UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA ESPECIALIZAÇÃO
EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

GUILHERME PEREIRA DE RESENDE

DESENVOLVIMENTO DE UMA APLICAÇÃO MÓVEL ANDROID PARA VISUALIZAÇÃO DE INFORMAÇÕES COM IDENTIFICAÇÃO VIA BLUETOOTH

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA

2018

GUILHERME PEREIRA DE RESENDE

DESENVOLVIMENTO DE UMA APLICAÇÃO MÓVEL ANDROID PARA VISUALIZAÇÃO DE INFORMAÇÕES COM IDENTIFICAÇÃO VIA BLUETOOTH

Monografia apresentada ao Departamento Acadêmico de Informática Especialização em Desenvolvimento para Dispositivos Móveis da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Especialista em Desenvolvimento para Dispositivos Móveis"

Orientador: Prof. Dr. Paulo Cezar Stadzisz

CURITIBA

2018

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

"Desenvolvimento de uma Aplicação Móvel Android para Visualização de Informações com Identificação via Bluetooth"

por

"Guilherme Pereira de Resende"

Este Trabalho de Conclusão de Curso foi apresentado às 18:10 do dia 20 de fevereiro de 2018 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

Prof. Paulo Cézar Stadzisz **(Presidente/Orientador – UTFPR/Curitiba)**	Profa. Maria Claudia Figueiredo Pereira Emer **(Avaliador 1 – UTFPR/Curitiba)**
Prof. Robson Ribeiro Linhares **(Avaliador 2 – UTFPR/Curitiba)**	

"A Ata de Aprovação assinada encontra-se na Coordenação do Curso."

AGRADECIMENTOS

Agradeço à Deus primeiramente, por Sua presença constante em minha vida. Agradeço aos meus pais por cada momento de apoio e motivação, à minha esposa Caroline por todo seu amor e compreensão, e às minhas filhas Julia e Giovanna. Por fim, agradeço ao meu orientador Prof. Dr. Paulo Cezar Stadzisz por todo o empenho e atenção durante o desenvolvimento do trabalho.

RESUMO

RESENDE, G.P.. Desenvolvimento de uma Aplicação Móvel Android para Visualização de Informações com Identificação via Bluetooth. 32 f. Monografia – Departamento Acadêmico de Informática Especialização em Desenvolvimento para Dispositivos Móveis, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

Este trabalho descreve o projeto e o desenvolvimento de um aplicativo móvel para viabilizar a identificação de objetos fazendo uso dos recursos de comunicação da tecnologia *bluetooth 4.0*, disponível nos *smartphones* mais avançados. A solução também utiliza serviço na web (*web-service*) que disponibiliza os detalhes dos objetos detectados à medida em que forem detectados pelo dispositivo móvel.

Palavras-chave: Bluetooth LE, Android, Webservice, Aplicação Móvel

ABSTRACT

RESENDE, G.P.. Mobile application development to identify bluetooth information. 32 f. Monografia – Departamento Acadêmico de Informática Especialização em Desenvolvimento para Dispositivos Móveis, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

This work describes the development of a mobile application to be used for the objects identification based on the Bluetooth 4.0 technology, which is available in the recent smartphones. The solution also uses a web service to provide the object details during the detection of the objects by the application.

Keywords: Bluetooth LE, Android, Webservice, Mobile Application

LISTA DE FIGURAS

LISTA DE SIGLAS

BLE	Bluetooth Low Energy
BR/EDR	Basic Rate / Enhanced Data Rate
P2P	point-to-point
GATT	Generic Attribute
GAP	Generic Access Profile
URL	Uniform Resource Locator
IDC	International Data Corporation
OHA	Open Handset Alliance
SDK	Software Development Kit
API	Application Programming Interface
IDE	Integrated Development Environment

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO DO TRABALHO

Os *smartphones* estão se tornando cada vez mais indispensáveis na vida moderna, e sua utilização e disseminação entre as pessoas tem crescido de forma acentuada. Através dos aplicativos dos *smartphones* os usuários tem acesso à uma infinidade de informações de forma extremamente rápida e precisa. Os serviços mais utilizados atualmente estão relacionados às redes sociais, soluções de localização por GPS, acesso à rede bancária, sem contar os aplicativos para entretenimento em geral como jogos e acesso aos conteúdos de música e vídeo. Mas há ainda muitas tecnologias e soluções a serem exploradas no mundo dos *smartphones*, e uma delas é a possibilidade de utilização da tecnologia Bluetooth não somente para comunicação mas também para localização de objetos ou produtos.

Este trabalho de conclusão de curso de especialização apresenta o projeto e o desenvolvimento uma plataforma capaz de auxiliar os usuários na tarefa de localização, identificação e fácil obtenção de informações de determinados objetos utilizando a tecnologia *Bluetooh Low Energy* (BLE).

1.2 OBJETIVOS

O objetivo geral deste trabalho é desenvolver uma aplicação móvel para identificar objetos com a tecnologia de comunicação por *Bluetooh LE*. Os objetivos específicos são:

- Ampliar o conhecimento sobre a tecnologia de comunicação *Bluetooh LE*;
- Projetar e implementar uma aplicação móvel para localização de objetos;
- Projetar e implementar um protótipo de um *Web Service* para disponibilizar as informações detalhadas dos objetos;
- Avaliar o sistema em operação;

1.3 MOTIVAÇÕES

A principal motivação para o desenvolvimento deste trabalho foi estudar e entender um pouco mais tecnologia *Bluetooth* no modo *iBeacon*, e que tem um grande potencial a ser explorado. Algumas empresas de tecnologia já estão lançando produtos comerciais utilizando essa tecnologia, a seguir serão apresentadas algumas das primeiras aplicações práticas da utilização do *iBeacon* em diversos setores:

1) Lojas de varejo: Os *iBeacons* já estão cada vez mais presentes nas lojas de varejo dos Estados Unidos, sendo utilizadas para aumentar a interação entre os clientes e os produtos expostos nas lojas. O objetivo é identificar qual é o departamento que o cliente demonstra maior interesse e enviar notificações customizadas ou promoções diretamente em seu celular. Recentemente o *Walmart* atualizou seu aplicativo para dar suporte à esses recursos.

2) Atrações turísticas: Uma das primeiras aplicações práticas dessa tecnologia foi a utilização em museus e exposições para fornecer informações específicas de obras para proporcionar uma nova experiência ao visitante, fazendo com que informações relevantes sejam mostradas no celular exatamente no momento em que o visitante se aproxima de determinadas obras. Com a facilidade de visualização das informações no idioma do visitante.

3) Educação: Um projeto piloto foi implantado em uma universidade para registrar de forma automática a presença em aula dos alunos. Neste caso o aplicativo disponibiliza as notas aos alunos e registra em quais salas os alunos estão e o tempo de permanência em cada ambiente.

4) Industria Hospitalar: Diversas são as aplicações nos ambientes hospitalares, mas uma das mais interessantes é sem dúvida a substituição do prontuário médico em papel pelo aplicativo baseado em *iBeacons*. O Hospital Universitário de Lausanne na Suíça foi o pioneiro. Neste caso, as informações de cada paciente e seu prontuário são atribuídos a um *iBeacon*, dessa maneira cada quarto do hospital deve ter instalado um dispositivo emissor de sinais *iBeacon*. Assim, sempre que os médicos e enfermeiros forem fazer uma visita, eles tem acesso ao dados dos pacientes e seu prontuário médico de forma rápida e automática, sem a necessidade de realizar buscas nos bancos de dados manualmente.

5) Indústria do entretenimento: Há aplicações específicas para utilização em cinemas, por exemplo, onde os clientes recebem em seus celulares as informações referentes aos filmes que estão em cartaz com acesso ao *trailler*, curiosidades, preços e horários, simplesmente ao passar próximo da bilheteria do cinema. Em boates, os frequentadores podem receber a programação do dia assim que passarem na frente do estabelecimento, inclusive com promoções exclusivas através do aplicativo.

6) Turismo: Em uma estação de trem, os passageiros podem receber informações sobre todas as linhas de trens, horários, atrasos, ou até mesmo o nome e localização geográfica das estações em que o trem se encontra. O conceito de cidades inteligentes tem se tornado cada vez mais popularmente difundido com o surgimento do *iBeacon*. Diversas cidades bem desenvolvidas tem aprimorado a experiência dos turistas, criando uma rede de *iBeacons* em locais estratégicos, para fornecer as informações mais relevantes no momento mais oportuno de forma bastante simples e rápida.

7) Setor Imobiliário: A ideia consiste em criar uma outra forma de promover os imóveis que se encontram para alugar ou vender. Com o uso do aplicativo, quando o usuário estiver buscando imóveis em uma região de interesse, ao passar em frente de uma casa cadastrada no sistema, o usuário deve receber todas as informações, fotos, preço, entre outros. Basta instalar um dispositivo emissor de *iBeacon* junto da placa que faz o anúncio de venda.

1.4 ESTRUTURA DO TRABALHO

No Capítulo 2 são abordados assuntos relacionados com as tecnologias envolvidas no tema, incluindo o *Bluetooth 4.0* e seus modos de funcionamento, os dispositivos móveis, a tecnologia *Android*, e ainda o banco de dados *SQLite*. O Capítulo 3 descreve a proposta do trabalho de forma mais detalhada. A especificação e as fases do desenvolvimento são mostrados no Capítulo 4. Em seguida, no Capítulo 5, são abordados os testes funcionais da aplicação final. Por fim, no Capítulo 6 são apresentadas as conclusões e propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve os fundamentos para o trabalho desenvolvido. Inicialmente, a tecnologia *Bluetooth 4.0* é apresentada, assim como seus principais modos de operação e topologias de funcionamento. São abordados, também, os temas relacionados ao *Bluetooth Low Energy* de forma geral e mais especificamente os *Beacons*. Em seguida uma seção trata dos dispositivos móveis e os principais sistemas operacionais móveis disponíveis no mercado. Na seção seguinte, as principais características e modo de funcionamento da plataforma *Android* são apresentadas. Por fim, a tecnologia de banco de dados utilizada é brevemente apresentada, o *SQLite*.

2.1 *BLUETOOTH 4.0*

A tecnologia *Bluetooth* é um sistema de comunicação sem fios usada para conectar dispositivos móveis a uma infinidade de equipamentos eletrônicos. As principais características dessa tecnologia estão relacionadas à robustez, baixo consumo de energia e baixo custo. Existem dois modos de operação: o *Basic Rate / Enhanced Data Rate* (BR/EDR) e o BLE. No modo *BR/EDR* é possível realizar comunicação sem fio contínua entre dispositivos usando a topologia de rede *point-to-point* (P2P), otimizada para transmissão de áudio. Já o modo BLE permite a comunicação sem fio através de pequenos pacotes de dados, mas que podem operar em múltiplas topologias de rede, como: a topologia P2P que é otimizada para transferir dados entre dois dispositivos conectados; a topologia *broadcast* ou *one-to-many* na qual há o envio de dados de forma unidirecional de um dispositivo fixo para vários dispositivos móveis realizando o compartilhamento informações, trazendo o conceito dos *beacons*; e ainda a topologia de rede *mesh*, para estabelecer comunicação *many-to-many*, que envolve a criação de redes de dispositivos que comunicam-se entre si, ideal para uso em redes de sensores ou automação ou rastreamento (BLUETOOTH, 2010).

A tecnologia *Bluetooth Low Energy* surgiu para viabilizar o avanço de novas aplicações e produtos para os quais a economia de energia é extremamente importante. Acessórios médicos

portáteis, *smartwatches*, pulseiras inteligentes e dispositivos de localização são exemplos de dispositivos que, por serem muito compactos, usam baterias de baixa capacidade. Para redução do consumo de energia, o BLE utiliza várias técnicas. Uma delas é a redução na velocidade de transferência de dados, que normalmente não passa de 1 Mb/s e aumento do tempo em "modo de descanso". Como não há muitos dados a serem transmitidos, uma conexão de poucos milissegundos é capaz de realizar a transferência de todos os dados necessários. Outra técnica é a redução do alcance da comunicação: o BLE trabalha bem com distâncias de até 30 metros.

2.2 BEACONS

Os *Beacons* são dispositivos que utilizam a tecnologia *Bluetooth Low Energy* para transmitir uma mensagem de *broadcast* para todos os dispositivos que estiverem em um raio de até 100 metros de distância e possuam a função *BLE* ativada. São dispositivos periféricos de comunicação e não devem ser utilizados para transferência de grandes volumes de dados, mas sim, uma pequena quantidade de dados. A nomenclatura "*Beacon*"significa farol em português, que dá uma boa ideia quanto ao seu modo de funcionamento: caracterizado pelo envio de um pequeno sinal de tempos tem tempos para que todos os dispositivos que estiverem em seu alcance possam receber, mas não podem enviar nenhuma resposta. Devem utilizar o sinal para identificação e localização simplesmente.

Atualmente existem diversos tipos de *Beacons*, sendo que os principais são *iBeacon*, *URI-Beacon* e *AltBeacon*, com diferentes padrões, vantagens e desvantagens. Alguns são de código aberto e outros são proprietários.

O *Bluetooth Low Energy* pode operar no modo conectado ou no modo *advertising*. O modo conectado faz uso da camada *Generic Attribute* (GATT) para transferência de dados na topologia P2P, já o modo *advertising* usa a camada *Generic Access Profile* (GAP) para enviar dados no modo *broadcast* para qualquer dispositivo que esteja escutando (MBED, 2015).

2.2.1 iBeacon

O *iBeacon* foi a primeira tecnologia *BLE Beacon* a surgir e assim a maioria dos *Beacons* foram inspirados no formato de dados do *iBeacon*. Os *iBeacons* estão disponíveis em muitas *SDKs* da *Apple* tornando seus produtos compatíveis com essa tecnologia. O *iBeacon* é proprietário da *Apple* e de código fechado.

A Figura 1 mostra a estrutura do pacote *iBeacon* que é formada inicialmente por um prefixo de 9 *bytes*, sendo: os 3 primeiros *bytes* (0x020106) uma indicação do tipo do pacote, neste caso

um *advertising*; em seguida 2 *bytes* (0x1AFF) formam o cabeçalho do pacote, informando que os próximos *bytes* são dados específicos do fabricante; mais 2 *bytes* (0x004C) indicam que o fabricante é a *Apple*; mais 1 *byte* para identificação secundária que é usada por todos os *beacons*; por fim vem a quantidade de *bytes* restantes no pacote.

Os demais dados do pacote são: 16 *bytes* para o UUID que é usado para identificação do *beacon*, que normalmente é único para cada fabricante; 2 *bytes* para o *Major Number* que identifica o *beacon* entre um grande grupo, normalmente usado para identificar as lojas (65536 possibilidades); 2 *bytes* para o *Minor Number* para identificação de forma mais específica, podendo ser usado para identificar objetos dentro de cada loja por exemplo (mais 65536 objetos); e 1 *byte* para *TX Power* para indicar a potência do sinal recebido (MBED, 2015).

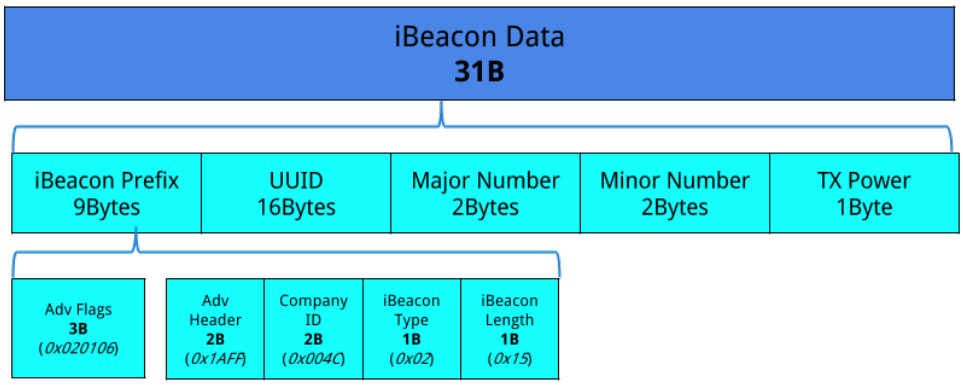

Figura 1: Especificação do iBeacon

2.2.2 AltBeacon

O *AltBeacon* é uma especificação aberta e livre desenvolvida pela *Radius Networks* que surgiu como uma resposta direta ao código fechado do *iBeacon*. O *AltBeacon* atende a todas as funcionalidades presentes no *iBeacon*, e foi desenvolvido para evitar o favorecimento de apenas uma empresa e para ser mais flexível e compatível com outras plataformas móveis. O *iBeacon* tem 20 *bytes* disponíveis para os dados de usuário *(UUID, Major e Minor)*, já o *AltBeacon* tem 26 *bytes* disponíveis *(MFG ID, BeaconCode, BeaconID, MFG RSVD)*, assim mais dados podem ser enviados por mensagem.

A Figura 2 mostra a especificação do *AltBeacon* que tem 28 *bytes*, sendo que apenas dois não podem ser alterados pelo usuário: o *AD Length* (0x1B) e o *AD Type* (0xFF). Os demais *bytes* do pacote podem ser modificados de acordo com a necessidade (MBED, 2015).

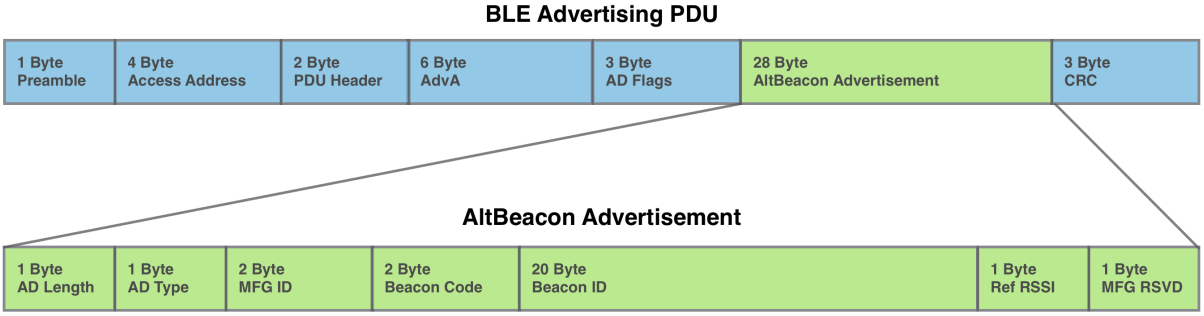

Figura 2: Especificação do AltBeacon

2.2.3 URIBeacon

O *URIBeacon* é um projeto do *Google* que busca enviar um *link* para um *website* no pacote de dados de um *beacon*, similar ao princípio de funcionamento dos *QR codes*. Esta especificação permite enviar pequenos endereços de páginas através dos pacotes *advertising BLE*. A ideia do *Google* basicamente foi o de criar um conceito um pouco mais amplo, de forma que não seja mais necessário que o usuário tenha que instalar um aplicativo específico para cada aplicação utilizando *Beacons*, neste caso, quando um *Beacon* enviar um link da web, o mesmo poderá ser acessado diretamente de qualquer navegador instalado no *smartphone*.

O modo de funcionamento dos *URIBeacons* é um pouco diferente dos *iBeacons* e *AltBeacons*, que a princípio devem ser configurados uma única vez e podem permanecer em operação para sempre. No caso dos *URIBeacons*, é necessário ter um serviço de configuração para atualizar periodicamente as informações que são enviadas pelos *beacons*.

Na especificação dos *URIBeacons* utilizam-se 28 dos 31 *bytes* disponíveis em um pacote *advertising*, como mostra a Figura 3. Dos 28 *bytes*, 19 são usados para montar a *Uniform Resource Locator* (URL. Para economizar alguns *bytes*, o prefixo (’www.’, ’http://’, etc) são codificados em apenas um *byte*, assim como o sulfixo (’.com/’, ’.org/’, etc) (MBED, 2015).

2.3 DISPOSITIVOS MÓVEIS

Com o avanço tecnológico, os computadores foram aumentando seu poder de processamento, capacidade de armazenamento, se tornaram mais práticos e fáceis de usar, e tiveram seu tamanho e peso reduzidos ao ponto de poder ser levados para qualquer lugar. Esses equipamentos são conhecidos como dispositivos móveis e os maiores exemplos são os *tablets* e *smartphones*. Com o constante aumento na demanda por esses equipamentos, houve consequentemente um aumento na procura por softwares capazes de operá-los de forma eficiente. As

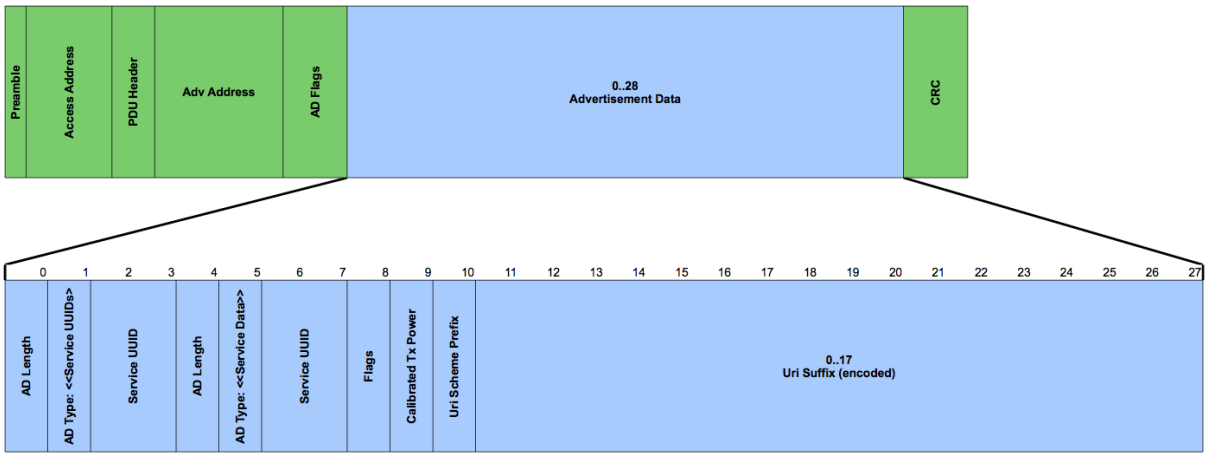

Figura 3: Especificação do URIBeacon

plataformas mais comuns no mercado atualmente são as plataformas *Android*, *iOS* e *Windows Phone* (SOARES, 2016).

De acordo com uma pesquisa publicada pela *International Data Corporation* (IDC (IDC, 2017), as empresas de telefonia venderam um total de 344,3 milhões de *smartphones* em todo o mundo no primeiro trimestre de 2017. A pesquisa aponta um crescimento de 3,4% nas vendas quando comparado com os dados do ano anterior. O gráfico da Figura 4 apresenta em números percentuais um panorama do mercado de *smartphones* destacando os principais sistemas operacionais utilizados atualmente. Os dados apontam que, no primeiro trimestre de 2017, os *smartphones* que utilizam o *Android* representam 85% do mercado, enquanto que 14,7% utilizam o sistema operacional *iOS* e apenas 0,2% utilizam a plataforma *Windows Phone* e outras.

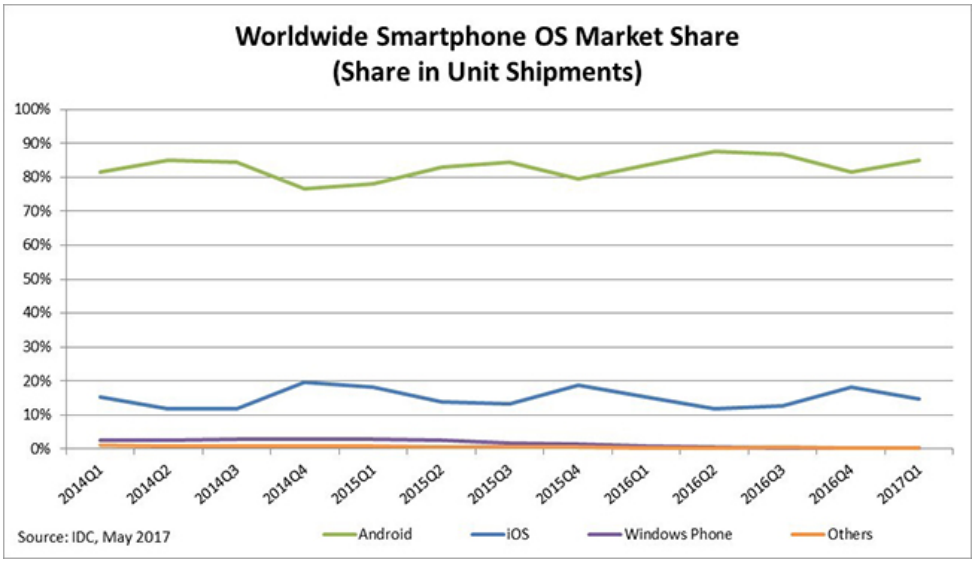

Figura 4: Mercado mundial de sistemas operacionais móveis

2.3.1 Android

O *Android* é uma pilha de softwares que inclui um sistema operacional baseado no *Linux*, um *Middleware* e um conjunto de aplicações chave. Foi projetado para ser executado em dispositivos móveis com telas sensíveis ao toque, como *smartphones* e *tablets*. A empresa Android Inc. foi fundada em Palo Alto na Califórnia por Andy Rubin, Rich Miner, Nick sears e Chris White em 2003. Em 2005, *Android Inc.* foi comprada pelo *Google*. Hoje é mantida pelo *Google* e pela *Open Handset Alliance* (OHA), que é uma aliança entre 86 empresas de hardware, software e telecomunicações que trabalha para ter um padrão aberto para dispositivos móveis (JONNALAGADDA, 2012).

As aplicações para *Android* normalmente são desenvolvidas utilizando-se a linguagem de programação *Java* através do *Android Software Development Kit* (SDK). O código java é compilado em um arquivo binário (.class) que é independente de qualquer plataforma. Um interpretador *Dalvik* converte o arquivo .class em arquivos executáveis .dex para que possam ser interpretados em tempo de execução. O SDK *Android* disponibiliza todas as *Application Programming Interface* (APIs) necessárias e todas as ferramentas para construção, testes e depuração de aplicações *Android*.

Existem cinco componentes essenciais para a construção de uma aplicação *Android*: *Activity*, *Intents*, *Broadcast Receiver*, *Services* e *Content Provider*. O entendimento em detalhes desses componentes é muito importante para um desenvolvedor, basicamente porque todas as principais ações (troca de telas, manipulação de banco de dados, tratamento de eventos, recebimento de notificações, etc.) desempenhadas por uma aplicação são dependentes delas (POUDEL, 2013).

Uma *Activity* é um componente que cria uma tela na qual os usuários podem interagir com a aplicação para executar determinadas tarefas. Uma aplicação pode ter muitas *Activities* pelas quais os usuários podem navegar. A classe *Activity* é disponibilizada pelo *Framework Android* e oferece um série de facilidades como apresentação da interface para o usuário, criação de um novo processo e alocação de memória para os objetos da interface. Normalmente um aplicativo *Android* inicia com uma *Main Activity*. Uma *Activity* pode iniciar ou parar outras *Activities* para executar diferentes ações dentro da aplicação. Quando o usuário inicia uma nova *Activity*, a *Activity* anterior é parada e o sistema *Android* preserva esse processo na pilha. Assim a *Activity* anterior pode continuar sua execução do ponto em que parou quando o botão para voltar for pressionado. O *Android* tem um ciclo de vida de *Activities* bem definido.

As *Intents* representam as ações ou eventos que dão início a uma *activity*, disparam o iní-

cio ou o término de um serviço, ou até um *broadcast* na aplicação. As *intents* permitem que você interaja com componentes do mesmo aplicativo e também com componentes de outras aplicações. Por exemplo, uma *activity* pode iniciar uma *activity* externa para tirar uma foto.

Os *Broadcast Receivers* são componentes responsáveis por receber e tratar eventos (*ou broadcasts*) provenientes do sistema ou de outras aplicações. Este componente permite que os usuários possam registrar eventos de sistema e receber notificações quando os eventos registrados ocorrerem, como uma notificação de SMS ou o nível crítico de bateria, por exemplo. O *receiver* é simplesmente uma camada de código que é executada somente quando um evento registrado for disparado. Os *broadcasts* podem ser enviados de uma parte da aplicação para outra, ou até mesmo para outra aplicação totalmente diferente.

Os *Services* são os componentes da aplicação que podem executar operações de longa duração em segundo plano (*background*). Os serviços são executados de forma invisível, atualizando fontes de dados ou *Activities* e disparando notificações. O *Android OS* disponibiliza e processa serviços do sistema que devem ser declarados em todas as aplicações *Android*.

Os *Content Providers* são componentes utilizados para gerenciar e compartilhar bancos de dados. Múltiplas aplicações podem compartilhar os mesmos dados de várias formas dependendo do tipo de dados. Várias aplicações podem acessar a mesma fonte de dados simultaneamente graças ao *Content Provider*. O *Android* tem os *content providers* nativos para gerenciar dados de áudio, vídeo, imagens e informações de contato pessoal.

Cada componente tem um papel específico e contribui para o comportamento da aplicação como um todo. Todos os componentes usados na aplicação e utilização de recursos devem ser declarados no arquivo *Manifest*. A aplicação *Android* também é composta de recursos que são separados do código fonte. Esses recursos, como imagens, arquivos de áudio, entre outros, contribuem para a apresentação visual da aplicação (JONNALAGADDA, 2012).

2.4 SQLITE

O *SQLite* é um banco de dados de código aberto que foi lançado em 2000. É muito estável e popular em muitos dispositivos compactos, incluindo os dispositivos que utilizam o *Android*. Há uma série de motivos pelos quais o *SQLite* é considerado uma ótima solução para ser utilizado no desenvolvimento de aplicativos *Android*: não é necessário realizar nenhum tipo de configuração da base de dados, o que torna seu uso extremamente simples; não há um servidor de banco de dados sendo executado, o *SQLite* é basicamente uma biblioteca que oferece as funcionalidades de uma base de dados; o banco de dados é composto por um único arquivo e é de

código aberto (GARGENTA, 2011).

O *Framework Android* oferece uma série de opções para utilização do *SQLite* de forma simples e eficiente para que os desenvolvedores possam implementar a persistência dos dados localmente no *smartphone*.

2.5 *ANDROID STUDIO*

Antigamente, o desenvolvimento para *Android* era realizado através da plataforma *Eclipse* com o *Android Development Kit* (ADK), fornecido pelo *Google*, que lançou a plataforma *Android Studio*. O *Android Studio* é o Ambiente de Desenvolvimento Integrado (IDE) oficial para a criação de aplicativos *Android* e é baseado no *IntelliJ IDEA*, a Figura 5 mostra a SDK.

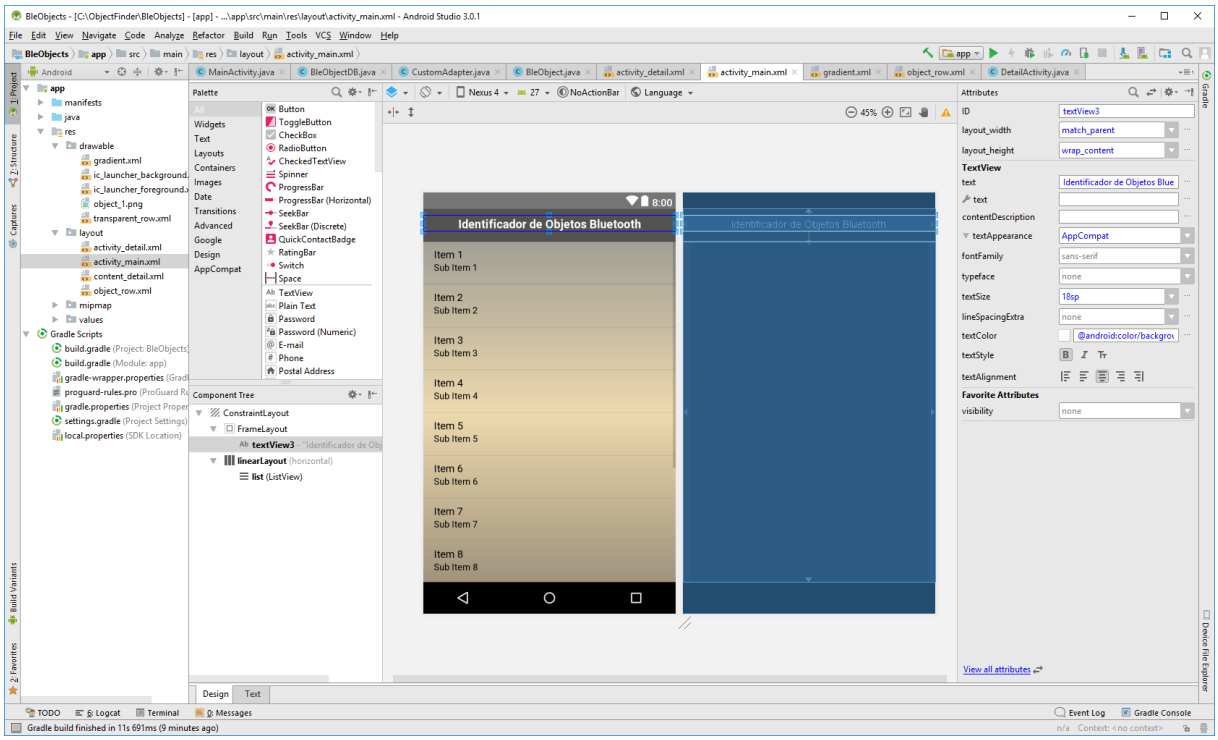

Figura 5: IDE do Android Studio

Além do editor e das ferramentas de desenvolvedor avançados do *IntelliJ*, o *Android Studio* oferece diversos recursos para aumentar a produtividade, como: um sistema de compilação flexível baseado no *Gradle*; emulador com suporte a diversos recursos; ambiente unificado para desenvolver para todos os dispositivos *Android*; *Instant Run* para aplicar alterações a aplicativos em execução sem precisar compilar um novo APK; modelos de códigos e integração com *GitHub* para ajudar a criar recursos comuns dos aplicativos e importar exemplos de código; ferramentas e estruturas de teste.

2.6 *WEB SERVICES*

Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Esta tecnologia permite que novas aplicações possam interagir com aquelas já existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *Web Services* são componentes que permitem a troca de dados entre dois sistemas. Cada aplicação pode ter uma linguagem de programação específica, que deve ser traduzida para uma linguagem universal como o XML (*Extensible Markup Language*), JSON (*JavaScript Object Notation*), CSV (*Comma-separated values*), entre outros.

O *Web Service* é composto por duas estruturas: o **serviço** e a **descrição do serviço**. O **serviço** consiste num módulo de software instalado numa plataforma computacional com acesso à rede e é oferecido pelo provedor de serviços. A **descrição do serviço** contém os detalhes de funcionamento da interface e informações específicas da implementação de um serviço (OLIVEIRA, 2012).

A definição mais aceita pela comunidade de Tecnologia da Informação (TI) é da *World Wide Web Consortium* (W3C) onde um *Web Service* pode ser considerado um sistema de software projetado para apoiar interações entre máquinas através da rede, fornecendo uma interface descrita em um formato processável por máquina, o WSDL (*Web Services Description Language*) (OLIVEIRA, 2012).

Para a representação e estruturação dos dados nas mensagens recebidas e enviadas é utilizado o XML. As chamadas às operações, incluindo os parâmetros de entrada e saída, são codificadas no protocolo SOAP (*Simple Object Access Protocol*). Os serviços (operações, mensagens, parâmetros, etc.) são descritos usando a linguagem WSDL. A Figura 6 apresenta o modo de funcionamento básico de um *Web Service* (OLIVEIRA, 2012).

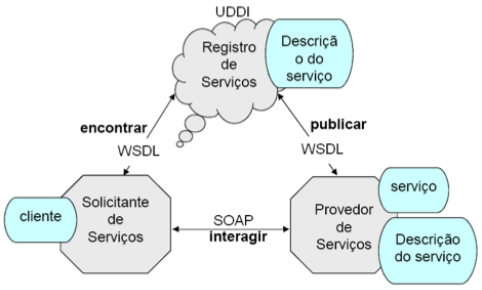

Figura 6: Web Services, Fonte: (MACEDO, 2018)

Para o desenvolvimento deste trabalho, um webservice foi criado com o auxílio da IDE NetBeans, utilizando um servidor GlassFish com acesso a um banco de dados Postgrees.

3 DESCRIÇÃO DA PROPOSTA

Este capítulo apresenta a proposta do trabalho de forma mais detalhada. Inicialmente a visão funcional da aplicação é apresentada através de um diagrama geral e, posteriormente, são mostrados os casos de uso. Por fim, um diagrama de sequência ilustra o fluxo de mensagens entre todas as entidades da aplicação.

3.1 VISÃO GERAL

O sistema proposto é composto de três módulos distintos: um *Web Service*; uma aplicação móvel; e módulos *Bluetooth LE*, como mostra a Figura 7.

Figura 7: Diagrama geral do sistema.

O *Web Service* deve disponibilizar um serviço capaz de fornecer as informações detalhadas de todos os objetos previamente cadastrados em um banco de dados. Sempre que a aplicação móvel detectar o sinal de um novo objeto, uma consulta ao servidor deverá ser realizada para que o objeto possa ser identificado e apresentado ao usuário. Os módulos *Bluetooth LE* são dispositivos que operam no modo *iBeacon*, que são configurados para enviar periodicamente uma mensagem curta contendo um número de identificação e dados relacionados à potência do sinal. Este número deve estar relacionado ao objeto no banco de dados para permitir sua identificação posterior.

3.2 CASOS DE USO

Os casos de uso foram agrupados em dois diagramas, a Figura 8 traz um diagrama com todos os casos de uso que representam as ações que o usuário pode realizar para interagir com o *Webservice*, e a Figura 9 relacionados com o aplicativo móvel, com todas as ações de interação que podem ocorrer com a aplicação.

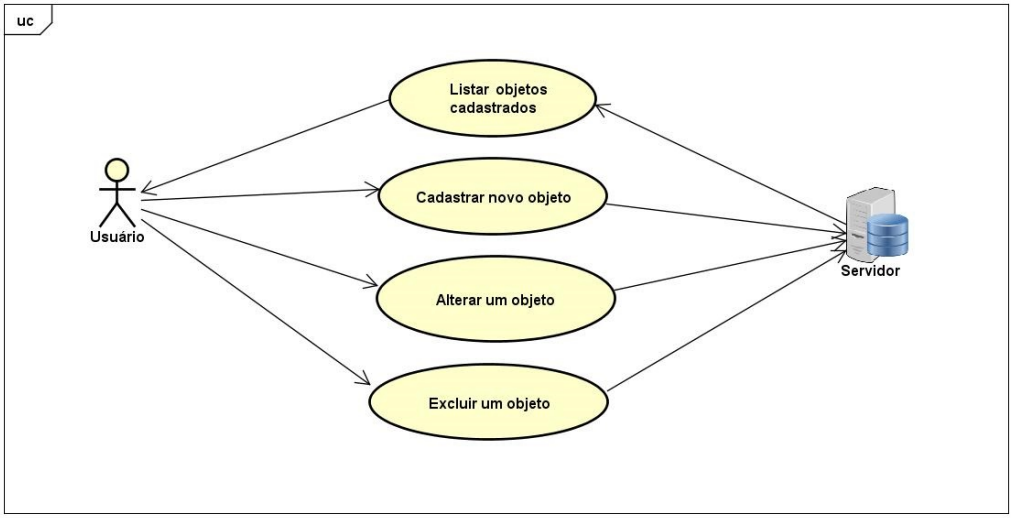

Figura 8: Casos de uso - Webservice

3.2.1 Listar objetos cadastrados

Este caso de uso permite que o usuário tenha acesso a uma lista de todos os objetos que estão efetivamente cadastrados no banco de dados do *Webservice*. Isto serve, justamente, para possibilitar a edição ou exclusão de determinados objetos.

3.2.2 Cadastrar novo objeto

O usuário pode cadastrar um novo objeto no banco de dados do servidor, lembrando que é necessário relacionar a identificação do *iBeacon* (UUID) com o novo registro. Ao cadastrar um novo objeto, o usuário deve informar todos os dados mais relevantes, como o nome, descrição, identificação do *iBeacon* além de fazer *upload* de uma foto.

3.2.3 Alterar um objeto

Após consultar a lista de objetos cadastrados, o usuário pode alterar qualquer campo de um registro, para manter sempre atualizado o cadastro.

3.2.4 Excluir um objeto

Este caso de uso permite que o usuário possa excluir um objeto que não deve ser mais identificado pela aplicação, ou também quando for o caso de alteração do *tag* de um objeto para outro.

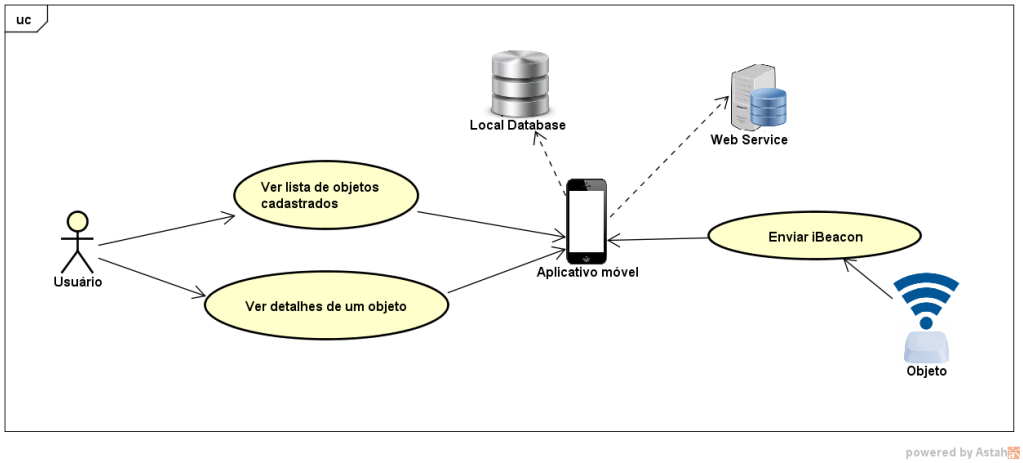

Figura 9: Casos de uso - Aplicativo

3.2.5 Ver lista de objetos ativos

Este caso de uso permite que o usuário do aplicativo móvel possa visualizar uma lista dos objetos ativos mais próximos, já previamente identificados através do servidor ou, até mesmo, os objetos que ainda não efetivaram sua consulta ao *Webservice*.

3.2.6 Ver detalhes de um objeto

O usuário tem a possibilidade de selecionar um objeto na lista para visualizar o detalhamento deste objeto sobre o qual tem interesse.

3.2.7 Enviar *iBeacon*

Este caso de uso descreve a comunicação entre os objetos e o aplicativo móvel. Neste caso, o objeto deve enviar periodicamente pacotes de iBeacon para que o aplicativo possa detectar seu número de identificação e a potência de seu sinal para estimativa da distância.

3.3 DIAGRAMA DE CLASSES

As classes da aplicação são mostradas na Figura 10.

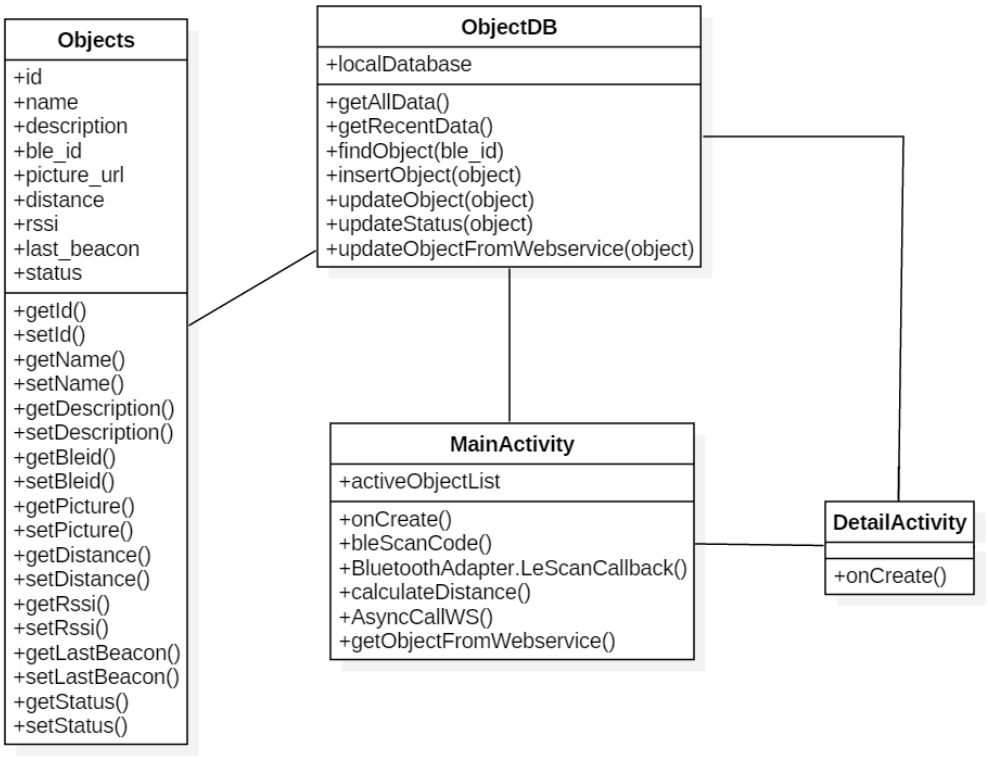

Figura 10: Diagrama de Classes

O desenvolvimento do aplicativo consistiu na criação de uma classe para representar um objeto a ser localizado pelo sistema, esta classe tem como atributos o nome do objeto, uma descrição detalhada, a identificação bluetooth, o caminho para uma imagem do objeto, a distância entre o objeto e o smartphone, a intensidade do último sinal de rádio recebido, a estampa de tempo do último sinal recebido, e por fim um status, para indicar se o sistema já tentou fazer a busca no webservice ou não.

Uma segunda classe para controlar o acesso banco de dados local foi criada com o nome de ObjectDB, através dessa classe é possível: buscar uma lista de todos os objetos cadastrados no banco; buscar uma lista com os objetos ativos durante os últimos 30 segundos; localizar um objeto específico através da identificação bluetooth; inserir um objeto novo no banco; atualizar os dados de um objeto; atualizar somente o atributo de status; ou ainda atualizar os dados com as informações recebidas do webservice.

As demais classes são duas activities para controlar as duas telas da aplicação. É na MainActivitie que as tarefas de ativação e controle do módulo de bluetooth são executadas, além das consultas no webservice quando necessário.

3.4 DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência da Figura 11 apresenta o fluxo de troca de mensagens entre todas as entidades envolvidas no processo.

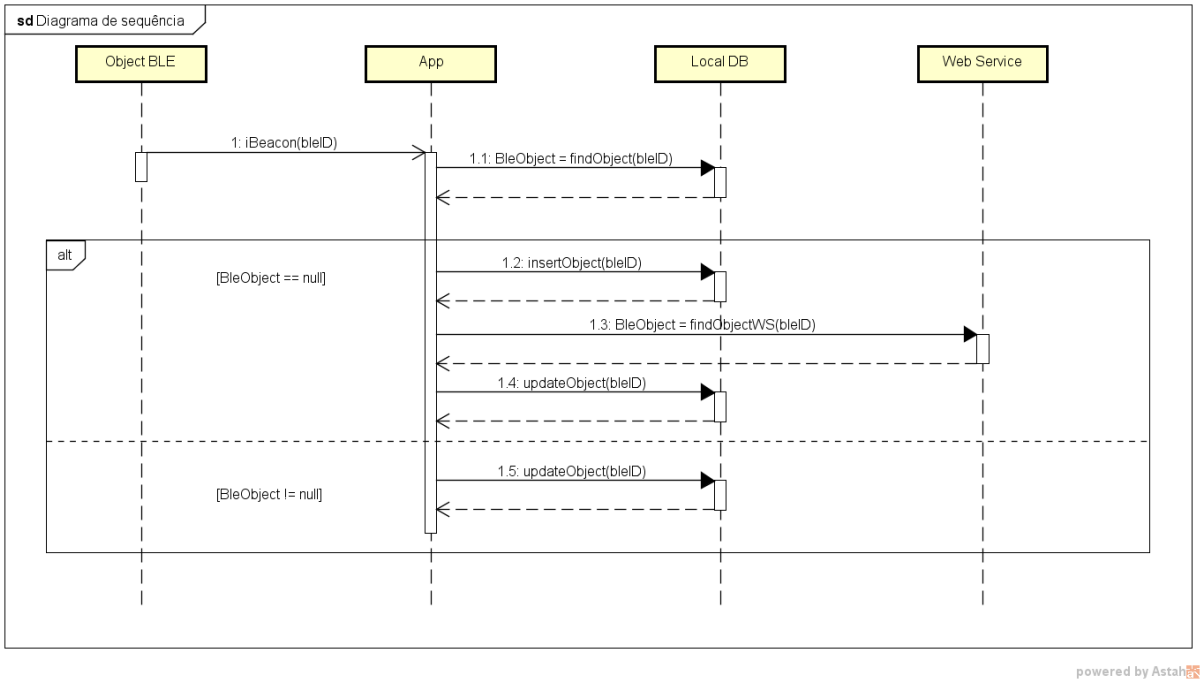

Figura 11: Diagrama de Sequência

Este caso de uso começa com o envio de uma mensagem do tipo *iBeacon* por um dispositivo BLE acoplado em um objeto. Quando essa mensagem é detectada pelo aplicativo móvel, uma busca no banco de dados local é realizada através do comando *findObject(bleID)* passando como parâmetro o número de identificação do objeto e obtendo-se como resposta o objeto encontrado, ou uma resposta nula.

Se a resposta for nula, um novo objeto é inserido no banco de dados local através do comando *insertObject(bleID)*. Em seguida, o aplicativo vai para a segunda etapa, que é a busca no servidor remoto através do comando *findObjectWS(bleID)*. Caso o *Webservice* retorne os dados do objeto encontrado, o aplicativo atualiza a base de dados local através do comando *updateObject(bleID)*.

Mas se a resposta não for nula, ou seja, se o objeto for encontrado na base de dados local, o aplicativo somente atualiza alguns dados no registro do objeto, através do comando *updateObject(bleID)*, como o campo RSSI, a distância entre o *Smartphone* e o objeto, e ainda a estampa de tempo da mensagem recebida.

3.5 MÓDULO BLUETOOTH LOW ENERGY

O módulo *BLE* é utilizado nesta aplicação atuando como um dispositivo que deve ser acoplado a algum objeto de interesse. Este módulo possui uma identificação única que deve ser cadastrada no banco de dados do *Webservice*. Assim, sempre que o sinal deste módulo for identificado pela aplicação, será possível consultar seus dados e fotos no momento apropriado.

Um dos módulos de Bluetooth LE mais comuns atualmente é o HM-10, mostrado na Figura 12, que utiliza um chip TI CC2540 com suporte à versão 4.0 do bluetooth. Este módulo opera na frequência de 2.4GHz com modulação GFSK (*Gaussian Frequency Shift Keying*). Deve ser alimentado com uma fonte de tensão de 3.3 volts e seu consumo de corrente pode variar de acordo com o modo de operação, sendo 400uA no modo Sleep e 8.5mA no modo ativo. Sua interface de comunicação é baseada em uma porta serial através de comandos AT, que podem ser usados para leitura e configuração de diversos parâmetros. No mercado nacional seu custo gira em torno de R$50,00 por unidade. Este módulo suporta os dois modos de operações do *Bluetooth Low Energy* podendo operar no modo conectado ou no modo *advertising*. No caso deste trabalho, foi utilizado somente o modo *advertising*, que é o modo como o *iBeacon* opera, justamente para se obter o menor consumo de energia enviando sinais periódicos em *broadcast*. O módulo foi configurado para enviar pacotes periódicos em *broadcast* contendo somente uma identificação e os dados de RSSI.

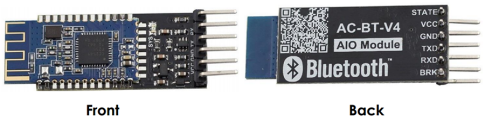

Figura 12: Módulo Bluetooth HM-10

Esta tecnologia já vem sendo explorada comercialmente por diversas empresas desde 2013. Os fabricantes oferecem uma série de módulos completos que se diferenciam pelo tipo do invólucro, autonomia de bateria, alcance do sinal *iBeacon*, entre outros. Exemplos de módulos comerciais são apresentados na Figura 13.

Figura 13: Módulos comerciais iBeacon

3.6 DESCRIÇÃO DO APLICATIVO

O aplicativo foi projetado para apresentar aos usuários uma interface bastante simples e foi desenvolvido para a plataforma *Android*. Logo após a instalação do aplicativo, é necessário dar permissão de localização geográfica, manualmente, para que a aplicação tenha acesso ao resultado da busca de pacotes *BLE*. Este procedimento é mostrado nas duas primeiras telas da Figura 14. Em seguida, ao iniciar a aplicação, caso o *bluetooth* esteja desligado, a aplicação solicita permissão para usar o *bluetooth*, como mostra a terceira tela da Figura 14.

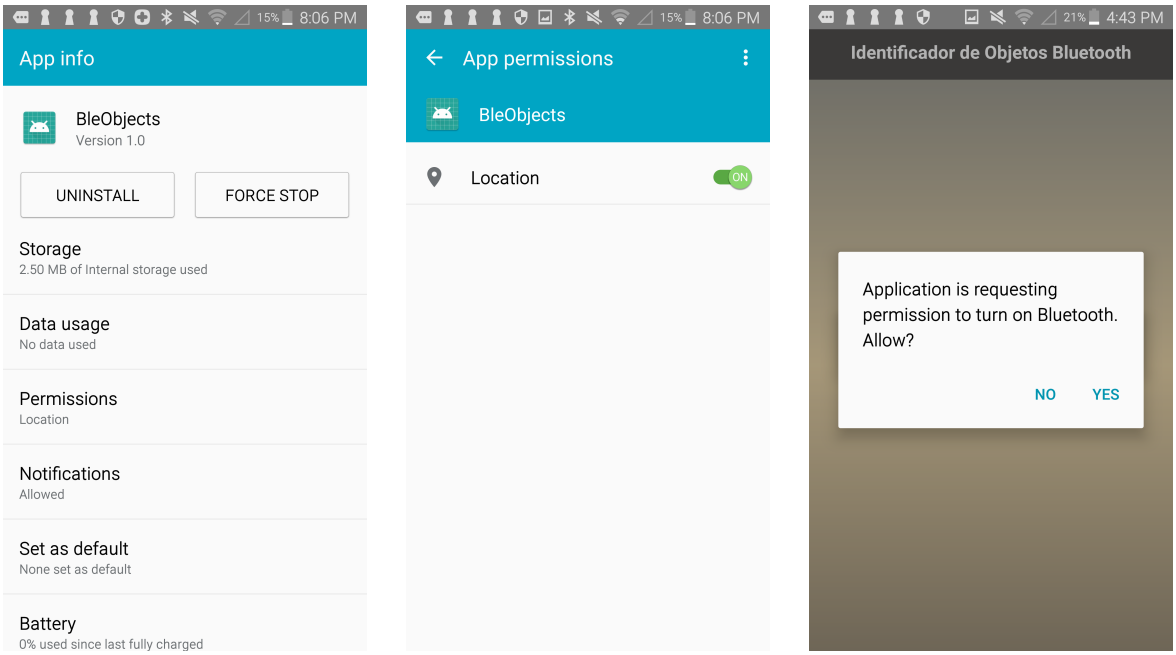

Figura 14: a) Informações do aplicativo; b) Permissão de localização; c) Permissão para ligar o *Bluetooth*

Para realizar a busca de sinais *BLE*, um mecanismo temporizado foi desenvolvido, fazendo com que o aplicativo execute uma busca por 100ms para localizar os dispositivos próximos e fique em repouso por 500ms. Esse ciclo se repete enquanto a aplicação estiver em operação.

Quando uma mensagem do tipo *iBeacon* for recebida, um registro deve ser adicionado na lista de objetos com o nome de objeto desconhecido. O aplicativo calcula a distância aproximada entre o *smartphone* e a fonte emissora do sinal de rádio através da informação de *RSSI (Received Signal Strength Indicator)*, que está relacionado com a potência do sinal recebido de um pacote *bluetooth*. Após a realização dos cálculos, o resultado é convertido em metros para que o usuário possa identificar quais objetos estão mais próximos. A Figura 15 (à esquerda) mostra uma tela com dois objetos que nunca foram identificados pelo *Web Service*.

Um banco de dados local em *SQLite* foi criado para manter um cadastro de todos os objetos

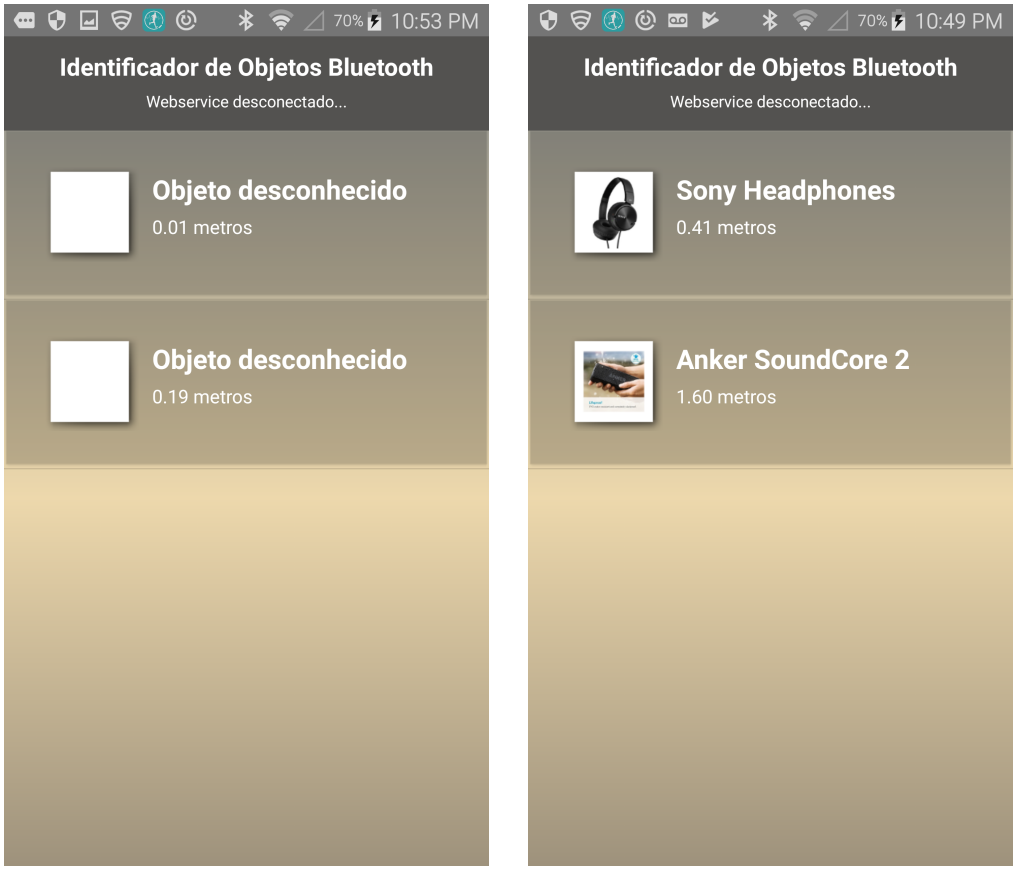

Figura 15: Tela principal com objetos desconhecidos (à esquerda), e objetos conhecidos (à direita)

já identificados pela aplicação. As imagens dos objetos são mantidas em cache no celular para garantir um bom desempenho durante sua utilização. Sempre que o sinal de um novo objeto for recebido pelo aplicativo, uma consulta ao banco de dados local é realizada na tentativa de encontrar o registro correspondente. Mas se o registro não for localizado, uma busca no *WebService* deve ser disparada em uma segunda tentativa de encontrar o objeto em questão. Caso o registro do objeto tenha sido encontrado no servidor, todos os dados são enviados para a aplicação, que se encarrega de efetivar a gravação desses dados no banco local e faz o *download* das imagens quando necessário. A Figura 15 (à direita) mostra uma tela com dois objetos já identificados na lista.

Os objetos localizados permanecem visíveis na interface enquanto o sinal *iBeacon* estiver presente. Mas se o aplicativo ficar mais de 30 segundos sem receber mensagens de um determinado objeto, o objeto é retirado da lista. Assim somente os objetos ativos e próximos são mostrados ao usuário, e a medida em que o usuário de move e se distancia de alguns objetos, eles ficam ocultos para que a interface não fique muito poluída. Os objetos localizados são apresentados ao usuário ordenados pela distância, assim, os objetos mais próximos sempre devem permanecer no topo da lista.

É possível selecionar os objetos da lista para visualização de seus detalhes e informações adicionais. Na tela de detalhamento de objetos, pelo menos uma imagem em tamanho grande é apresentada, seguida de um texto com a descrição completa do objeto. Nesta tela, futuramente, outras informações poderão estar disponíveis, como vídeos, áudios ou até mesmo *links* para outros *websites* correlacionados com o objeto. A Figura 16 mostra duas telas com os detalhes de dois objetos já identificados.

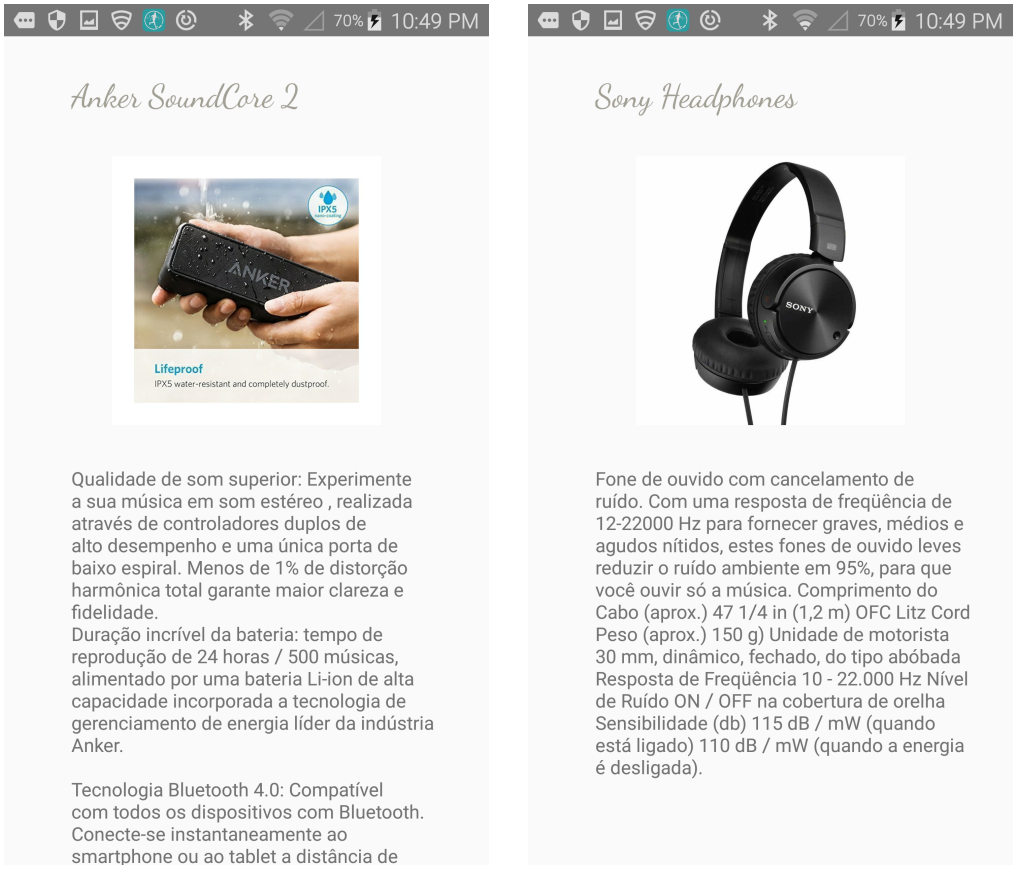

Figura 16: Telas de detalhamento de objetos já identificados

4 AVALIAÇÃO DO APLICATIVO MÓVEL

Para validar o funcionamento do aplicativo, realizou-se um teste para identificar um sinal *Bluetooth* através do aplicativo, já com o número de identificação cadastrado no banco de dados local.

Para realização do teste, utilizou-se um módulo *Bluetooth* modelo AC-BT-V4, que utiliza uma placa HM-10 previamente configurado para operar no modo *iBeacon*. O aplicativo foi instalado em um *Smartphone Samsung Galaxy Note 4* que oferece suporte à tecnologia *Bluetooth 4.0*.

O módulo foi energizado conectando-o à porta USB de um computador. Ao iniciar o aplicativo, primeiramente foi solicitado ao usuário a permissão para acessar o recurso *Bluetooth* do aparelho, e em seguida, a tela principal foi apresentada. Imediatamente uma lista com apenas um objeto foi mostrada, juntamente com sua foto em tamanho reduzido, indicando uma distância de 13 centímetros, como mostra a Figura 17.

Em seguida, o módulo foi afastado do aparelho. Constatou-se que a indicação de distância também teve seu valor alterado para um número maior. Verificou-se, no entanto, que a distância não se apresentou muito estável, apresentando um erro de até 2 metros. Este comportamento já era esperado por se tratar de sinais de rádio.

Para validar o funcionamento do *Web Service*, dois objetos foram cadastrados manualmente no banco de dados do servidor, com todos os campos preenchidos. Em seguida, com o serviço ainda desligado, o aplicativo foi apagado para excluir o banco de dados local, e reinstalado. Ao iniciar o aplicativo, dois *iBeacons* foram energizados para que o sistema pudesse fazer o cadastro no banco de dados local com o nome de Objeto Desconhecido. Somente então o *Web Service* foi iniciado para confirmar que o aplicativo estava de fato fazendo as consultas corretamente, identificando os objetos somente pelo ID do *iBeacon*. Imediatamente, o aplicativo alterou a indicação de Objeto desconhecido para o nome correto de cada objeto assim como as imagens dos objetos foram mostradas corretamente.

Figura 17: Teste de funcionamento

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho envolveu o projeto e o desenvolvimento de uma solução completa para localização de objetos utilizando a tecnologia *Bluetooth Low Energy*, mais especificamente no modo *iBeacon*. Com relação aos aspectos técnicos, é possível afirmar que todos os objetivos propostos foram alcançados, resultando em um aplicativo funcional, inclusive com consultas em um *Web Service* para identificação e recebimento dos dados de cada objeto localizado. Durante a implementação do aplicativo, os maiores problemas enfrentados foram no acesso e utilização do *bluetoot"* no modo *iBeacon* e também no desenvolvimento do *Web Service*.

Para trabalhos futuros, pretende-se focar em apenas um setor, desenvolvendo um sistema focado em atender os anseios de usuários específicos desse setor. O *Web Service* deve ser aprimorado para suportar uma possível demanda de muitos usuários simultâneos e uma série melhorias poderão ser aplicadas ao aplicativo para melhorar a usabilidade do sistema. Uma interface para cadastro de objetos no *Web Service* também precisa ser desenvolvida para viabilizar a utilização do sistema.

REFERÊNCIAS

BLUETOOTH. **Specification of the Bluetooth System**. 2010.

GARGENTA, M. **Learning Android**. First edition. [S.l.: s.n.], 2011.

IDC. **Smartphone OS Market Share**. 2017. Disponível em: <https://www.idc.com/promo/smartphone-market-share/os>.

JONNALAGADDA, S.

Android Application for Library Resource Access (mathesis), 2012.

MACEDO, D. **Web Services**. 2018. Disponível em: <http://www.diegomacedo.com.br/web-services/>.

MBED. **Understanding the different types of BLE Beacons**. 2015. Disponível em: <https://os.mbed.com/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/>.

OLIVEIRA, R. R. de. **Avaliação de Manutenibilidade entre as Abordagens de Web Services RESTful e SOAP-WSDL**. Dissertação (Mestrado), 2012.

POUDEL, A.

Mobile Application Development for Android Operating System (candthesis), 2013.

SOARES, M. A.

Aplicativo móvel para academia: Estudo de Tecnologias e desenvolvimento (candthesis), 2016.