicos, um sistema que muda a maneira como as pessoas administram negócios no mundo físico e como os consumidores interagem com produtos e instalações do mundo real.

A principal área de interesse são as lojas físicas. Mais de 95% das transações em todo o mundo ainda são feitas em locais físicos [17]. Além disso, mais da metade dos consumidores que visitam lojas têm smartphones, e esse número está crescendo rapidamente.

Graças às tecnologias de comunicação populares, como Bluetooth e Wi-Fi, há uma oportunidade de entender como as pessoas se comportam e se envolvem com os produtos nas lojas. Novas tecnologias de dados e comunicação podem ser usadas para melhorar a experiência do cliente, trazendo novos fluxos de receita para as lojas ou reduzindo seus custos.

O SDK (*Software Development Kit*) do Estimote pode ser incorporado a aplicativos móveis, que podem usar sinais de sensores embutidos ou beacons próximos para estimar o contexto e a micro-localização de um evento. Com API simples, os desenvolvedores podem criar facilmente aplicativos de localização interna ou de proximidade e acionar ações pré-programadas ou enviar mensagens relevantes.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Geomarketing e Marketing com Beacons

O Geomarketing é uma técnica de mercado que se baseia na localização das pessoas para definir públicos-alvo e melhores táticas para conseguir alcançar o consumidor final no local e momento correto. Atrelado ao aumento no uso de dispositivos móveis, tablets, celulares e outros aparelhos, a informação da localização e dos locais visitados por uma pessoa, se usadas de forma eficiente por um algoritmo ou IA (Inteligência Artificial), podem trazer para as empresas informações valiosas para montar uma estratégia de marketing eficaz, conseguindo entregar anúncios e ofertas que sejam mais relevantes para seu cliente final.

Os Beacons Bluetooth Low Energy são uma maneira conveniente para os profissionais de marketing se comunicarem com os clientes de maneira relevante e personalizada. O crescimento exponencial de tecnologias Bluetooth e aplicativos mobile, fazem com que os beacons se tornem cada vez mais visados para os profissionais de marketing.

O Geomarketing com beacons vem crescendo e chama a atenção quanto às taxas de cliques em anúncios, as chamadas CTR (click-through rates). As taxas de cliques das campanhas de beacons são muito superiores às das tecnologias concorrentes. Essas tecnologias incluem RFID, NFC, códigos QR e WiFi.

De acordo com a empresa Beaconstac [6], o marketing de mídia social, como os anúncios do Facebook e do Twitter, também possui taxas de cliques abaixo da média das campanhas de beacons. A Taxa de Cliques média dos anúncios do Facebook em 2018 é de 0,119%, enquanto a Taxa de Cliques média dos anúncios do Twitter é de 1 a 3%. No entanto, a Taxa de Cliques média das campanhas de beacons é de 2-4%. Campanhas de sinalização bem executadas geram uma taxa de 12-15%. Com esses dados é possível notar que a tecnologia de anúncios utilizando beacons está evoluindo rapidamente.
Pode-se observar na figura abaixo as taxas de clique das propagandas em diferentes plataformas.

	Propagandas	*Taxa de Cliques*
Mostrar Anúncio	Toda a tela	0.05%
	Topo da tela	0.04%
	Seção lado direito	0.04%
Propagandas em Redes Sociais	Facebook	0.119%
	Twitter	1-3%
	Google Ads	0.10%
Anúncios de Beacon	Propagandas com Beacons Baseadas em Localização	2-4%

Fig.3.1 – Taxas de Cliques média em anúncios [6]

3.2 Bluetooth Low Energy (BLE)

BLE significa Bluetooth Low Energy ou Bluetooth LE, ele é comercializado como Bluetooth Smart.

O BLE é uma forma de comunicação sem fio projetada especialmente para comunicação de curto alcance. O BLE destina-se a situações em que a duração da bateria é preferida em relação às altas velocidades de transferência de dados. Por exemplo, se um usuário quiser transmitir campanhas de marketing nas proximidades de um telefone, a quantidade de dados que é necessária para transferir para outro smartphone extremamente pequena, portanto, os beacons compatíveis com BLE realizam o trabalho rapidamente sem esgotar a bateria.

A maioria dos smartphones e tablets hoje é compatível com BLE, o que significa que eles podem se comunicar facilmente com fones de ouvido sem fio habilitados para Bluetooth, sinalização digital, aparelhos de som automotivos, rastreadores de fitness, smartwatches e dispositivos de hardware como beacons.

A transferência de dados BLE pode ser bidirecional como modo orientado a conexão, porém é essencialmente uma comunicação unidirecional quando estamos falando de broadcast, que é o caso para transmissão de beacons.

Um beacon Bluetooth Low Energy transmite pacotes de dados em intervalos regulares de tempo, esses pacotes de dados são detectados por aplicativos em smartphones nas proximidades (como é o caso de nossa aplicação, demonstrada na parte 3 deste relatório). Essa comunicação BLE aciona ações como, por exemplo, enviar uma mensagem, promover um aplicativo ou enviar um anúncio.

Para economizar energia e fornecer maior velocidade de transferência de dados, toda a estrutura de comunicação BLE consiste em dois tipos de canais: advertising channels e os data channels, totalizando 40 canais de frequência, separados por 2MHz.

Três destes canais são os canais de publicidade (advertising channels) e são usados para descobrir outros dispositivos, estabelecer uma conexão e transmissão broadcast. Enquanto os restantes 37 canais são conhecidos como canais de dados ou data channels, eles são usados para a comunicação bi-direcional entre os dispositivos conectados.

É importante ressaltar que o Bluetooth clássico utiliza 32 canais como advertising, o que faz com que o tempo de busca de dispositivos (22.5 ms) seja muito maior do que o necessário para o BLE (0.6 - 1.2 ms).

A comunicação Bluetooth começa com os três primeiros canais de anúncio e depois descarrega para os canais de dados. Um exemplo dessa comunicação pode ser observado na figura 3.2.

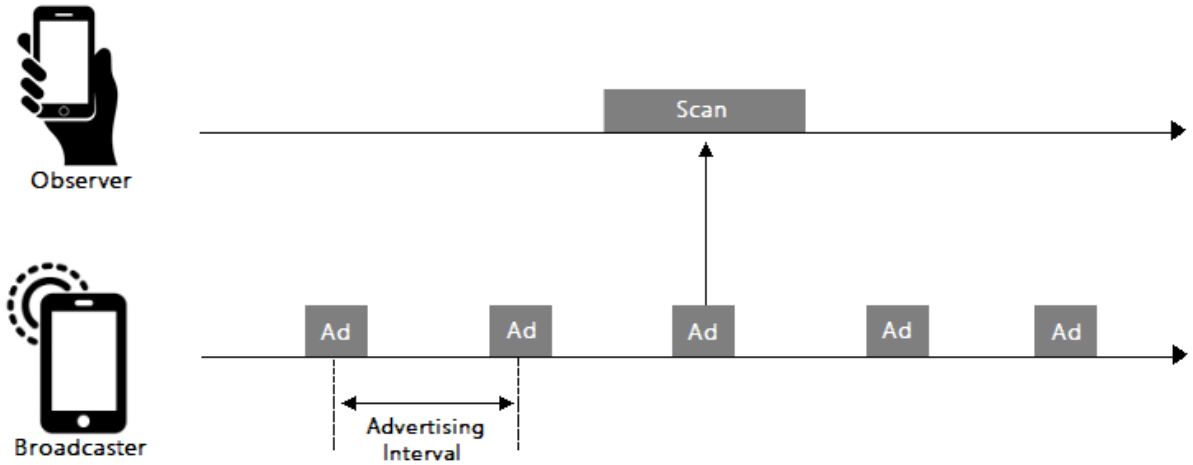

Fig.3.2 – Exemplo de comunicação BLE [6]

Aparelhos que suportam a tecnologia Bluetooth Low Energy:

Tabela 3.1 – Aparelhos com suporte BLE [5]

Aparelho	**Modelos com suporte BLE**
iPhone	iPhone 4 ou mais novos
iPad	iPad 3ª geração ou mais novos
iPod Touch	iPod Touch 5ª geração ou mais novos
Celulares Android	Android 4.3 ou mais novos
Tablets Android	Android 4.3 ou mais novos
iPad Mini	iPad mini 1ª Geração ou mais novos

A Tabela 3.1 mostra que a maioria dos aparelhos desenvolvidos a partir de 2012 suportam a tecnologia BLE. Entretanto, como os celulares Android possuem maior variedade e a população brasileira, em sua maioria, possui celular com o Sistema Operacional Android [21], optou-se por desenvolver para esta plataforma, por maior adesão do público e maior abrangência de aparelhos com suporte BLE.

Existem duas tecnologias principais dentro da especificação Bluetooth: Bluetooth Classic e Bluetooth Low Energy, ou Bluetooth Smart. A principal diferença entre os dois está no consumo de energia em cada caso. No entanto, existem outros fatores pelos quais o Bluetooth Smart está sendo usado para aplicações tecnológicas.

O Bluetooth Classic consome uma quantidade significante a mais de energia do que o Bluetooth Low Energy. O Bluetooth Classic é ideal para aplicativos que exigem fluxo contínuo de dados, como fones de ouvido. No entanto, o BLE é adequado para aplicativos que funcionam bem com uma transferência periódica de dados e, portanto, reduz uma quantidade significativa de uso de bateria. Isso torna o BLE adequado para aplicativos relacionados a marketing de proximidade, que é o caso dos Beacons e deste projeto.

O BLE pode estabelecer até 20 conexões simultaneamente, ele suporta mais conexões simultâneas pois transfere pequenos pacotes de dados e estabelece conexões rápidas. Já o Bluetooth Classic pode iniciar apenas 7 conexões simultâneas.

3.3 Beacons Bluetooth Low Energy

Beacons são pequenos transmissores Bluetooth que transmitem um identificador exclusivo para dispositivos eletrônicos próximos o suficiente para captarem seu sinal.

Os Beacons transmitem pequenas quantidades de dados via Bluetooth Low Energy (BLE), consequentemente tendo limitações de distâncias, como resultado, costumam ser usados para tecnologia de localização interna, embora os Beacons também possam ser utilizados em ambientes externos.

O identificador exclusivo e o campo de dados bytes transmitidos pelo Beacon podem ser utilizados para determinar a localização física do dispositivo, rastrear clientes ou acionar uma ação baseada em local no dispositivo, como uma notificação ao chegar em um lugar.

Primeiramente, é importante informar que os beacons são utilizados no Bluetooth Low Energy, exatamente por ter uma melhor economia de energia e por possuir transferência periódica de dados, e não um fluxo contínuo. Essas características fazem o BLE ser ideal para transmissão de Beacons e propagandas. Logo, ele é ideal para este projeto de Geomarketing baseado em Beacons.

Os Beacons são pequenos pacotes transmitidos via sinais BLE em um determinado intervalo. Este intervalo depende da capacidade do hardware. Em média, um beacon Bluetooth Low Energy pode transmitir sinais BLE para 80 metros em ambiente externo [12]. Este sinal BLE do beacon é capaz de realizar uma ação específica relevante para a localização, como por exemplo, mostrar um anúncio.

Outros métodos de publicidade também são possíveis com beacons: URIBeacon e Eddystone do Google permitem um modo de transmissão de URI. Os beacons de URI transmitem um URI que pode ser um link para uma página da Web e o usuário verá esse URI diretamente em seu telefone.

Os Beacons transmitem pacotes de dados e os componentes desses pacotes são ligeiramente diferentes para os iBeacons e para o Eddystone. Mas, em geral, eles apenas contêm um ID com seus dados, um componente que indica o status do Beacon e uma URL, apenas para Beacons Eddystone.

Os Beacons enviam um número de ID através dos canais BLE, aproximadamente 10 vezes por segundo. Um dispositivo habilitado para Bluetooth nas proximidades do Beacon capta esse número de ID. Quando um aplicativo ou serviço pré-instalado como o Google Nearby reconhece o número de ID, vincula o Beacon a uma ação, como o download de um aplicativo ou uma oferta de marketing, e a exibe no smartphone.

Existem diversas possibilidades de implementações para Beacons, entre elas estão Geomarketing e Marketing por proximidade, o foco do projeto é que usuários consigam enviar anúncios para as pessoas ao redor, utilizando Beacons BLE.

3.5 Google Nearby

O Google Nearby tem como característica a proximidade e comunicação entre dispositivos. A plataforma Nearby facilita a descoberta de dispositivos próximos e o estabelecimento de comunicação com eles. Ele usa tecnologias como Bluetooth, Wi-Fi, IP e áudio, no caso do nosso projeto a API foi utilizada com a tecnologia Bluetooth. O Google Nearby construiu duas API’s baseadas no Nearby, são elas: (i) Nearby Connections e (ii) Nearby Messages.

(i) Nearby Connections: Permite publicidade, descoberta e conexões entre dispositivos próximos de maneira ponto a ponto totalmente offline. As conexões entre dispositivos utilizam grande largura de banda, baixa latência e são totalmente criptografadas para permitir transferências de dados rápidas e seguras.

O principal objetivo desta API é fornecer uma plataforma simples, confiável e com bom desempenho. A API usa uma combinação de pontos de acesso Bluetooth, BLE e Wifi, aproveitando os pontos fortes de cada um e contornando suas respectivas fraquezas. Isso efetivamente abstrai os caprichos do Bluetooth e Wifi em uma variedade de versões e hardwares do sistema operacional Android, permitindo que os desenvolvedores se concentrem nos recursos que são importantes para seus usuários.

Por conveniência, os usuários não precisam ativar o Bluetooth ou o Wi-Fi, o Nearby Connections habilita esses recursos conforme necessário e restaura o dispositivo ao seu estado anterior assim que o aplicativo para de usar a API, garantindo uma experiência confortável ao usuário.

(ii) Nearby Messages: uma API de publicação e assinatura que permite transmitir pequenas cargas binárias entre dispositivos Android conectados à Internet. Os dispositivos não precisam estar na mesma rede, mas precisam estar conectados à Internet.

O Nearby usa uma combinação de Bluetooth e Bluetooth Low Energy, para comunicar um código de emparelhamento exclusivo no tempo entre os dispositivos. O servidor facilita a troca de mensagens entre dispositivos que detectam o mesmo código de emparelhamento. Quando um dispositivo detecta um código de emparelhamento de um dispositivo próximo, envia o código de emparelhamento ao servidor do Nearby Messages para validação e para verificar se há alguma mensagem a ser entregue para o conjunto atual de assinaturas do aplicativo.

O recurso Mensagens próximas não é autenticado e não requer uma Conta do Google. O mecanismo exato para troca de dados pode variar de versão para versão. A sequência a seguir mostra os eventos que levam à troca de mensagens:

1. O dispositivo de publicação faz uma solicitação para associar uma carga binária (a mensagem) a um código de emparelhamento exclusivo (token). O servidor faz uma associação temporária entre a mensagem e o token.
2. O dispositivo de publicação usa uma combinação de Bluetooth, Bluetooth de baixa energia e Wi-Fi para tornar o token detectável por dispositivos próximos. O dispositivo de publicação também usa essas tecnologias para procurar tokens de outros dispositivos.
3. Um aplicativo de subscrição associa sua assinatura a um token e usa uma combinação das tecnologias acima para enviar seu token ao editor e detectar o token do editor.
4. Quando um dos lados detecta o token do outro, ele o reporta ao servidor.
5. O servidor facilita a troca de mensagens entre dois dispositivos quando ambos estão associados a um token comum.

3.6 Proximity Beacon

A API do Proximity Beacon faz parte da plataforma de beacon Bluetooth de baixa energia (BLE), que também inclui o Eddystone, um formato de beacon aberto do Google.

A API Proximity Beacon é um serviço de nuvem que permite gerenciar dados associados aos seus beacons BLE usando uma interface REST (Representational State Transfer).

Você pode associar dados aos seus Beacons registrados como anexos. Anexos são bolhas arbitrárias de dados que são veiculadas como mensagens para seus aplicativos Android e iOS por meio da API de Mensagens próximas. Você pode atualizar anexos remotamente, eliminando a necessidade de visitar fisicamente cada Beacon.

Você pode monitorar a saúde de seus beacons através do terminal de diagnóstico da API Proximity Beacon, para detectar níveis de bateria, determinar se um beacon foi movido e muito mais.

Quando você registra beacons com a API Proximity Beacon, os seguintes campos são usados como sinal pela API do Google Places: Coordenadas de latitude e longitude; nível do piso; ID do local da API do Google Places.

4 DESENVOLVIMENTO DO APLICATIVO

O aplicativo desenvolvido consiste em duas telas:uma tela de venda e outra tela para compras. A tela de vendas serve para registrar e publicar anúncios e a tela de ofertas para fazer a varredura de anúncios. Uma visão da aplicação encontra-se no fluxograma da figura 4.1.

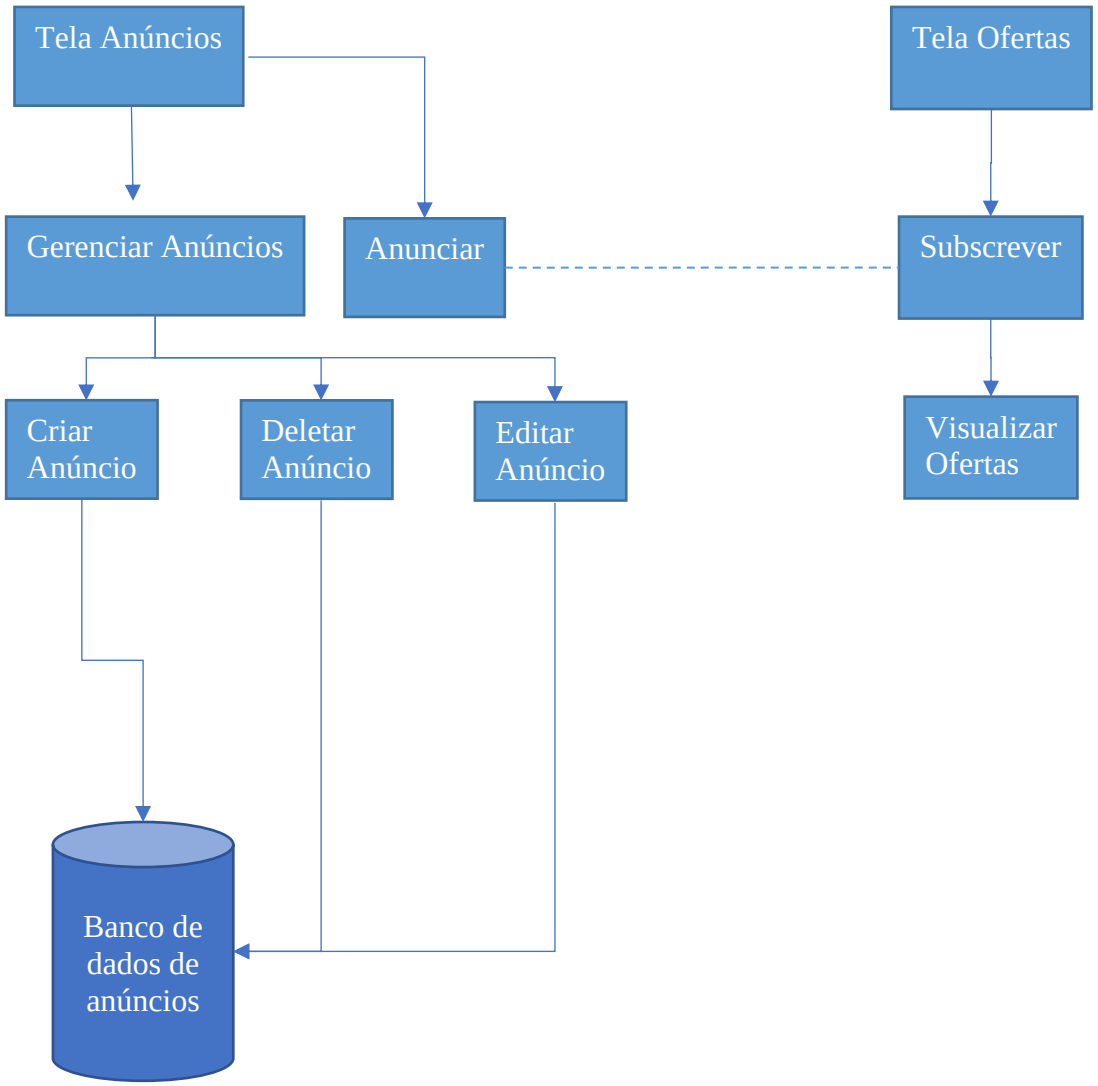

Figura 4.1: Fluxograma do aplicativo.

4.1 Ambiente de Desenvolvimento

O software que foi desenvolvido neste projeto foi escrito utilizando linguagem de programação JAVA com a utilização do ambiente de desenvolvimento integrado (IDE) Android Studio.

O sistema de compilação da IDE é baseada em Gradle, o Gradle é uma ferramenta open-source voltada para construção de softwares.

O Android Studio foi a IDE escolhida pois facilita no trabalho de construção de aplicativos Android que é o tipo de dispositivo alvo para a aplicação construída, dentro da IDE além de poder criar emuladores de ambiente de teste é possível compilar e testar o código diretamente em um dispositivo Android.

A IDE conta com integração com GitHub para o controle de versão do software, ferramentas de verificação de erros de código e ferramentas de auxílio para criação de designs utilizando drag and drop e xml.

A estrutura do projeto Android dentro da IDE se dá da seguinte forma (ver figura 4.2):

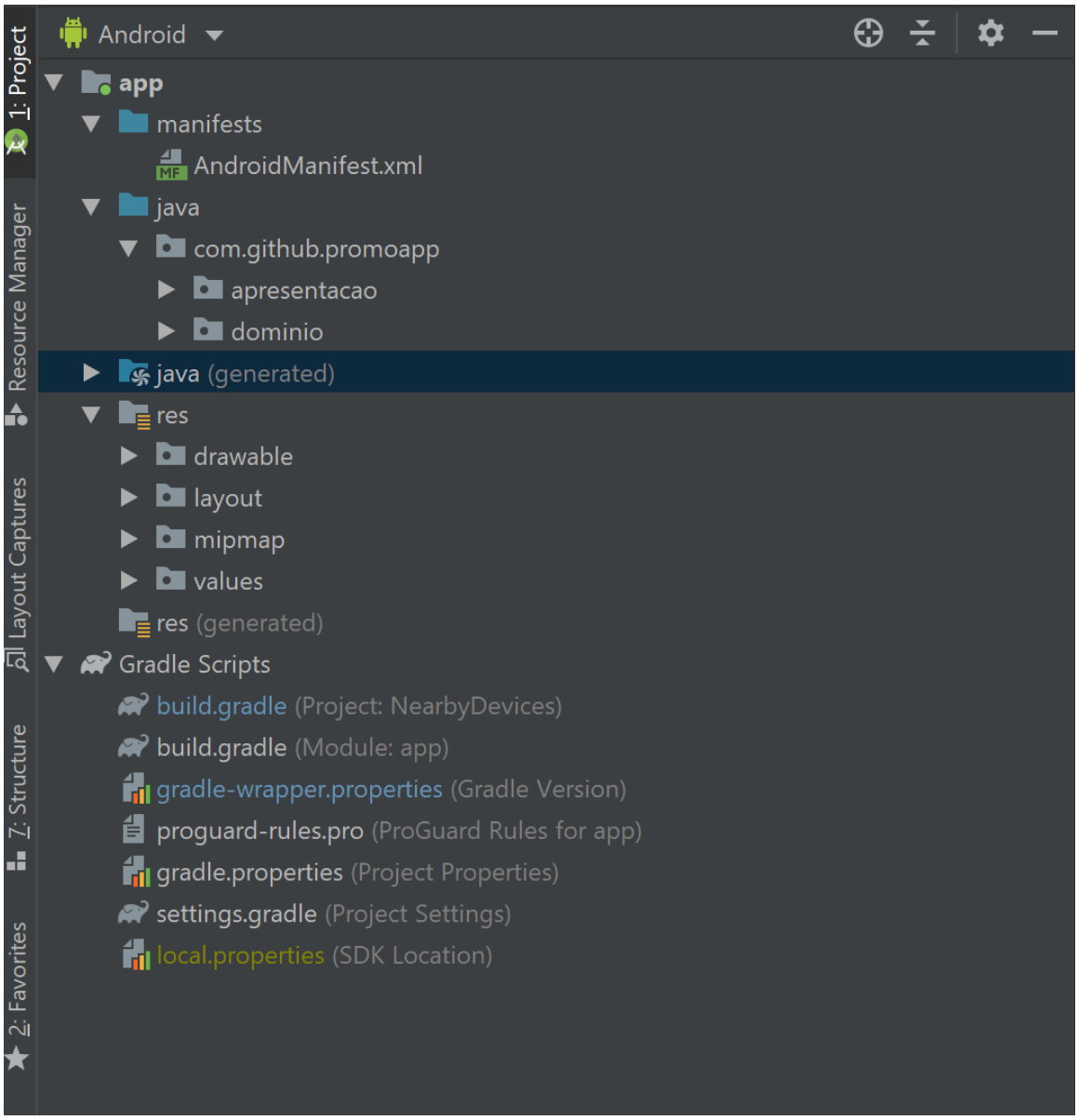

Figura 4.2: Estrutura do projeto na IDE.

- Manifestos: Esse módulo contém o arquivo *AndroidManifest.xml* , o arquivo *manifest* apresenta informações do aplicativo ao sistema Android, necessárias para o sistema antes que ele possa executar o código do aplicativo. É no arquivo *manifest* que são declaradas as permissões necessárias para acessar partes protegidas da API e para interagir com outros apps assim como as permissões necessárias que o dispositivo precisa liberar para acessar os componentes do aplicativo, por exemplo as permissões para uso do BLE, conforme a figura 4.3:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.gms.nearby.messages.samples.nearbydevices">

    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Figura 4.3: Snippet arquivo manifest

- Java: Contém os arquivos do código-fonte do Java, tais como os códigos de teste. É neste módulo que é possível criar os atributos do aplicativo e as classes e métodos que ditam como as telas se comportam e as ações que são feitas pelos botões, switches e outros componentes aplicativo. Na figura 4.4 é possível verificar uma amostra desse módulo no projeto de *NearbyMessages* utilizado para a aplicação BLE:

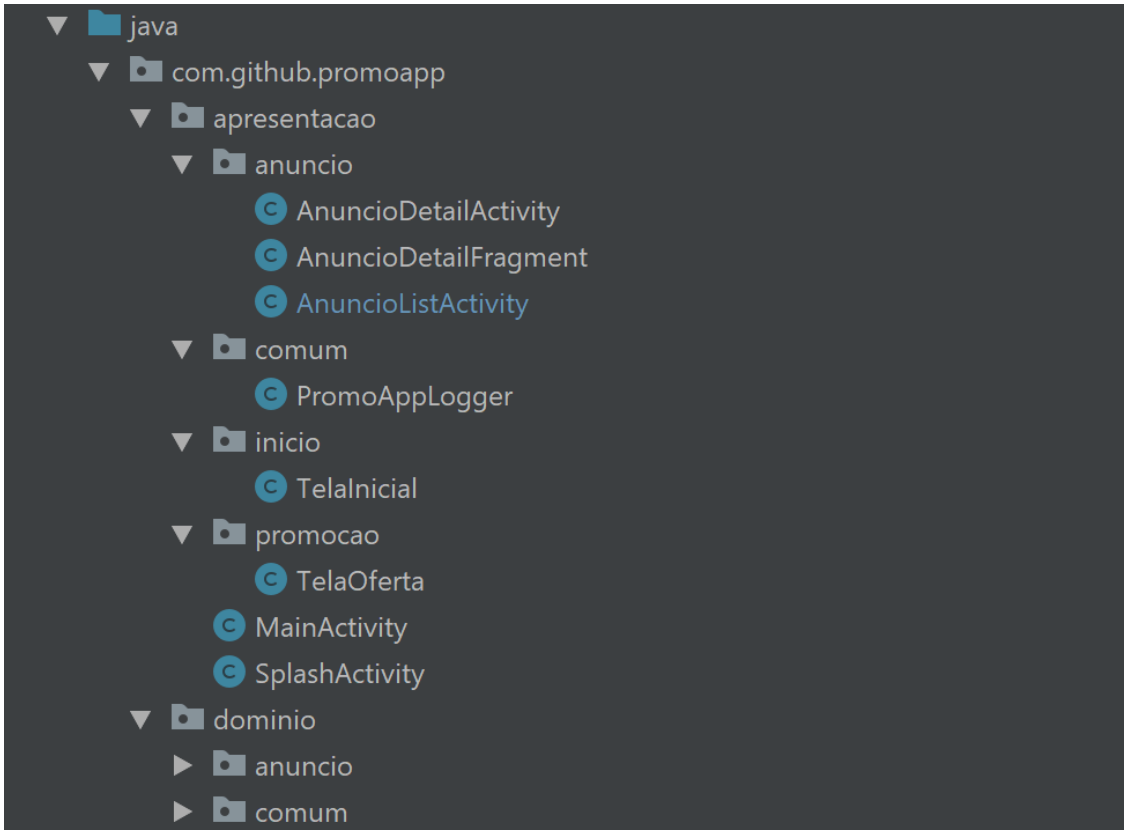

Figura 4.4:Snippet módulo Java.

- Res: O módulo "*Resources*" ou recursos, contém os recursos que podem ser utilizados nos outros módulos, contém layouts(arquivos xml com modelos de telas a serem usados no app), imagens, ícones, *values*(*strings*, números inteiros e cores). Na figura 4.5 pode ser visualizado um *snipet* do módulo res da aplicação *NearbyDevices*:

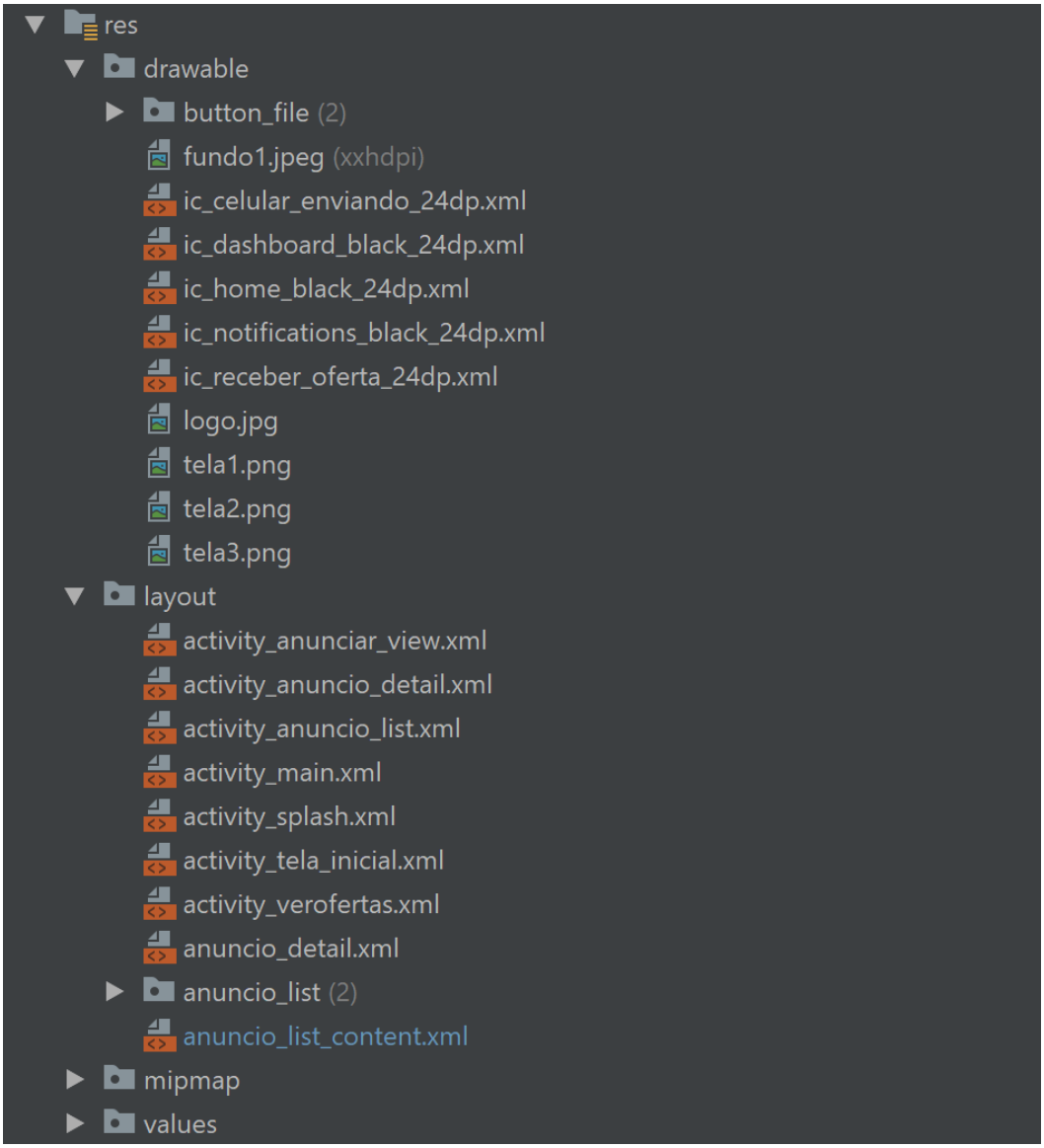

Figura 4.5: *Snipet* módulo *res*.

4.2 API Nearby Messages

A API utilizada para a construção do aplicativo é a *Nearby Messages* API da Google, a API faz uso tanto do Bluetooth, BLE e Wi-Fi para fazer com dispositivos se comuniquem, o servidor faz com que seja fácil a troca de mensagens entre dispositivos com o mesmo tipo codificação de pareamento.

A troca de mensagens entre dispositivos se dá da seguinte forma:

I. Um aplicativo de publicação faz uma solicitação para associar a mensagem em um código de pareamento exclusivo (token).

II. O dispositivo de publicação usa uma combinação de Bluetooth, Bluetooth Low Energy, Wi-Fi para tornar o token detectável por dispositivos próximos. O dispositivo de publicação também usa essas tecnologias para procurar tokens de outros dispositivos.

III. Um aplicativo de subscrição associa sua assinatura a um token e usa uma combinação das tecnologias acima para enviar seu token ao dispositivo de publicação e para detectar o token do dispositivo de publicação.

IV. Quando um dos lados detecta o token do outro, ele o informa ao servidor.

V. O servidor facilita a troca de mensagens entre dois dispositivos quando ambos estão associados a um token comum e as chaves de API usadas pelos aplicativos de chamada são associadas ao mesmo projeto no Google Developers Console.

Para integrar a aplicação com a Google Play *services*, é necessário criar um método *buildGoogleApiClient*(), tanto na classe responsável por publicar quanto na classe responsável por realizar a subscrição (ver figura 4.6).

```
private void buildGoogleApiClient() {
    if (mGoogleApiClient != null) {
        return;
    }
    mGoogleApiClient = new GoogleApiClient.Builder( context: this)
            .addApi(Nearby.MESSAGES_API, new MessagesOptions.Builder()
                    .setPermissions(NearbyPermissions.BLE).build())
            .addConnectionCallbacks(this)
            .enableAutoManage( fragmentActivity: this,  onConnectionFailedListener: this)
            .build();
}
```

Figura 4.6:Método *buildGoogleApiClient*().

4.2.1 Google Devolepers Console

Para usar a API *Nearby Messages* é necessário ter uma conta Google, porém para o usuário final do aplicativo não é necessário ter esta conta.

Para utilizar a API é necessário criar uma *API KEY* no *Google Developers Console*, a *API KEY* serve também para restringir quais aplicações podem ter acesso aos Beacons com as mensagens publicadas pelos publicadores, para o cenário proposto, somente as aplicações sobre a mesma *API KEY* podem visualizar os beacons.

A *API KEY* é configurada no arquivo AndroidManifest.xml (ver figura 4.7).

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.sample.app" >
    <application ...>
        <meta-data
            android:name="com.google.android.nearby.messages.API_KEY"
            android:value="API_KEY" />
        <activity>
        ...
        </activity>
    </application>
</manifest>
```

Figura 4.7: Configurando *API KEY* no *AndroidManifest.xml*.

4.3 Publicando e Subscrevendo Beacons

Primeiramente, para certificar que somente Beacons serão publicados e subscritos, é necessário configurar a estratégia. A *Messages API* da Google fornece 2 estratégias:

- *Default*: O comportamento padrão está atualmente fazendo publicações e subscrições, usando todos os sensores disponíveis, para descobrir dispositivos próximos, independentemente da distância.

-*BLE_Only*: O comportamento do *BLE-Only* é de fazer publicações e subscrições apenas utilizando beacons, no caso essa é a estratégia a ser utilizada para o APP e para estudar as propriedades dos Beacons e do Bluetooth Low Energy.

Escolhida a estratégia a ser utilizada, agora pode-se criar os métodos para publicar e subscrever os beacons, os métodos foram criados em classes separadas, uma classe java é responsável pela publicação de Beacons e outra classe java responsável pela subscrição dos Beacons.

4.3.1 Classes do projeto - Apresentação

Dentro da pasta java foram criadas as pastas: apresentação e domínio. (ver figura 4.8)

Figura 4.8: Classes do projeto.

Dentro da pasta apresentação estão as classes que guardam as informações de como cada tela se comporta dentro do aplicativo.

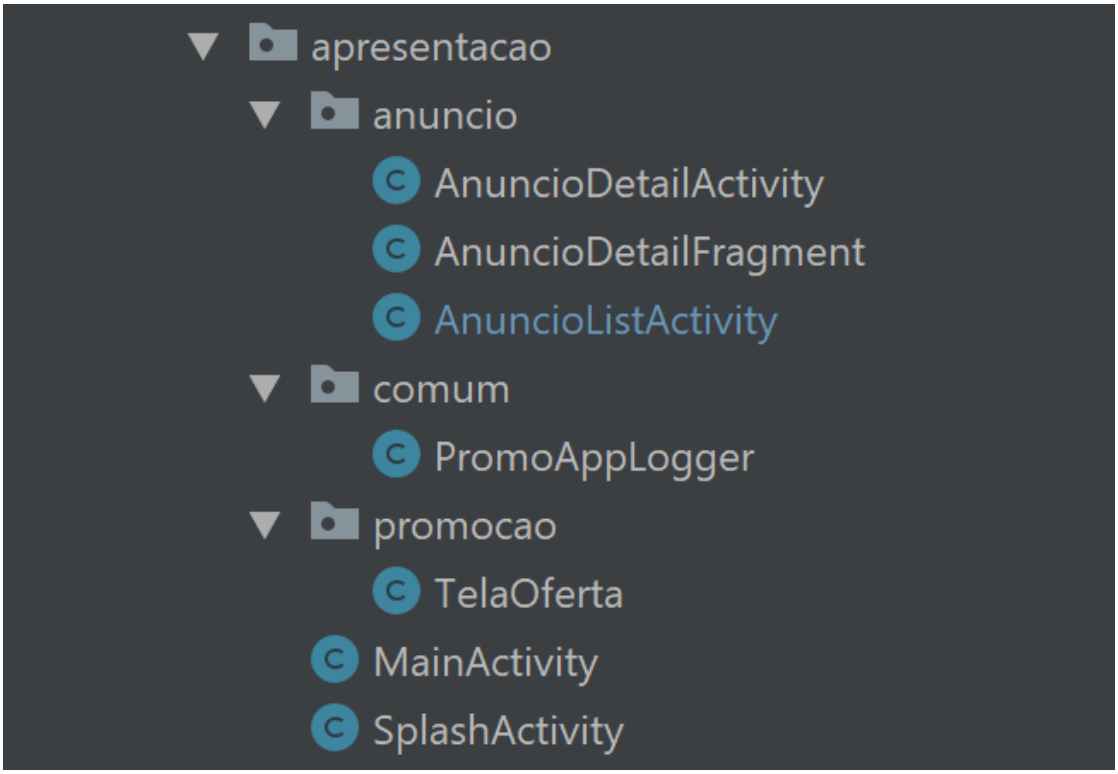

Figura 4.9: Pasta apresentacao

Como pode ser visto na figura 4.9, a apresentação está dividida em:

- Anúncio;
- Comum;
- Promoção;
- Raiz do diretório;

4.3.1.1 Anúncio

Dentro da pasta anúncio estão as classes referentes a tela de anúncios, onde é possível criar um anúncio no banco de dados, editar ou deletar anúncios criados, e por fim publicar os anúncios desejados para outros dispositivos.

Dentro da classe *AnuncioListActivity* é criado o método *private void publish*() ,dentro das opções de publicação existe o parâmetro *.setStrategy*, dentro da classe *Strategy* da *API Nearby Messages*, tem o construtor para o *BLE ONLY* (ver figura 4.10).

```
public static final Strategy BLE_ONLY = (new Strategy.Builder()).zzjS(2).setTtlSeconds(2147483647).build();
```

Figura 4.10: Construtor *Strategy BLE_ONLY*.

E logo no começo da classe *TelaAnuncio* é criado uma variável para armazenar essa estratégia (ver figura 4.11).

```
private static final Strategy PUB_SUB_STRATEGY = (new Strategy.Builder()).zzjS(2)
        .setTtlSeconds(TTL_IN_SECONDS).build();
```

Figura 4.11: Variável para armazenar estratégia.

Com isto feito é possível criar o método *publish*(), com o parâmetro de estratégia adotado (ver figura 4.12).

```
private void publish() {
    Log.i(TAG, msg: "Publishing");
    PublishOptions options = new PublishOptions.Builder()
            .setStrategy(PUB_SUB_STRATEGY)
            .setCallback(onExpired() → {
                    super.onExpired();
                    Log.i(TAG, msg: "No longer publishing");
                    runOnUiThread(() → {
                            mPublishSwitch.setChecked(false);
                    });
            }).build();

    Nearby.Messages.publish(mGoogleApiClient, mPubMessage, options)
            .setResultCallback((ResultCallback) (status) → {
                    if (status.isSuccess()) {
                        Log.i(TAG, msg: "Published successfully.");
                    } else {
                        logAndShowSnackbar( text: "Could not publish, status = " + status);
                        mPublishSwitch.setChecked(false);
                    }
            });
}
```

Figura 4.12: Método *publish*().

Na classe *AnuncioDetailFragment* estão os métodos que permitem editar ou criar novos anúncios no banco de dados (ver figura 4.13), os detalhes de como esses métodos funcionam serão melhor explicados na seção 4.3.2- Domínio, onde serão melhor explicados os métodos referentes ao banco de dados.

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

        if (getArguments().containsKey(ARG_ITEM_ID)) {
            Long id = Long.parseLong(getArguments().getString(ARG_ITEM_ID));
            AnuncioRepository anuncioRepository = new AnuncioRepository(getContext());

            if (id != 0) {
                mItem = anuncioRepository.recuperarPorId(id);
            } else {
                mItem = new Anuncio();
            }

            Activity activity = this.getActivity();
            CollapsingToolbarLayout appBarLayout = activity.findViewById(R.id.toolbar_layout);

            if (appBarLayout != null) {
//                appBarLayout.setTitle("Anúncio");
            }
        }
    }
```

Figura 4.13: Criando ou editando anúncios.

A condição *If* verifica se o anúncio existe utilizando o operador diferença, ou seja, se o ID do anúncio for diferente de zero é porque o anúncio já existe no banco de dados, sendo assim o aplicativo permite que seja editado o anúncio selecionado através do método *recuperarPorId*(), caso contrário o aplicativo que seja criado um novo objeto do tipo Anuncio.

Na classe *AnuncioListActivity* é possível realizar as ações de deletar ou publicar um anúncio através do botão de deletar e do *toogle* de publicar respectivamente. (Ver figuras 4.14 e 4.15).

```
mDeleteButton.setOnClickListener((view) → {
        try {
                AnuncioRepository anuncioRepository = new AnuncioRepository(
                        mActivity.getApplicationContext());
                Long id = mAnuncio.getId();
                mAnuncio = anuncioRepository.recuperarPorId(id);

                anuncioRepository.deletar(mAnuncio);

            Snackbar.make(view, text: "Anúncio deletado com sucesso", Snackbar.LENGTH_LONG)
                    .setAction( text: "Action", listener: null).show();
        } catch (Exception ex) {
            Snackbar.make(view, text: "Ocorreu um erro ao tentar deletar o anúncio",
                    Snackbar.LENGTH_LONG)
                    .setAction( text: "Action", listener: null).show();
            Log.i(getClass().getSimpleName(),ex.getMessage());
            ex.printStackTrace();

        }
});
```

Figura 4.14: Deletando anúncios.

Para deletar um anúncio primeiro é necessário recuperá-lo do banco de dados através do método *recuperarPorId*(), e em seguida aplicar o método deletar, ambos os métodos serão explicados na seção 4.3.2 – Domínio.

```
mPublishSwitch.setOnCheckedChangeListener((buttonView, isChecked) → {
        if (mGoogleApiClient != null && mGoogleApiClient.isConnected()) {
            if (isChecked) {
                publish();
            } else {
                unpublish();
            }
        }
});
```

Figura 4.15: Publicando e não publicando anúncios.

É criado um *Toogle* que ativa o método *publish*() que já foi visto na figura 4.12, o *Toogle* ativa o método para publicar quando ativo e quando desativo o *toogle* interrompe a publicação do beacon, o entendimento dessas funcionalidade fica fácil de compreender na seção 4.4 que mostra a aparência do aplicativo.

Vale ressaltar que antes de iniciar a publicação, o aplicativo vai enviar para a *Google Developer Console* a informação da *API KEY*, após esse passo será possível publicar para outros aplicativos que também possuam a mesma *API KEY* validada conforme explicado na seção 4.2.1.

O método *unpublish* é definido utilizando um método já existente da *API Nearby Messages*, conforme pode ser visto na Figura 4.16:

```
private void unpublish() {
    Log.i(getClass().getSimpleName(), msg: "Unpublishing.");
    Nearby.Messages.unpublish(mGoogleApiClient, mPubMessage);
}
```

Figura 4.16: Método *unpublish*()

4.3.1.2 Comum

Dentro da pasta comum é definida a classe *PromoAppLogger*, essa classe é responsável por criar uma barra de mensagem na parte inferior da tela com um aviso, essa funcionalidade é chamada de *snackbar*, a mensagem que é mostrada pelo *snackbar* é um feedback sobre alguma operação executada pelo usuário.

```
public class PromoAppLogger {

    public static void logarExibirSnackbar(View view, String text) {
        Log.w(view.getClass().getSimpleName(), text);

        if (view != null) {
            Snackbar.make(view, text, Snackbar.LENGTH_LONG).show();
        }
    }
}
```

Figura 4.17: Classe *PromoAppLogger*

4.3.1.3 Promoção

Para visualizar as ofertas primeiro é necessário que tenha algum publicador no alcance do dispositivo, sendo essa condição cumprida, o dispositivo que deseja realizar a subscrição executará o método *subscribe*() que se encontra na classe *TelaOferta* localizado dentro da pasta promoção.

O método *subscribe* é ativado assim que o usuário se direciona a *TelaOferta*, uma vez que o usuário entra na tela de ofertas o celular funcionará como uma bússola e irá encontrar as ofertas dentro do alcance do BLE do dispositivo e jogá-las para dentro de um *array* de anúncios.

Conforme pode ser visto na figura 4.18, o método *onCreate*() é reponsável por criar a tela com seus atributos, basicamente os atributos de visualização definidos no arquivo *.xml* correspondente. Em seguida tem-se o método *onFound*(), esse método é onde os anúncios são adicionados no *array mAnuncios*.

Diferente da seção 4.3.1.2, os anúncios agora não são mais salvos em um banco de dados, uma vez que o anúncio de um ofertante sai do alcance do dispositivo, o anúncio não estará mais disponível.

O método *onLost*() que também pode ser visto na figura 4.18, é o método que remove o anúncio do *array* assim que o dispositivo não se encontra mais no alcance de subscrição.

Em seguida é utilizado o método *buildGoogleApiClient*(), o qual serve para se comunicar com a API do Google, os detalhes desse método já foram expostos na seção 4.2. Conforme pode ser visto na figura 4.18 o método responsável por realizar a subscrição dos beacons – *subscribe*(), está dentro de um condicional "*if*" que faz com que o método apenas seja executado após cumprido os requisitos de autenticação e conexão na API.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_promocao_list);

    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    toolbar.setTitle(getTitle());

    if (findViewById(R.id.promocao_detail_container) != null) {
        mTwoPane = true;
    }

    View recyclerView = findViewById(R.id.promocao_list);
    setupRecyclerView((RecyclerView) recyclerView);

    mMessageListener = new MessageListener() {
        @Override
        public void onFound(final Message message) {
            Anuncio anuncio = AnuncioMessage.fromNearbyMessage(message).getAnuncio();
            mAnuncios.add(anuncio);

            View recyclerView = findViewById(R.id.promocao_list);
            setupRecyclerView((RecyclerView) recyclerView);
        }

        @Override
        public void onLost(final Message message) {
            mAnuncios.remove(AnuncioMessage.fromNearbyMessage(message).getAnuncio());
            View recyclerView = findViewById(R.id.promocao_list);
            setupRecyclerView((RecyclerView) recyclerView);
        }
    };

    buildGoogleApiClient();

    if (mGoogleApiClient != null && mGoogleApiClient.isConnected()) {
        subscribe();
    }
```

Figura 4.18: Criando *TelaOferta*

Assim como no caso da publicação é necessário definir a estratégia adotada para a subscrição, e assim como anteriormente só interessa subscrever em Beacons, portanto, a estratégia utilizada permanece sendo *BLE ONLY* conforme pode ser verificado na figura 4.19 que mostra como é criado o método *subscribe*().

```
private void subscribe() {
    Log.i(TAG,  msg: "Subscribing");
    mNearbyDevicesArrayAdapter.clear();
    SubscribeOptions options = new SubscribeOptions.Builder()
            .setStrategy(PUB_SUB_STRATEGY)
            .setCallback(onExpired() → {
                    super.onExpired();
                    Log.i(TAG,  msg: "No longer subscribing");
                    runOnUiThread(() → {
                            mSubscribeSwitch.setChecked(false);
                    });
            }).build();

    Nearby.Messages.subscribe(mGoogleApiClient, mMessageListener, options)
            .setResultCallback((ResultCallback) (status) → {
                    if (status.isSuccess()) {
                        Log.i(TAG,  msg: "Subscribed successfully.");
                    } else {
                        logAndShowSnackbar( text: "Could not subscribe, status = " + status);
                        mSubscribeSwitch.setChecked(false);
                    }
            });
}
```

Figura 4.19: Método *subscribe*().

Vale ressaltar que antes de iniciar a visualização de anúncios publicados, o aplicativo vai enviar para a *Google Developer Console* a informação da *API KEY*, após esse passo será possível subscrever em outros aplicativos que também possuam a mesma *API KEY* validada conforme explicado na seção 4.2.1.

4.3.1.4 Raiz do diretório apresentação

Na raiz do diretório apresentação estão as classes *MainActivity* e *SplashActivity*.

A classe *MainActivity* (ver figura 4.20) contém dois botões que direcionam o usuário para a tela de gerenciamento de anúncios conforme seção 4.3.1.1 ou para tela de gerenciamento de ofertas conforme seção 4.3.1.3, sempre após a inicialização do aplicativo o usuário será direcionado diretamente para essa tela, portanto é a tela principal do aplicativo.

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    //Button Anuncios
    public void exibirAnuncioListActivity(View view) {
        Intent intent = new Intent( packageContext: MainActivity.this, AnuncioListActivity.class);
        startActivity(intent);
    }



    //Button Ofertas
    public void openTelaOfertas(View view) {
        Intent intent = new Intent( packageContext: MainActivity.this, TelaOferta.class);
        startActivity(intent);
    }
```

Figura 4.20:Classe MainActivity

A classe *SplashActivity* (ver figura 4.21) é responsável por mostrar a tela enquanto o aplicativo está inicializando, essa tela apenas é mostrada quando o usuário abre a aplicação antes de ser carregada a tela *MainActivity*.

```java
public class SplashActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        new Handler().postDelayed(() -> {
                startActivity(new Intent(getBaseContext(), MainActivity.class));
                finish();
        }, delayMillis: 3000);

    }
}
```

Figura 4.21:Classe SplashActivity

4.3.2 Classes do projeto - Domínio

Dentro deste diretório estão classes que não fazem direcionamento para telas, nesta pasta estão definidas classes que prestam suporte para que as demais classes possam realizar os seus métodos.

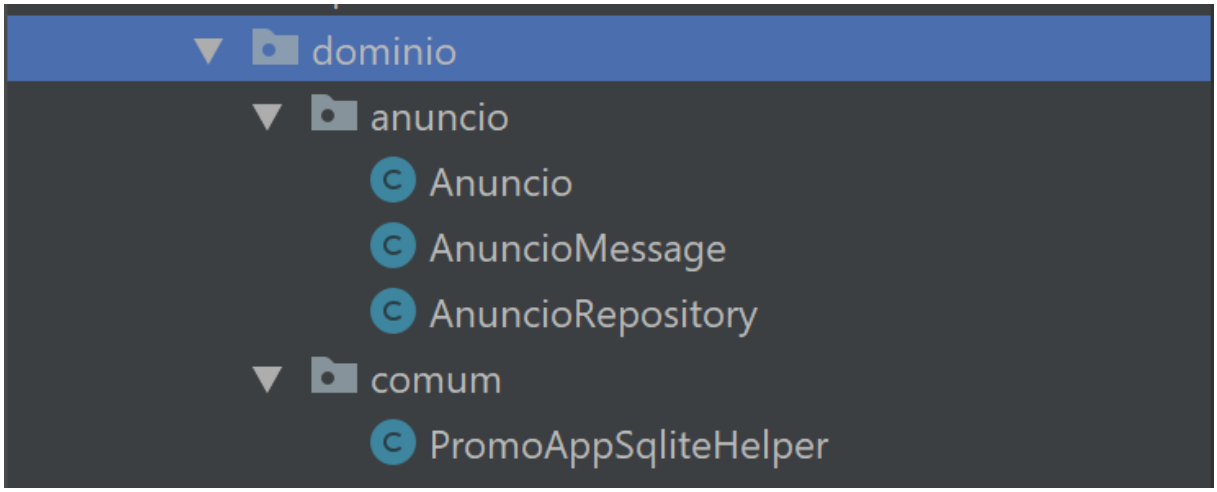

Figura 4.22:Pasta domínio

4.3.2.1 Classe Anuncio

A primeira classe é a *Anuncio*, nessa classe serão definidos os atributos que um objeto do tipo *Anuncio* deverá receber. (Ver figura 4.23)

```
public class Anuncio {

    private Long id;
    private String nome;
    private String descricao;
    private Double preco=0.0;
    private String url;
    private Date validade;

    public Anuncio() {
    }

    public Anuncio(Long id, String nome, String descricao, Double preco, String url, Date validade)
        this.id = id;
        this.nome = nome;
        this.descricao = descricao;
        this.preco = preco;
        this.url = url;
        this.validade = validade;
    }
```

Figura 4.23:Classe *Anuncio*

A classe Anuncio define os atributos ID, nome, descrição, preço, url e a validade, esses atributos serão recuperados quando o usuário criar um anúncio e colocar esses valores, conforme pode ser visto no construtor da classe, é necessário passar esses parâmetros para se obter um objeto do tipo anúncio antes de salvá-lo no banco de dados.

Ainda na classe Anuncio, o método *toString* é sobrescrito, assim é possível recuperar os campos do anúncio na *TelaOferta* (seção 4.3.1.3), apenas utilizando o método *toString*(). (Ver figura 4.24).

```
@Override
public String toString() {
        return "Anuncio{" +
                "nome='" + nome + '\'' +
                ", descricao='" + descricao + '\'' +
                ", preco=" + preco +
                ", url='" + url + '\'' +
                '}';
}
```

Figura 4.24:Método *toString* da classe Anuncio

4.3.2.2 Classe AnuncioMessage

Dentro da classe *AnuncioMessage* é definida a mensagem de anúncio a ser enviada ou recebida no Beacon (ver figura 4.25).

```
public class AnuncioMessage {
    private static final Gson gson = new Gson();

    private final String mUUID;
    private final Anuncio mAnuncio;

    public static Message newNearbyMessage(String instanceId, Anuncio anuncio) {
        AnuncioMessage anuncioMessage = new AnuncioMessage(instanceId, anuncio);
        return new Message(gson.toJson(anuncioMessage).getBytes(Charset.forName("UTF-8")));
    }
    public static AnuncioMessage fromNearbyMessage(Message message) {
        String nearbyMessageString = new String(message.getContent()).trim();

        return gson.fromJson(
                (new String(nearbyMessageString.getBytes(Charset.forName("UTF-8")))),
                AnuncioMessage.class);
    }
    private AnuncioMessage(String uuid, Anuncio anuncio) {
        mUUID = uuid;
        mAnuncio = anuncio;
    }
    public Anuncio getAnuncio() { return mAnuncio; }
```

Figura 4.25:Classe *AnuncioMessage*

O primeiro método *Message newNearbyMessage*() constrói um objeto do tipo mensagem já definido pela *API Neraby Messages* (seção 4.2) e identifica com um identificador único.

O segundo método *AnuncioMessage fromNearbyMessage*() é responsável por ler e guardar o *payload* da mensagem.

O terceiro método *AnuncioMessage*() é responsável por receber os valores da mensagem do segundo método e guardar na forma de um objeto do tipo Anuncio.

O quarto método *getAnuncio*() serve para que os métodos de outras classes possam recuperar ou enviar um anúncio.

4.3.2.3 Classe AnuncioRepository

Essa classe é responsável por manipular as tabelas do *SqLite* referentes ao anúncio, é na classe *AnuncioRepository* que estão os métodos que permitem criar, deletar e alterar um anúncio, nessa classe também estão os métodos que permitem recuperar os anúncios da tabela, seja para listá-los ou envia-los no Beacon ou até mesmo para edita-los através da recuperação por ID conforme visto em seções anteriores.

Nessa classe estão definidos dois métodos importantes, o primeiro método é o *criarAnuncio*() que permite a criação do anúncio a partir dos dados preenchidos no campo de texto pelo usuário, e o segundo método é o *salvar*() que permite salvar o anúncio criado no banco de dados. (Ver figura 4.26)

```
private Anuncio criarAnuncio(Cursor cursor) {
    Anuncio anuncio = new Anuncio();
    anuncio.setId(cursor.getLong( columnIndex: 0));
    anuncio.setNome(cursor.getString( columnIndex: 1));
    anuncio.setDescricao(cursor.getString( columnIndex: 2));
    anuncio.setPreco(cursor.getDouble( columnIndex: 3));
    anuncio.setUrl(cursor.getString( columnIndex: 4));
    anuncio.setValidade(new Date(cursor.getLong( columnIndex: 5)));
    return anuncio;
}

public void salvar(Anuncio anuncio) {
    ContentValues contentValues = new ContentValues();
    contentValues.put("NOME", anuncio.getNome());
    contentValues.put("DESCRICAO", anuncio.getDescricao());
    contentValues.put("PRECO", anuncio.getPreco());
    contentValues.put("URL", anuncio.getUrl());
    contentValues.put("VALIDADE", anuncio.getValidade().getTime());

    PromoAppSqliteHelper openHelper = new PromoAppSqliteHelper(this.context);
    SQLiteDatabase database = openHelper.getWritableDatabase();

    if (anuncio.getId() == null) {
        anuncio.setId(database.insert( table: "ANUNCIO",  nullColumnHack: null, contentValues));
    } else {
        database.update( table: "ANUNCIO", contentValues,  whereClause: "ID = ?",
                new String[]{anuncio.getId().toString()});
    }
```

Figura 4.26:Métodos *criarAnuncio*() e *salvar*()

Nessa classe também estão definidos os métodos que permitem que o usuário interaja com os anúncios já criados na tabela, o método *recuperarTodos*() recupera os anúncios da tabela e os agrupa em uma lista permitindo que o usuário selecione os anúncios que deseja enviar via beacon, o método *recuperarPorId*() permite que um anúncio específico seja recuperado da tabela seja para que uma ação possa ser realizada com esse anúncio. (Ver figura 4.27)

```
public List<Anuncio> recuperarTodos() {
    List<Anuncio> anuncios = new ArrayList<>();
    PromoAppSqliteHelper openHelper = new PromoAppSqliteHelper(this.context);
    SQLiteDatabase database = openHelper.getReadableDatabase();
    String[] campos = {"ID", "NOME", "DESCRICAO", "PRECO", "URL", "VALIDADE"};

    Cursor cursor = database.query( table: "ANUNCIO", campos, selection: null,
            selectionArgs: null, groupBy: null, having: null, orderBy: "VALIDADE DESC");

    if (cursor != null) {
        while (cursor.moveToNext()) {
            anuncios.add(criarAnuncio(cursor));
        }
    }

    database.close();
    return anuncios;
}

public Anuncio recuperarPorId(Long id) {
    Anuncio anuncio = null;
    PromoAppSqliteHelper openHelper = new PromoAppSqliteHelper(this.context);
    SQLiteDatabase database = openHelper.getReadableDatabase();
    String[] campos = {"ID", "NOME", "DESCRICAO", "PRECO", "URL", "VALIDADE"};

    Cursor cursor = database.query( table: "ANUNCIO", campos, selection: "ID = ?",
            new String[]{id.toString()}, groupBy: null, having: null, orderBy: "VALIDADE DESC");

    if (cursor != null) {
        while (cursor.moveToNext()) {
            anuncio = criarAnuncio(cursor);
        }
    }

    database.close();
    return anuncio;
}
```

Figura 4.27:Métodos *recuperarTodos*() e *recuperarPorId*()

E por último é na classe *AnuncioRepository* que é definido o método que permite deletar um anúncio da tabela do *SqLite*.(Ver figura 4.28)

```
public void deletar(Anuncio anuncio){

    PromoAppSqliteHelper openHelper = new PromoAppSqliteHelper(this.context);
    SQLiteDatabase database = openHelper.getWritableDatabase();


        database.delete( table: "ANUNCIO", whereClause: "ID = ?",
                new String[]{anuncio.getId().toString()});

    database.close();
}
```

Figura 4.28:Método *deletar*()

4.3.2.4 Classe PromoAppSqliteHelper

Essa classe é herdada da classe *SQLiteOpenHelper*, o seu propósito é gerenciar a criação do banco de dados e realizar o gerenciamento de versões.

```
public class PromoAppSqliteHelper extends SQLiteOpenHelper {

    public PromoAppSqliteHelper(Context context) { super(context, name: "promo-app.db", factory: null, version: 1); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        StringBuilder sqlBuilder = new StringBuilder();
        sqlBuilder.append("CREATE TABLE ANUNCIO (\n");
        sqlBuilder.append("    ID INTEGER PRIMARY KEY AUTOINCREMENT,\n");
        sqlBuilder.append("    NOME TEXT NOT NULL,\n");
        sqlBuilder.append("    DESCRICAO TEXT NOT NULL,\n");
        sqlBuilder.append("    PRECO REAL NULL,\n");
        sqlBuilder.append("    URL TEXT NOT NULL,\n");
        sqlBuilder.append("    VALIDADE INTEGER NOT NULL\n");
        sqlBuilder.append(")");
        db.execSQL(sqlBuilder.toString());

        sqlBuilder = new StringBuilder();
        sqlBuilder.append("CREATE TABLE ANUNCIO_RECEBIDO (\n");
        sqlBuilder.append("    ID INTEGER PRIMARY KEY AUTOINCREMENT,\n");
        sqlBuilder.append("    NOME TEXT NOT NULL,\n");
        sqlBuilder.append("    DESCRICAO TEXT NOT NULL,\n");
        sqlBuilder.append("    PRECO REAL NULL,\n");
        sqlBuilder.append("    URL TEXT NOT NULL,\n");
        sqlBuilder.append("    VALIDADE INTEGER NOT NULL\n");
        sqlBuilder.append(")");
        db.execSQL(sqlBuilder.toString());
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

Figura 4.29:Classe *PromoAppSqliteHelper*.

A classe *PromoAppSqliteHelper* é responsável por abrir o banco de dados caso o banco exista, criar o banco de dados caso não exista e atualizá-lo sempre que necessário.

4.4 Aparência do aplicativo

O aplicativo é baseado no conceito de telas. Cada botão da tela inicial é responsável por executar uma ação correspondente as seções 4.3.1.1 e 4.3.1.3.

Os botões da tela inicial são (ver figura 4.30):

-Anúncios (seção 4.3.1.1);
-Ofertas (seção 4.3.1.3);

Figura 4.30:Tela inicial do aplicativo

A Tela inicial consiste em 2 botões como pode ser observado na fig. 4.31, esses botões redirecionam o usuário às telas referentes.

```
ivity_main.xml | anuncio_detail.xml | anuncio_list.xml | anuncio_list_content.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/tela1"
    android:orientation="vertical"
    android:paddingLeft="64dp"
    android:paddingTop="16dp"
    android:paddingRight="64dp"
    android:paddingBottom="16dp"
    tools:context="com.github.promoapp.apresentacao.MainActivity">


    <Button
        android:id="@+id/buttonOfertas"
        android:layout_width="match_parent"
        android:layout_height="88dp"
        android:background="@color/AmareloClaro"
        android:onClick="openTelaOfertas"
        android:text="Ofertas" />

    <Button
        android:id="@+id/anunciosButton"
        android:layout_width="match_parent"
        android:layout_height="88dp"
        android:background="#FFFE91"
        android:onClick="exibirAnuncioListActivity"
        android:text="Anúncios" />


</LinearLayout>
```

Fig.4.31: Arquivo *.xml* da Tela Inicial

4.4.1 Tela Criar Anúncio

Ao clicar no botão para criar um anúncio, com o símbolo ‘+’, o usuário será direcionado para parte de cadastrar os produtos que deseja anunciar (ver figuras 4.31 e 4.32).

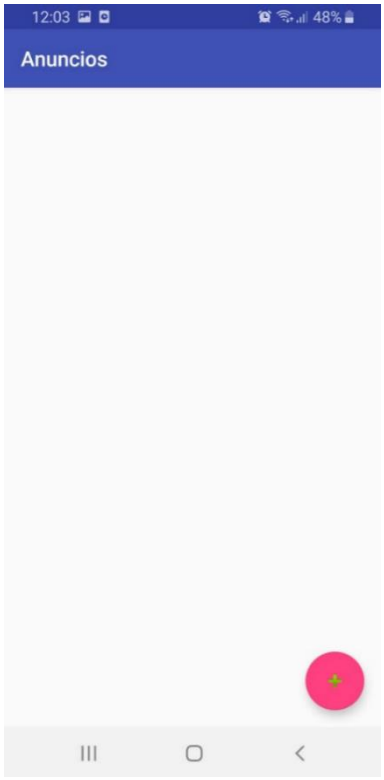

Figura 4.32: Tela de Criar Anúncio

Figura 4.33: Tela de Cadastrar o Anúncio

Para criar um anúncio o usuário deverá preencher os campos:

-Nome do Anúncio;
-Nome do Produto;
-Preço do Produto;
-Link do Produto;

Estes campos estão programados em um arquivo *.xml*, como pode ser visto na fig. 4.34. As entradas inseridas nestes campos são salvas para serem aplicadas na publicação presente na tela de ofertas.

```
anuncio_detail.xml | anuncio_list.xml | anuncio_list_content.xml | activ
5   xmlns:tools="http://schemas.android.com/tools"
6   android:id="@+id/anuncio_detail"
7   android:layout_width="match_parent"
8   android:layout_height="match_parent"
9   android:orientation="vertical"
10  android:padding="16dp"
11  tools:context=".apresentacao.anuncio.AnuncioDetailFragment">
12
13  <EditText
14      android:id="@+id/nomeText"
15      android:layout_width="match_parent"
16      android:layout_height="wrap_content"
17      android:ems="10"
18      android:hint="Nome do Anúncio"
19      android:inputType="textPersonName" />
20
21  <EditText
22      android:id="@+id/nomeProdutoText"
23      android:layout_width="match_parent"
24      android:layout_height="wrap_content"
25      android:ems="10"
26      android:hint="Nome do Produto:"
27      android:inputType="textPersonName" />
28
29  <EditText
30      android:id="@+id/precoProdutoText"
31      android:layout_width="match_parent"
32      android:layout_height="wrap_content"
33      android:ems="10"
34      android:hint="Preço do produto"
35      android:inputType="textPersonName" />
36
37  <EditText
38      android:id="@+id/linkProdutoText"
39      android:layout_width="match_parent"
40      android:layout_height="wrap_content"
41      android:ems="10"
```

Figura 4.34: Arquivo *.xml* do Cadastro de Anúncio

Após preencher esses dados e clicar no botão de salvar, o anúncio estará disponível no banco de dados de anúncios e pronto para ser publicado (ver figura 4.35).

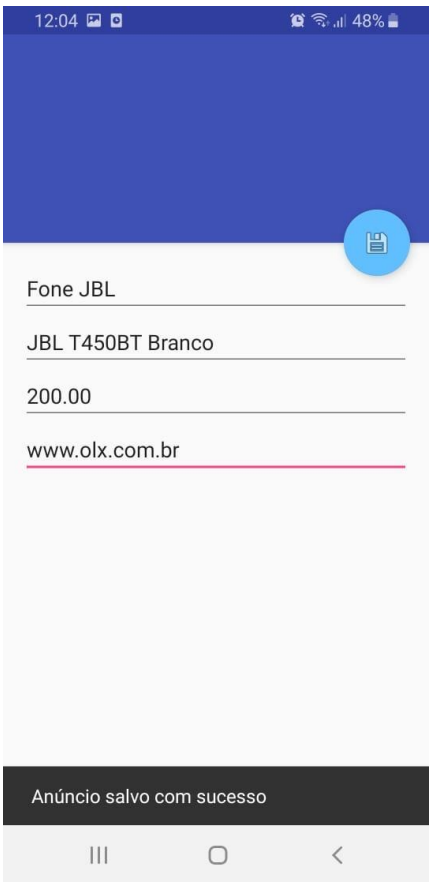

Figura 4.35: Tela de Anúncio Salvo

4.4.2 Tela Publicar Anúncio

Ao clicar no botão para anunciar o usuário poderá executar a ação de enviar os Beacons à celulares próximos (ver figura 4.34).

Figura 4.36: Tela Publicar Anúncio.

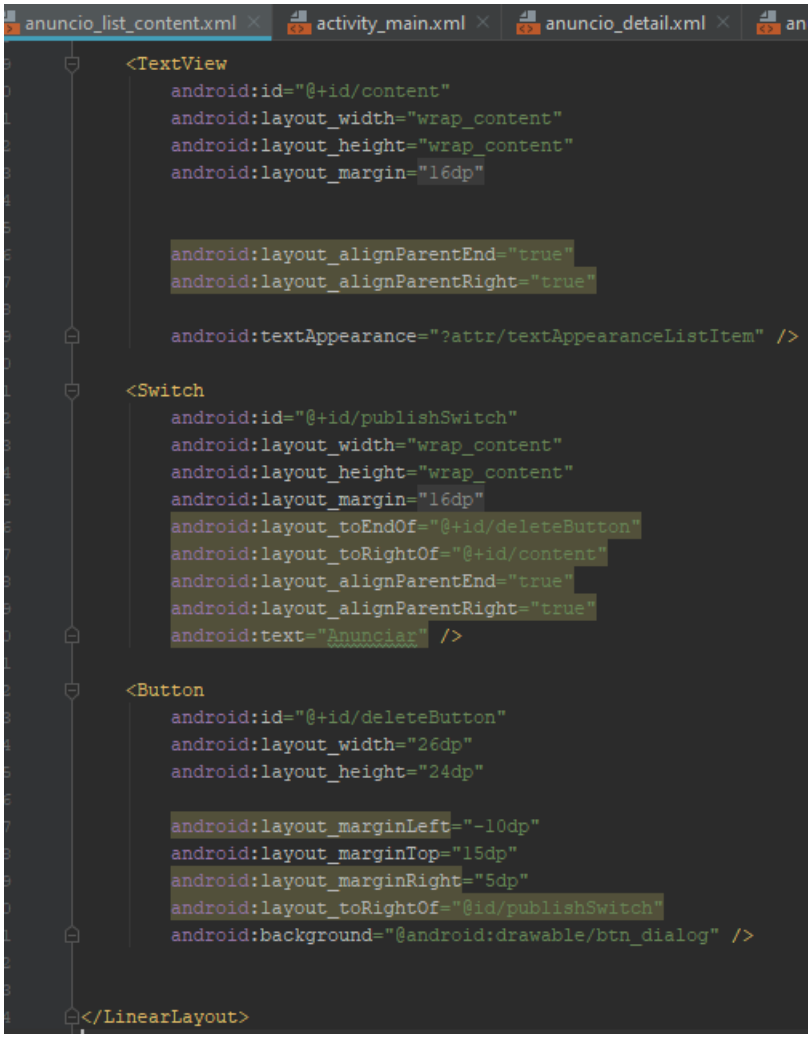

Figura 4.37: Arquivo .xml Listar Anúncio

Para listar os anúncios, cada anúncio salvo é submetido a um formato padrão que consiste de um título, um switch para ativar a publicação e um botão de deletar. O layout pode ser observado na fig. 4.37, *TextView* para o título, Switch padrão para publicação e Button para o botão de deletar.

Nessa tela fig. (4.36) o usuário deverá selecionar os anúncios que deseja publicar, e em seguida ativar o "*toggle* switch: Anunciar" para começar a publicar. Vale ressaltar que enquanto o *toggle* estiver ativo não será possível modificar as publicações que estão sendo feitas, sendo necessário desativar o *toggle* e modificar os anúncios a serem publicados.

4.4.3 Tela Ofertas

Ao clicar no botão da tela inicial Ofertas (figura 4.35) o usuário poderá verificar os anúncios que estão sendo publicados ao seu redor limitado ao alcance da tecnologia Bluetooth Low Energy.

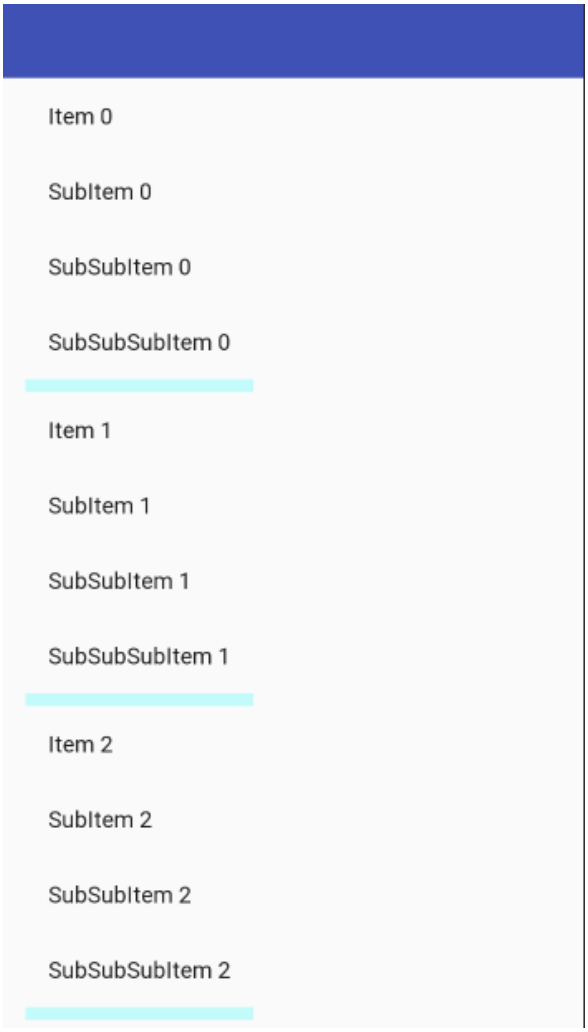

Figura 4.38: Tela Ofertas.

4.5 Testes do aplicativo

Após finalizada a programação do aplicativo foram realizados diversos testes em diferentes modelos de telefone celular dentre eles: Samsung Galaxy A10 , Asus Zen Phone 5, Xiomi redmi 7, LG K10 e Huawei P20 pro.

Após o usuário permitir o uso de sua localização e do Bluetooth no dispositivo, o aplicativo realizou tanto o anúncio dos Beacons quanto a subscrição de Beacons, porém com algumas pequenas variações no alcance de acordo com cada dispositivo.

Infelizmente não foi possível fazer com que o aplicativo fosse executado em segunda plano, fazendo assim com que o usuário se mantenha na tela de anúncio para poder realizar o anúncio de Beacons ou na tela de ofertas para fazer a subscrição de Beacons. Está sendo estudada uma tratativa para esse problema para ser implementada em uma versão futura do aplicativo.

5 Conclusão e Trabalhos Futuros

Este trabalho apresenta um método ainda novo no mercado através da utilização de mensagens Beacon e tecnologia BLE para aprimorar ainda mais as estratégias de publicidade por meio do uso do Geomarketing. O uso de BLE e Beacons é notável, pois eles podem ser usados para inovar várias áreas como comércio, trânsito, soluções domésticas e outras áreas de aplicações internas, com os benefícios de um gasto reduzido de bateria e a possibilidade de várias conexões simultâneas.

O uso dessas tecnologias ainda tem muitos desafios em seu horizonte, porém, esta aplicação e suas funcionalidades servem como um primeiro passo para alcançar os objetivos futuros do setor de Geomarketing, mudando consideravelmente a maneira como o consumidor e o vendedor se comunicam.

Para o futuro, pretendemos continuar trabalhando na solução de Geomarketing e avançaremos com o aprimoramento do software. Após ajustes e feedback dos usuários, pretendemos testar a aplicação em um cenário real de mercado, analisaremos dados de compras usando Big Data e IA (Inteligência Artificial), como a Análise de sentimentos, para oferecer aos compradores em tempo real promoções e ofertas personalizadas.

Referências Bibliográficas

[1] Developer Android. **Android**. Disponível em: <https://developer.android.com>. Acesso em:23/05/2019

[2] Developers Google. **Nearby Google**. Disponível em: <https://developers.google.com/nearby/messages/android/>. Acesso em 23/05/2019

[3] Developers Google. **Nearby Messages**. Disponível em: <https://developers.google.com/android/reference/com/google/android/gms/nearby/messages/Strategy >. Acesso em 23/05/2019

[4] Gradle Documentation. **Gradle**. Disponível em: <https://docs.gradle.org/current/userguide/what_is_gradle.html#what_is_gradle>. Acesso em: 23/05/2019

[5] UFRJ. **Bluetooth Low Energy**. Disponível em: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2012_2/bluetooth/ble.htm>. Acesso em 30/05/2019.

[6] Beaconstac. **Bluetooth Beacon**. Disponível em: <https://www.beaconstac.com/what-is-a-bluetooth-beacon>. Acesso em 02/06/2019

[7] Dalal Zaim. **Bluetooth Low Energy (BLE) Based Geomarketing System.** Disponível em: <https://ieeexplore.ieee.org/document/7772263 >. Acesso em 10/06/19.

[8] Meera Radhakrishnan, Archan Misra, Rajesh Krishna Balan, and Youngki Lee**. Smartphones & BLE Services: Empirical Insights.** Disponível em: <http://smedia.ust.hk/james/projects/people_aware_smart_city_applications/paper/3.pdf>. Acesso em 20/06/2019.

[9] Mostafa Bellafkih. **Developing a Geomarketing Solution.** Disponível em: <https://www.sciencedirect.com/>. Acesso em 14/06/19.

[10] Mohammad Afaneh. **INTRO TO BLUETOOTH LOW ENERGY.** Disponibilizado pela professora Cláudia Barenco. Disponível para compra em: < https://www.amazon.com/Intro-Bluetooth-Low-Energy-easiest/dp/1790198151/ref=tmm_pap_swatch_0?_encoding=UTF8&qid=&sr= >. Acesso em 23/06/19.

[11] You-Wei and Chi-Yi Lin. **An Interactive Real-Time Locating System Based on Bluetooth Low-Energy Beacon Network.** Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5981856/ >. Acesso em 24/06/19

[12] Beaconstac. **What is Bluetooth Marketing.** Disponível em: < https://www.beaconstac.com/bluetooth-marketing>. Acesso em 01/09/19.

[13] iBeacon. **What is iBeacon.** Disponível em: < http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/>. Acesso em 01/09/19.

[14] Developers Google. **Eddystone**. Disponível em: <https://developers.google.com/beacons/eddystone>. Acesso em 01/09/2019

[15] URIBeacon. **Getting Started with URIBeacon**. Disponível em: <https://learn.adafruit.com/google-physical-web-uribeacon-with-the-bluefruit-le-friend >. Acesso em 02/09/2019

[16] Gradle. **Gradle Build Tool.** Disponível em: < https://gradle.org/>. Acesso em 02/09/19.

[17] Estimote. **Estimote Beacon.** Disponível em: < https://estimote.com/about/>. Acesso em 12/10/19.

[18] Android database sqlite. **SQLite Database Management.** Disponível em: <https://developer.android.com/reference/android/database/sqlite/package-summary/>. Acesso em

10/08/2019

[19] Snackbar SqLite. **Snackbar for android.** Disponível em: <https://developer.android.com/reference/android/support/design/widget/Snackbar> .Acesso em 10/08/2019

[20] Classe SqLiteOpenHelper. SqLiteOpenHelper. Diponível em:< https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper> . Acesso em 11/08/2019

[21] COUTINHO, Gustavo Leuzinger. A era dos smartphones: um estudo exploratório sobre o uso dos smartphones no Brasil. 2014. 60 f., il. Monografia (Bacharelado em Comunicação Social)— Universidade de Brasília, Brasília, 2014.

Anexo I

O Artigo abaixo foi submetido à conferência ICMarktech'19 - Maia, Portugal e foi aceito para publicação na revista Springer, série SIST - Smart Innovation, Systems and Technologies em 2020.

Geomarketing based on Beacons BLE

Isaac Aleixo[1], João Paulo Fernandes Bezerra[2],Cláudia Jacy Barenco Abbas[3]
Faculty of Technology
Department of Electrical Engineering
Campus Universitário Darcy Ribeiro
Brasília – D.F. - Brazil
isaac.aleixo@gmail.com[1],joaopaulofb83@gmail.com[2],barenco@unb.br[3]

Abstract. The breakthrough in the field of wireless technologies, such as Bluetooth Low Energy (BLE), the availability of large volume of customer data and the growing use of beacon messages, all of these contribute to the development of a solution that makes shopping more enjoyable and efficient for the customer with the real time monitoring facilitating the business management. The objective of this paper is to present a BLE solution using beacon technology applied to Geomarketing. The study addresses the construction of an application that makes use of BLE technology.

Keywords: Geomarketing; Beacons; Bluetooth Low Energy.

1. Introduction

The purpose of this work is to develop a Geomarketing solution: an application where the user will be able to advertise and get ads based on their location. To understand the goals of the application it is necessary to understand the following key concepts: BLE – Bluetooth Low Energy; Beacons; and Geomarketing that will be described in the following sub sections.

1.1 Bluetooth Low Energy

BLE stands for Bluetooth Low Energy [1], BLE is a form of wireless communication designed especially for short-range communications. The BLE is intended for situations where battery life is preferred over high data transfer rates. For example, if a user wants to broadcast marketing campaigns in the vicinity of a phone, the amount of data that is required to transfer to another is extremely small so BLE-compliant beacons do the job quickly without depleting the battery.

A BLE beacon transmits packets of data at regular intervals; these packets of data are detected by applications or services pre-installed on nearby smartphones. This BLE communication triggers actions such as sending a message, promoting an application, or sending an advertisement.

To save energy and provide higher data transfer speeds, the entire BLE communication structure consists of 40 frequency channels, separated by 2MHz. 3 of these channels are the main advertising channels, while the remaining

37 channels are secondary channels, also known as data channels. Bluetooth communication starts with the first three advertisement channels and transfer data on secondary channels (see figure 1).

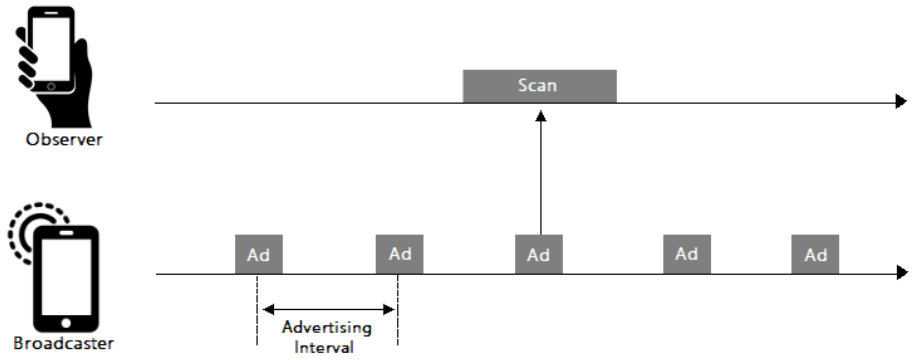

Figure 1. BLE communication example [1]

1.2 Beacons BLE

Beacons are small Bluetooth transmitters [2], they transmit a unique identifier to electronic devices close enough to pick up their signal.

Beacons transmit small amounts of data via BLE packets, therefore having distance limitations, as a result, are often used for indoor localization technology, although beacons can also be used in outdoor environments as well.

The unique identifier and multiple bytes that are transmitted by the beacon can be used to determine the physical location of the device, track customers, or trigger a location-based action on the device, such as a notification when it arrives at a location.

Firstly, it is important to inform that beacons are used in BLE precisely because they allow for better energy savings and because they have periodic data transfer, not a continuous flow. These features make BLE ideal for broadcasting Beacons and advertisements. Therefore, it is ideal for the beacon-based Geomarketing project.

Beacons are small packets transmitted via BLE signals at a given interval. This range depends on the hardware capacity. On average, a BLE beacon can transmit BLE signals up to 80 meters. This beacon BLE signal is able to perform a specific action relevant to the location, such as showing an ad.

Other advertising methods are also possible with Google's beacons, URIBeacon, and Eddystone allow for a URI transmission mode. URI beacons transmit a URI that can be a link to a web page and the user will see that URI directly on their phone.

The beacons send an ID number through the BLE channels, approximately 10 times per second. A Bluetooth enabled nearby beacon handles this ID number. When an application or service pre-installed such as Google Nearby recognizes the ID number, it binds the beacon to an action, such as downloading an application or a marketing offer, and displays it on your smartphone.

1.3 Geomarketing and Marketing with Beacons

Geomarketing is a marketing technique that relies on user location to define target audiences and better tactics to reach the end consumer at the right time and place [3]. Coupled with the increase in the use of mobile devices, tablets, cell phones and other devices, the information of the location and places visited by a person, if used efficiently by an algorithm or AI (Artificial Intelligence), can lend companies valuable information to assemble a strategy of effective marketing, delivering ads and offers that are more relevant to your end customer.

BLE Beacons are a convenient way for marketers to communicate with customers in a meaningful and personalized manner.

According to Beaconstac [4], social media marketing, such as Facebook and Twitter ads, also has below-average click-through rates for beacons. The average click-through rate for Facebook ads in 2018 is 0.119%, while the

average click-through rate for Twitter ads is 1% to 3%. However, the Average Click-Through Rate for beacon campaigns is 2-4%. Well-executed signaling campaigns can generate a rate of 12-15%.

With this data it is possible to notice that the technology of beacons used in ads is evolving rapidly.

1.4 Differences between Bluetooth Classic and BLE

There are two main technologies within the specification Bluetooth: Bluetooth Classic and BLE also known as Bluetooth Smart. The main difference between the two is in the energy consumption in each case. However, there are other factors why Bluetooth Smart is being used for technological applications [5].

Bluetooth Classic consumes more power than BLE. Bluetooth Classic is ideal for applications that require continuous data flow, such as headphones. However, BLE is suitable for applications that work well with periodic data transfer, and therefore reduces a significant amount of battery usage. This makes BLE suitable for applications related to proximity marketing, which is the case with the Beacons in this project.

BLE can establish up to 20 connections simultaneously, it supports more simultaneous connections as it transfers small data packets and establishes fast connections. The Bluetooth Classic can only initiate 7 simultaneous connections.

2. Developing the Application

The application was developed using Java programming language and is limited for Android operational systems. To understand how the application was developed, it is necessary to comprehend the core tools used to

develop the application: Android Studio; Google Developer Console; API Nearby Messages, that are shown in the following sub sections.

2.1 API Nearby Messages

The API used for building the application is the Google Nearby Messages API [6] The API makes use of both Bluetooth, BLE and Wi-Fi to make devices communicate, the server makes it easy to exchange messages between devices with the same pairing type encoding.

The exchange of messages between devices takes place as follows:

I. A publishing application makes a request to associate the message with a unique pairing code (token).

II. The publishing device uses a combination of Bluetooth, Bluetooth Low Energy and Wi-Fi to make the token discoverable by nearby devices. The publishing device also uses these technologies to search for tokens from other devices.

III. A subscription application associates its signature with a token and uses a combination of the above technologies to send its token to the publishing device and to detect the publishing device token.

IV. When one side detects the token of the other, it informs the server.

V. The server makes it easy to exchange messages between two devices when both are associated with a common token, and the API keys used by the calling applications are associated with the same project in the Google Developers Console.

In order to integrate the application with Google Play services, you need to create a buildGoogleApiClient () method, both in the class responsible for publishing and in the class responsible for subscribing (see figure 2).

```
private void buildGoogleApiClient() {
    if (mGoogleApiClient != null) {
        return;
    }
    mGoogleApiClient = new GoogleApiClient.Builder( context: this)
            .addApi(Nearby.MESSAGES_API, new MessagesOptions.Builder()
                    .setPermissions(NearbyPermissions.BLE).build())
            .addConnectionCallbacks(this)
            .enableAutoManage( fragmentActivity: this,  onConnectionFailedListener: this)
            .build();
}
```

Figure 2. buildGoogleApiClient() Method.

2.2 Google Developers Console

To use the API Nearby Messages you must have a Google account, but for the end user of the application is not necessary to have an account.

To use the API it is necessary to create a KEY API in the Google Developers Console [5], the KEY API is also used to restrict which applications can access the beacons with the messages published by the publishers. In this solution only the applications with the same KEY API can view the beacons.

The KEY API is configured in the AndroidManifest.xml file.

2.3 Publishing and Subscribing with Beacons

First, to make sure that only beacons will be published and subscribed, it is necessary to configure the strategy, the Google Messages API provides 2 strategies:

- Default: Default behavior is currently making publications and subscriptions, using all available sensors, to discover nearby devices regardless of distance.

-BLE_Only: The behavior of BLE Only is to make publications and subscriptions only using beacons, this is the strategy to be used for the application and to study the properties of beacons and BLE.

Once the strategy to be used has been chosen, you can now create the methods to publish and subscribe the beacons, the methods have been created in separate classes, a java class is responsible for publishing beacons and another java class responsible for beacons subscription.

Within the folder "java" were created the classes: "TelaAnuncio" which stands for AdvertisingScreen and "TelaOferta" which stands for OfferScreen responsible for publishing and subscribing respectively.

Within the class the method private void publish () is created, within the publishing options there is the parameter .setStrategy, within the Strategy class of the Nearby Messages API, is the constructor for BLE ONLY.

And at the very beginning of the "TelaAnuncio" class, a variable is created to store this strategy (see figure 3).

```
private static final Strategy PUB_SUB_STRATEGY = (new Strategy.Builder()).zzjS(2)
        .setTtlSeconds(TTL_IN_SECONDS).build();
```

Figure 3. Variable created to store Strategy.

With this done it is possible to create the publish () method, with the adopted strategy parameter (see figure 4).

```
private void publish() {
    Log.i(TAG, msg: "Publishing");
    PublishOptions options = new PublishOptions.Builder()
            .setStrategy(PUB_SUB_STRATEGY)
            .setCallback(onExpired() → {
                    super.onExpired();
                    Log.i(TAG, msg: "No longer publishing");
                    runOnUiThread(() → {
                            mPublishSwitch.setChecked(false);
                    });
            }).build();

    Nearby.Messages.publish(mGoogleApiClient, mPubMessage, options)
            .setResultCallback((ResultCallback) (status) → {
                    if (status.isSuccess()) {
                        Log.i(TAG, msg: "Published successfully.");
                    } else {
                        logAndShowSnackbar( text: "Could not publish, status = " + status);
                        mPublishSwitch.setChecked(false);
                    }
            });
}
```

Figure 4. Publish method.

To subscribe, you must first have a publisher in the range of the device, and if this condition is fulfilled, the device you want to subscribe to will perform the subscribe method, that is in the class "TelaOferta" which stands for OfferScreen.

Just as in the case of publication it is necessary to define the strategy adopted for the subscription, and as previously only interested in subscribing in beacons, so the strategy used remains BLE ONLY.

In figure 5 there is an application diagram showing how each of the steps mentioned works.

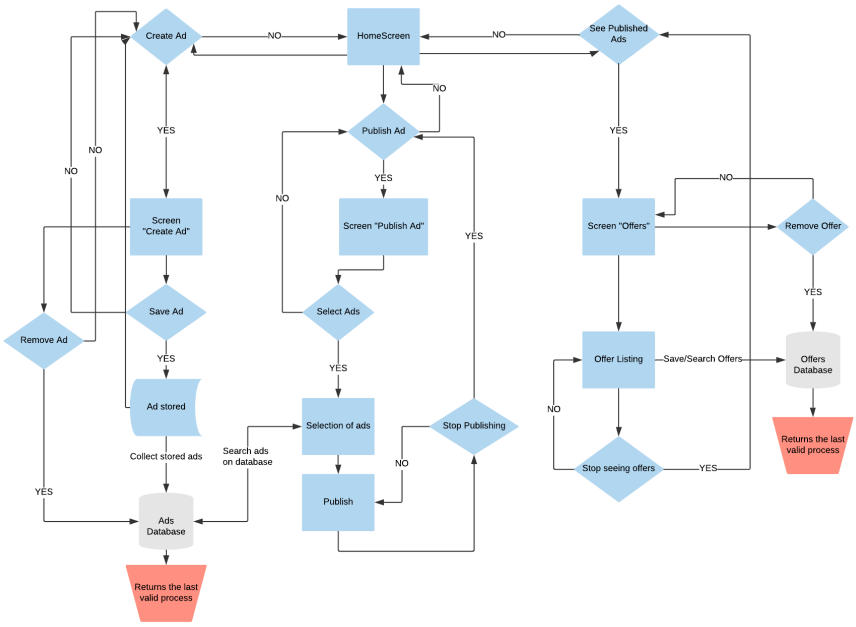

Figure 5. Application Flowchart.

From the home screen you can make 3 main decisions: Create an ad; Publish ad; or View published ads.

2.4 Create Advertisement

When the application user chooses to create an advertisement, they will be taken to the screen “TelaCriaAnuncio” which stands for “CreateAd”. In this class the methods responsible for creating, storing and deleting an advertisement are defined.

2.5 Publish Advertisement

Before publishing, the application will send to the Google Developer Console the API KEY information, after this step it will be possible to publish to other applications that also have the same validated API KEY.

When the user of the application selects the option to publish they will be taken to the screen “TelaAnuncio” which stands for “AdScreen”. In this class are defined the methods responsible for selecting the ads to be published, the publication of the selected ads and to stop publishing.

In order to make the announcement, the user must select which ads created and stored in the ad database they want to post. If they want to start a new post with different ads, the current post must be stopped using the unpublish () method of the class, and a new post should be made with the desired announcements. From this screen the user can stop publishing at any time using the unpublish () method.

2.6 View Published Ads

Before you begin viewing served ads, the application will send the Google Developer Console API KEY information, after which you can subscribe to other applications that also have the same validated API KEY.

When the user selects the option to view the ads that are being published, they will be taken to the screen “TelaOferta” which stands for “OfferScreen”, in this class are defined the methods responsible for listing and

saving the ads that have been published by devices in which the application has been subscribed, as well as methods for deleting offers from the offer database and also stopping the subscription.

When subscribe is selected, the application will automatically start a scan of devices that are publishing within the range supported by the physical limitations of the device, up to 80 meters. Upon finding a device in range of the application will then store all ads sent by the advertiser's beacon into an offer database.

The offers database is used to group offers by categories, so users can filter ads according to their needs as well as delete offers that do not attract their attention.

If the user wishes to receive ads, they may simply confirm the subscription through the unsubscribe () method.

2.7 App Appearance

Each home screen button is responsible for performing an action corresponding to sub sections 2.4, 2.5 and 2.6 described before (see figures 6 and 7).

The home screen buttons are:

-Criar Anúncio “CreateAd” (section 2.4);

-Anunciar “Advertise” (section 2.5);

-Ofertas “Offers” (section 2.6);

Figure 6. Home Screen

Figure 7. Screen “Create Ad”

2.8 Screen “CreateAd”

By clicking on the button “Create an Ad” the user can perform the actions explained in sub section 2.4.

To create an ad the user must fill in the fields:

-Ad Name; (Nome do Anúncio)

-Product's name; (Nome do Produto)

-Price of the product; (Preço do Produto)

-Link of the Product; (Link do Produto)

After filling in this data and clicking the save button, your ad will be available in the ad database and ready to be published.

2.9 Screen “PublishAd”

By clicking the button “Announce” the user can perform the actions explained in the sub section 2.5.

On this screen the user must select the ads they want to post, and then activate the “toggle switch: Announce” to start posting. It is noteworthy that while toggle is active it will not be possible to modify the publications being

made, in the case the user desires to make any alterations, they must then disable the toggle and modify the ads to be published.

2.10 Screen “Offers”

By clicking on the button to view the offers the user can perform the actions explained in sub section 2.6.
From this screen you can view offers being published and store them in the offer database, as well as manage the offers saved in the database.

3. Conclusion

This paper presents a powerful method of employing beacon messages and BLE technology to further enhance advertisement strategies through the use of Geo-marketing. The use of BLE and beacons is notable, as they can be used to innovate in multiple areas, such as in those of commerce, transit, home solutions and other indoor applications, with the benefits of reduced battery expenditure and the possibility for multiple simultaneous connections.

The use of these technologies has yet many challenges in its horizon, however, this application and its functions serve as a first step in accomplishing future goals of the geo-marketing sector, notably changing the way the consumer and seller dynamic functions.

This paper shows a powerful solution for Geomarketing using beacons messages and BLE technology.

BLE and beacons can be used to innovate on a lot of areas such as selling, car traffic, home solutions and others indoor applications.

There are a lot of challenges on future, and this work is only one step to accomplish the future goals in the Geomarketing area that is changing the way of the relationship between consumer and dealer.

References

1. UFRJ. Bluetooth Low Energy. Retrieved from: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2012_2/bluetooth/ble.htm>.
2. Beaconstac. Bluetooth Beacon. Retrieved from: <https://www.beaconstac.com/what-is-a-bluetooth-beacon>.
3. Dalal Zaim. Bluetooth Low Energy (BLE) Based Geomarketing System. Retrieved from: https://ieeexplore.ieee.org/document/7772263
4. Mostafa Bellafkih. Developing a Geomarketing Solution. Retrieved from: <https://www.sciencedirect.com/>.
5. Meera Radhakrishnan, Archan Misra, Rajesh Krishna Balan, and Youngki Lee. Smartphones & BLE Services: Empirical Insights. Retrieved from: http://smedia.ust.hk/james/projects/people_aware_smart_city_applications/paper/3.pdf
6. Developer Android. Android. Retrieved from: <https://developer.android.com>.
7. Developers Google. Nearby Google. Retrieved from: <https://developers.google.com/nearby/messages/android/>