UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

MAPA DE CALOR BASEADO EM GEOLOCALIZAÇÃO INTERNA COM UTILIZAÇÃO DE BEACONS

LEONARDO ADRIANO REICHERT

BLUMENAU
2017

LEONARDO ADRIANO REICHERT

MAPA DE CALOR BASEADO EM GEOLOCALIZAÇÃO INTERNA COM UTILIZAÇÃO DE BEACONS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Profa. Luciana Pereira de Araújo, Mestra - Orientadora

BLUMENAU
2017

MAPA DE CALOR BASEADO EM GEOLOCALIZAÇÃO INTERNA COM UTILIZAÇÃO DE BEACONS

Por

LEONARDO ADRIANO REICHERT

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Profa. Leonardo Adriano Reichert, Mestre – Orientadora, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Profa. Liliane Heiden, Bacharel – FURB

Blumenau, 12 de dezembro de 2017

Dedico este trabalho a todos que me apoiaram e principalmente as pessoas que de alguma forma contribuíram para o resultado final, seja com dúvidas ou com indicações e opiniões.

AGRADECIMENTOS

Aos meus amigos que me apoiaram e incentivaram durante todo o processo do trabalho.

À minha orientadora, Prof.ª. Luciana Pereira de Araújo, que participou de cada etapa sempre disposta a ajudar e entender cada opinião minha.

À Liliane Heiden por parte da Cia Hering, em todo apoio e aquisição dos beacons para a realização dos testes.

Aos colegas da área que muitas vezes me socorreram com respostas para as dúvidas no decorrer do desenvolvimento.

“When the snows fall and the white winds blow, the lone wolf dies, but the pack survives”

Eddard Stark

RESUMO

Este trabalho apresenta a especificação, desenvolvimento e operacionalidade de uma ferramenta que tem por objetivo mapear o uso de beacons de modelo eddystone em localização *indoor*, através do uso de dispositivos Android. A ferramenta não faz uso de aplicação instalada no dispositivo, ela utiliza de meios nativos do Android para operar. Além disso, o trabalho desenvolvido também expôs um meio de geração de páginas HTML sem necessidade de conhecimento em linguagens de programação. A ferramenta foi desenvolvida utilizando a linguagem de programação JavaScript, Angular JS, Java e o banco de dados relacional PostgreSQL. A ferramenta completa foi publicada em um servidor na Internet. Os resultados obtidos foram satisfatórios e mostraram que as tecnologias adotadas foram aderentes ao objetivo, bem como cumpriram com exatidão o que foi proposto. Outra conclusão realizada é com relação aos *frameworks* adotados e a ferramenta em si, ambos se mostraram muito extensivos e abertos às novas possibilidades.

Palavras-chave: Beacons. Eddystone. Android. HTML. Sem instalação.

ABSTRACT

This paper presents the specification, development and operability of a tool that aims to map the use of eddystone model beacons in indoor locations using Android devices. The tool does not use an application installed on the device, it uses Android's native system to operate. Furthermore, the developed tool also exposes a way of generating HTML pages without needing programming knowledge. The tool was developed using the programming language JavaScript, Angular JS, Java and the relational database PostgreSQL. The complete tool was published to a server on the Internet. The results obtained were satisfactory and showed that the adopted technologies were adherent to the objective and fulfilled precisely what was proposed, another conclusion made is related to the adopted frameworks and the tool itself, both were very extensive and open to new possibilities.

Key-words: Beacons. Eddystone. Android. HTML. No installation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BLE - Bluetooth Low Energy

CSS - Cascading Style Sheets

FH-CDMA - Frequency Hopping - Code-Division Multiple Access

FH/TDD - Frequency Hopping - Time Division Duplex

FIPS - Federal Information Processing Standards

FURB - Fundação da Universidade Regional de Blumenau

GPS - Global Positioning System

HTML - HyperText Markup Language

ID - Identify

IPV6 - Internet Protocol Version 6

ISM - Industrial, Scientific, Medical

JDBC - Java Database Connectivity

JSON - JavaScript Object Notation

JSP - java server pages

MVC - Model View Controller

MVP - Model View Presenter

MVVM - Model View Viewmodel

MVW - Model View Whatever

RF - Requisitos Funcionais

RFID - Radio-Frequency Identification

RNF - Requisitos Não Funcionais

RSSI - Received Signal Strength Indication

UC - Use Case

UID - Unique Identifier

URL - Uniform Resource Locator

URLID - Uniform Resource Locator Identifier

SUMÁRIO

1 INTRODUÇÃO

O entendimento da informação ainda passa por muita dificuldade (SILVA, 2011). A obtenção dos dados é um processo conhecido e explorado, mas o uso deles é que cria um dilema. Por mais que se tenha uma quantidade enorme de informação se não souber como, onde ou quando usá-las elas acabam se tornando informações desencontradas que necessitam de ligações para assim terem um significado (SILVA, 2011). Com o passar do tempo as informações e conhecimentos tem tido um destaque por terem seu impacto reconhecido em vários meios, dentre eles o organizacional (ALCARÁ et al., 2010). A respeito ao compartilhamento de informação é possível definir que:

> Na literatura o compartilhamento é um processo de partilha, de divisão de conhecimentos obtidos; já a informação é um conjunto de conhecimentos, experiências e habilidades adquiridos por alguém sobre alguma coisa. Compartilhamento de informações seria estender a outras pessoas aquelas informações adquiridas (SETTE et al., 2009).

A coleta, processamento, armazenamento, análise e distribuição das informações é um processo que demanda de uma rápida resposta dentro de um contexto. É correto afirmar que uma ferramenta que consiga realizar esta operação de uma forma satisfatória é praticamente uma agregação de valor para quem quer que tenha acesso (SILVA, 2011). Para realizar esses processos, as ferramentas e tecnologias adotadas são de grande auxílio, dentre várias existem os beacons que auxiliam na coleta das informações.

Os beacons são dispositivos que utilizam uma tecnologia chamada Bluetooth Low Energy (BLE) para se identificar e servir de marcador no interior de ambientes. Esses dispositivos são constantemente utilizados para monitoramento dentro de supermercados, e já foram utilizados em hospitais para identificação de pacientes e até mesmo para uso pessoal para as pessoas mais esquecidas (CARNEIRO, 2016). Dentre os modelos existentes, há o Eddystone que se destaca ao disponibilizar uma Uniform Resource Locator (URL) que é emitida por ele. Esse modelo se comunica nativamente com o Google Chrome para acesso dessa URL (PAIVA, 2016). Através do acesso a essa URL por parte do usuário é possível gerar dados de acesso ao beacon, fazendo assim um mapeamento de localização *indoor* do usuário e a montagem de um mapa de calor de uso dos beacons.

Um mapa de calor é uma demonstração numérica interpretada por cores (YAU, 2010). As cores representam números com diversos valores, sendo que esses valores são substituídos pelas cores mais marcantes para melhorar e facilitar a interpretação (YAU, 2010).

Diante do exposto, este trabalho apresenta o desenvolvimento de uma ferramenta para coleta de informações de localização *indoor* e montagem de um mapa de calor que representa

o uso dos beacons envolvidos no processo. Além disso, é disponibilizada através da ferramenta, uma forma para a geração de páginas HTML vinculada a cada beacon. A ferramenta também disponibilizará serviços web para requisições e recolhimento de acessos das páginas HTML de cada beacon.

1.1 OBJETIVOS

O objetivo deste trabalho é disponibilizar uma ferramenta para mapear uso de beacons em localização *indoor*.

Os objetivos específicos são:

a) disponibilizar uma ferramenta para a geração de HTML de acordo com o beacon escolhido;
b) disponibilizar uma ferramenta para visualização dos beacons mais visitados através de um mapa de calor;
c) disponibilizar uma estrutura dinâmica de exibição de páginas HTML de acordo com o beacon acessado.

1.2 ESTRUTURA

Este trabalho é dividido em quatro capítulos. O primeiro capítulo apresenta a introdução, justificativa para o trabalho desenvolvido, os objetivos e a definição de sua estrutura.

No segundo capítulo é apresentada a fundamentação teórica utilizada para o projeto. Dessa forma são abordados os conceitos de *bluetooth*, beacons e suas utilidades, monitoramento de pessoas e mapa de calor, além dos trabalhos correlatos. O terceiro capítulo apresenta o desenvolvimento do trabalho. Nele são detalhados os requisitos do trabalho desenvolvido e seus diagramas, assim como sua implementação e ferramentas utilizadas no processo. Nesse capítulo também é detalhado o processo de avaliação do trabalho e, por fim, são detalhadas as discussões e resultados obtidos durante o mesmo. O quarto e último capítulo aborda as principais conclusões. Ainda, são apresentadas as limitações e sugestões para futuros trabalhos.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais assuntos estudados para o desenvolvimento do trabalho. Na seção 2.1 é descrito sobre sinal *bluetooth*. Na seção 2.2 são apresentados os dispositivos beacons e seu funcionamento. Na seção 3.3.1.2 é abordado a respeito de monitoramento do fluxo de pessoas. Por fim, na seção 2.4, são apresentados os conceitos de mapa de calor.

2.1 SINAL BLUETOOTH

O sinal *bluetooth* surgiu em meados de 1994 com a intenção de criar uma tecnologia de comunicação entre dispositivos utilizando sinal de rádio de baixo consumo (ALECRIM, 2017). Na época as empresas que encabeçaram a pesquisa foram a Ericsson, Intel, IBM, Toshiba e Nokia, porém foram seguidas por outras que também tinham interesse no resultado (ALECRIM, 2017). O trabalho conjunto de todas essas empresas foi importante, pois se criou um padrão no sinal que foi seguido por todas elas (ALECRIM, 2017).

O sinal bluetooh é dividido em quatro classes, sendo as classes de 1 a 4, como é relacionado no Quadro 1. A classe 1 é normalmente usada em dispositivos que podem depender de longas distâncias para se comunicar, como é o caso de telefones sem fio (AMARIZ, 2016). A classe 2 é a mais utilizada atualmente, ela está presente nos smartphones, *tablets*, rádios de carros, fones de ouvido, etc. Ela se fez tão popular por conseguir unir a distância certa com o uso de bateria (AMARIZ, 2016). A classe 3 é utilizada em usos específicos como por exemplo mouses, teclados, etc., que não dependem de uma distância tão grande por normalmente o dispositivo conectado estar perto (AMARIZ, 2016). Já a classe 4 é focada em dispositivos que dependem de um consumo menor de energia e se comunicam em pequenas distâncias (ALECRIM, 2017).

Quadro 1 - Classes bluetooth

Classe	Potência(Mw)	Alcance(m)
1	100	100
2	2,5	10
3	1	1
4	0,5	0,5

Fonte: elaborado pelo autor.

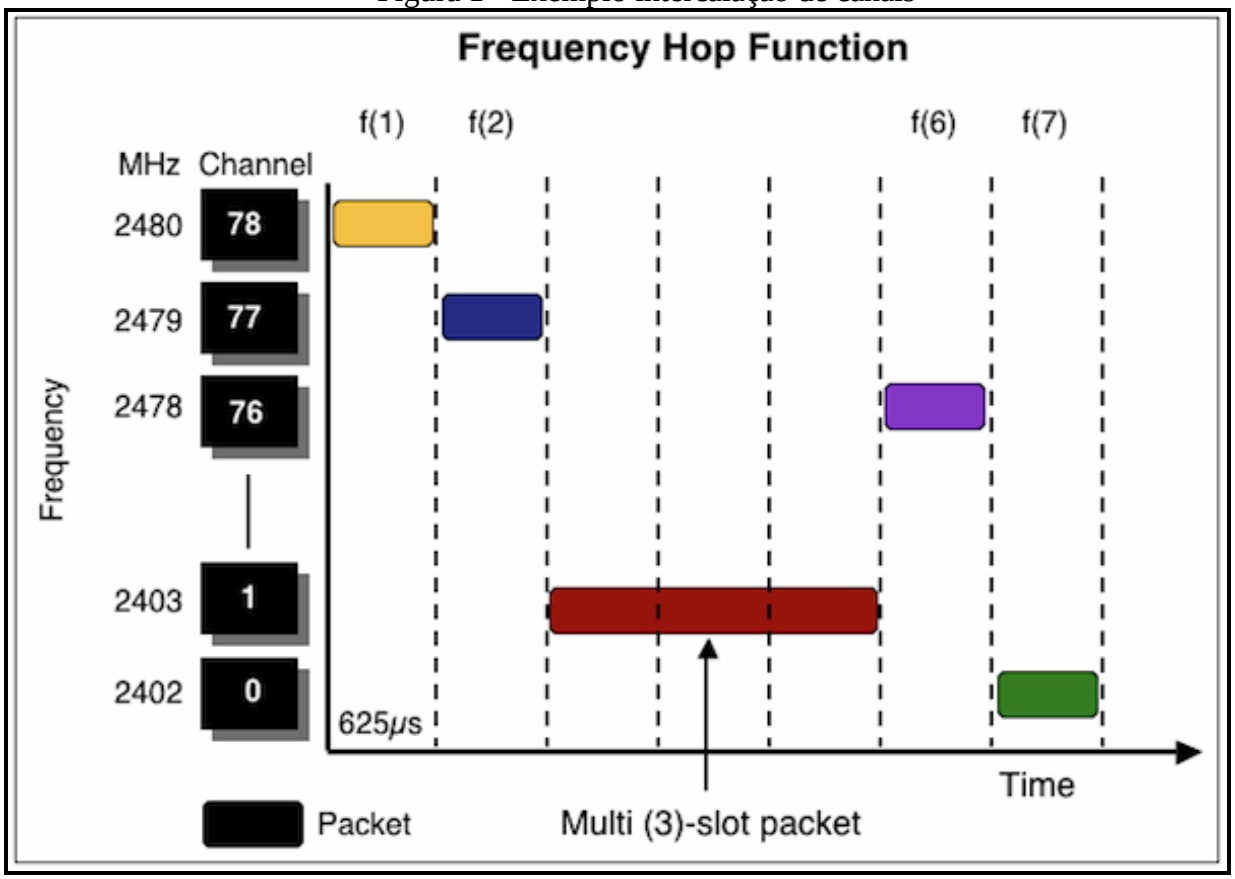
Inicialmente a taxa de transferência do sinal era razoavelmente baixa chegando no máximo a 1 Mb/s. Com o passar do tempo a tecnologia foi evoluindo, chegando hoje a até 50 Mb/s (ALECRIM, 2017). As próximas subseções abordam sobre a frequência transmitida pelo sinal *bluetooth*, bem como sobre as versões existentes.

2.1.1 Frequência

A tecnologia *bluetooth* foi desenvolvida para funcionar em todas as regiões do mundo, por adotar uma frequência que varia de 2,4 GHz a 2,5 GHz operando na faixa *Industrial, Scientific, Medical* (ISM) que é aberta e pode ser utilizada por qualquer sistema de comunicação. Para garantir que o sinal não seja muito afetado por interferência foi adotado um esquema de comunicação chamado Frequency Hopping - Code-Division Multiple Access (FH-CDMA), dividindo a frequência em vários canais (ALECRIM, 2017).

Essa divisão acaba abrindo a oportunidade para o dispositivo que se conectar a ela fique intercalando entre os canais de uma largura de banda muito pequena (ALECRIM, 2017). Podendo chegar até a 1 MHz de diferença, diminuindo as chances de interferência (ALECRIM, 2017). Como o *bluetooth* pode tanto receber quanto enviar dados, é feita uma alternância na transmissão adotando o esquema Frequency Hopping - Time Division Duplex (FH/TDD) (STAAB, 2013). O uso dos *slots* é determinado em períodos de 625 µs (microssegundos), onde cada salto de frequência feito ocupa um *slot*, resultando 1.600 saltos em questão de um segundo (ALECRIM, 2017). Conforme ilustra o gráfico da Figura 1, o eixo *x* representa o tempo de conexão via *bluetooth* e o eixo *y* os múltiplos canais criados para intercalação. O gráfico dividido entre *slots* demonstra o uso dos canais daquela banda, sendo que cada cor simboliza uma conexão momentânea e transferência de dados. Há um caso no qual um *slot package* ocupa 3 *slots*, isso ocorre quando há a real necessidade de uma transferência contínua de um pacote um pouco maior que os demais (STAAB, 2013).

Figura 1 - Exemplo intercalação de canais



Fonte: Staab (2013).

2.1.2 Versões bluetooth

Com o passar do tempo, o sinal *bluetooth* foi evoluindo e sofrendo melhorias nesse processo, assim como adaptações. Seu uso foi crescendo e com isso as necessidades acabaram surgindo e resultando em novas versões dessa tecnologia (ALECRIM, 2017). O Quadro 2 realiza uma comparação entre as versões.

Quadro 2 - Comparação versões bluetooth

Versão	Lançamento	Taxa de Transferência	Novidades
Bluetooth 1.0	Meados de 1994	721 kb/s	
Bluetooth 1.1	Fevereiro de 2001	721 kb/s	Estabelecido como padrão IEEE 802.15; Suporte ao RSSI.
Bluetooth 1.2	Novembro de 2003	721 kb/s	Conexões melhoradas; Otimização contra interferências; Suporte aperfeiçoado a *scatternets*; Processamento de voz.
Bluetooth 2.0	Novembro de 2004	3 Mb/s (2.1 Mb/s efetivos)	Melhor comunicação entre os dispositivos; Passou a contar com o padrão *EDR*.
Bluetooth 2.1	Agosto de 2007	3 Mb/s (2.1 Mb/s efetivos)	Mais informações nos sinais *Inquiry*; Melhorias na segurança; Melhoria no consumo de energia.
Bluetooth 3.0	Abril de 2009	24 Mb/s	Transmissões 802.11; Controle inteligente energia.
Bluetooth 4.0	Dezembro de 2009	24 Mb/s	Consumo menor de energia quando o dispositivo está ocioso.
Bluetooth LE (Low Energy)	Dezembro de 2009	1 Mb/s	Consumo bem menor de energia; Distância de até 30 metros;
Bluetooth 4.1	Final de 2013	24 Mb/s	Quase inativo se afastado de uma conexão; Melhorias em protocolos e parâmetros.
Bluetooth 4.2	Final de 2014	24 Mb/s	Maior tráfego de dados; Suporte ao uso da criptografia FIPS; Suporte a IPv6
Bluetooth 5	Final de 2016	50 Mb/s	Distância de até 40 metros; Diminuição de interferências de redes Wi-Fi ou LTE; Múltipla conexão com dispositivos;

Fonte: Alecrim (2017).

Como se pode observar no Quadro 2, conforme a tecnologia foi evoluindo e ganhando novas versões, o suporte a várias outras coisas como RSSI, IPV6 e FIPS foi sendo incorporado ao sinal bluetooth (ALECRIM, 2017). Outro aspecto que se pode notar é que a questão sobre o consumo de energia sempre foi algo importante, pois praticamente em toda versão houve uma melhoria nesse quesito, assim como a preocupação com segurança e qualidade do sinal (FARNELL, 2017). Além disso, a taxa de transmissão foi ficando cada vez maior para melhor atender as necessidades que foram surgindo (ALECRIM, 2017).

Dentre todas as versões apresentadas, uma foi lançada especificamente focada em baixo consumo de energia e comunicação utilizando pouco tráfego de dados (ALECRIM, 2017), sendo a versão Bluetooth Low Energy (BLE). Essa versão surgiu para atender dispositivos compactos e com baixa capacidade de bateria, como os portáveis e pulseiras inteligentes (PESSOA, 2016). O BLE consegue ficar boa parte do tempo ocioso e transmitir

os dados necessários em questão de milissegundos. Ainda, quanto menor a distância do dispositivo, menos energia ele gasta para transmitir algo (ALECRIM, 2017).

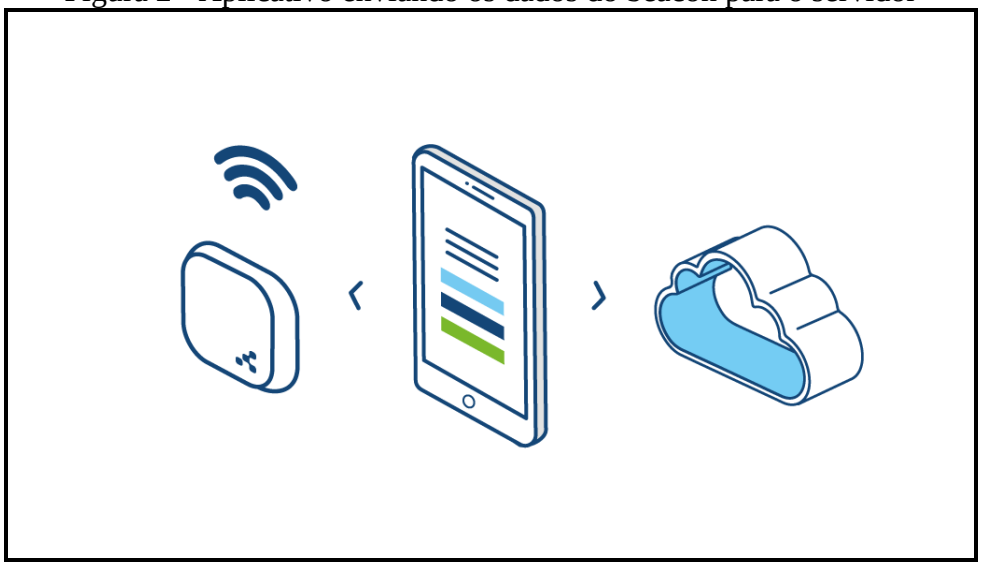
2.2 BEACONS E SEU FUNCIONAMENTO

Beacons são dispositivos que vem ganhando cada vez mais destaque pelo seu uso em diversos setores, entre eles o varejo (ENDEAVOR, 2015). Outra característica que o destaca é sua relação com a internet das coisas (LAGE, 2016). O dispositivo permite localizar objetos dentro de ambientes fechados, com isso é possível determinar a localização de uma pessoa nesse espaço, por exemplo, desde que se tenha um smartphone com ela (TEIXEIRA, 2014).

O funcionamento do dispositivo se dá por conta da tecnologia Bluetooth Low Energy (BLE) que é um sinal emitido pelo beacon para os dispositivos receptores próximos, podendo ser smartphones, *smartwatches*, pulseiras inteligentes, entre outros (CARNEIRO, 2016). Esse sinal transmite um identificador do beacon que é recebido igualmente para os aparelhos ao seu redor (TEIXEIRA, 2014).

O beacon por si só não envia dados ao usuário e muito menos recebe dados dele (TEIXEIRA, 2014). O dispositivo funciona de modo passivo apenas emitindo o sinal e seu identificador, sendo que toda a inteligência está no aplicativo previamente instalado no smartphone (TEIXEIRA, 2014). Como exemplo, a Figura 2 mostra que após a identificação do beacon o smartphone notifica o servidor sobre a iteração e o servidor por sua vez realiza as operações referentes aquela iteração com aquele beacon específico (KONTAKTIO.IO, 2017).

Figura 2 - Aplicativo enviando os dados do beacon para o servidor



Fonte: Kontaktio.io (2017).

Como vantagens do uso desta tecnologia, tem-se que o uso do BLE traz uma economia na bateria dos dispositivos receptores, pois consome pouca energia se comparado a outras

tecnologias como Wi-Fi e GPS (BORGES, 2016). Além disso, ele é independente de dados da operadora (BORGES, 2016).

Outra vantagem deles é o alcance do sinal, pois conseguem facilmente percorrer 50 metros dentro do espaço fechado. Além disso, a implementação de aplicativos para acessa-los é simples, pois é necessário apenas a autorização do usuário para acessar o bluetooth (BORGES, 2016).

Por outro lado, como desvantagens em utilizar o beacon, tem-se que ainda é necessário ter aplicativos pré-instalados para a utilização dos recursos que o beacon oferece (BORGES, 2016). Outro ponto é que a iteração depende de o aparelho estar com o bluetooth ligado. No entanto, uma pesquisa realizar por Teixeira (2014) indicou que somente 12% dos usuários de smartphones costumam andar com essa funcionalidade ligada.

2.2.1 Eddystone: o beacon melhorado

Eddystone é um modelo de beacon que utiliza o padrão eddystone de transmissão criado pelo Google (BABU, 2016). Ele supre toda a necessidade que o modelo iBeacon do beacon tradicional suporta e traz mais funcionalidades (GASIOREK, 2016).

A principal diferença entre iBeacon e Eddystone está nas informações transmitidas, enquanto o iBeacon transmite apenas o sinal UID, o Eddystosne além do UID trasmite uma URL chamada Eddystone URL, além das informações de telemetria do beacon (BABU, 2016). Outra diferença entre eles é que o Eddystone é *open-source* e o iBeacon não é (BABU, 2016).

Outro fator importe é a forte ligação entre Eddystone e *Physical web*, pois ajudam a impulsionar a tecnologia (GASIOREK, 2016). O conceito de *Physical web* é a existência de um ecossistema que tem como delimitadores balizas, sendo assim aquelas páginas somente existem naquele local físico (BABU, 2016). Através da Eddystone URL é possível acessar uma *Physical web local*, tirando a desvantagem de necessitar de aplicativo (BABU, 2016). Essa *Physical web* pode funcionar semelhante a um aplicativo só que não necessita de instalação, a não ser do Google Chrome (BABU, 2016).

2.3 MONITORAMENTO DE PESSOAS

O monitoramento de pessoas está relacionado a algum meio para rastrear indivíduos (LIFELINK, 2016). Isso traz consigo o fator de segurança e privacidade de quem está sendo monitorado, sendo importante ressaltar que o monitoramento ocorre com consentimento das partes envolvidas (LIFELINK, 2016).

Existem vários objetivos que podem ser cumpridos com o monitoramento, um deles é o de segurança, seja de pessoas que necessitam de mais atenção como crianças e idosos ou na parte de sistema penitenciário no qual o indivíduo monitorado é o detento (LIFELINK, 2016). No entanto, um ponto que está em alta é no setor de varejo físico, sendo que o monitoramento está sendo utilizado para coleta de dados do fluxo de clientes dentro das lojas (TAVARES, 2015).

O monitoramento do trajeto dos consumidores dentro dos estabelecimentos são informações precisas e valiosas (TAVARES, 2016). As informações obtidas do monitoramento auxiliam desde a disposição de produtos dentro da loja, como por exemplo, colocar produtos mais procurado lado a lado para aumentar suas vendas, até em criar *hot zones* para verificar as áreas do estabelecimento onde os clientes passam mais tempo ou costumam passar com mais frequência (TAVARES, 2015).

As tecnologias utilizadas para esse meio podem ir desde câmeras dentro das lojas, até etiquetas *Radio-Frequency IDentification* (RFID) (OLIVEIRA, 2015). Existem tecnologias como infravermelho e câmera 3D que fazem apenas a contagem de pessoas, tornando mais custosas e menos eficazes (DORES, 2016).

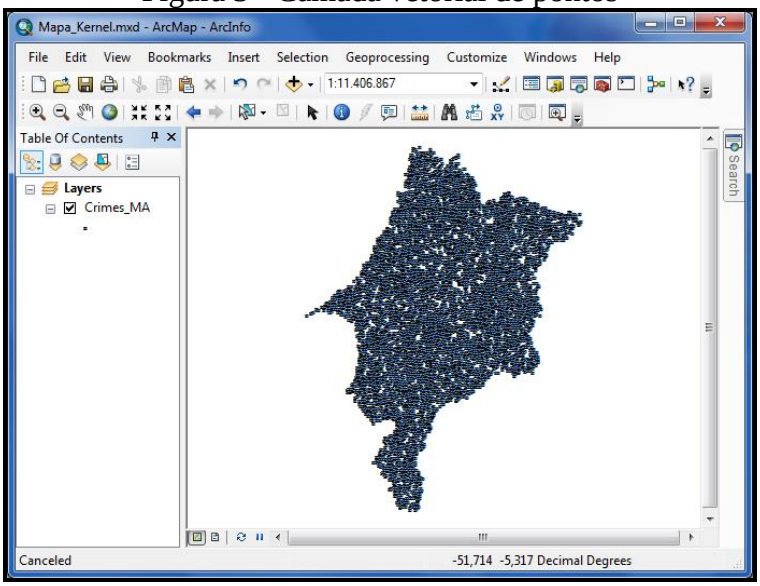
Empresas como a Hering já investem e utilizam essa abordagem para mensurar fluxo de pessoas em algumas de suas lojas (DORES, 2016). Informações como quantas pessoas realmente entraram na loja, quantas olharam o produto, mas não compraram quais locais mais frequentados, etc. Essas informações ajudam e muito na hora de redirecionar campanhas e realocação de funcionários para atender melhor as demandas de atendimento em horários que se foi mensurado anteriormente um fluxo maior de pessoas (DORES, 2016).

2.4 MAPA DE CALOR

Mapa de calor é uma representação de intensidade de dados em determinados pontos geográficos (GOOGLE, 2017). Também pode-se dizer que o mapa de calor é uma alternativa para análise de comportamento e padrões (MEDEIROS, 2012).

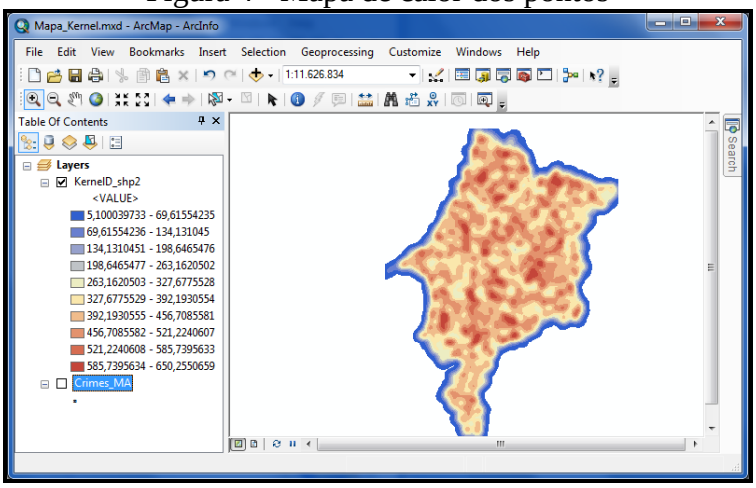
O mapa contém uma grande quantidade de pontos e cada ponto com valores individuais (ESRI, 2015). Se considerar a análise somente dos pontos individualmente, alguns poderiam se sobrepor ou até difícil entendê-los quando centenas são exibidos simultaneamente (ESRI, 2015). A conversão de pontos de valores individuais em mapa de calor pode ser exemplificada com o uso da ferramenta ArcMap, ela possibilita a inserção de camadas vetoriais conforme ilustra a Figura 3 em um mapa de calor demonstrado na Figura 4 (MEDEIROS, 2012).

Figura 3 - Camada vetorial de pontos



Fonte: Medeiros (2012).

Figura 4 - Mapa de calor dos pontos

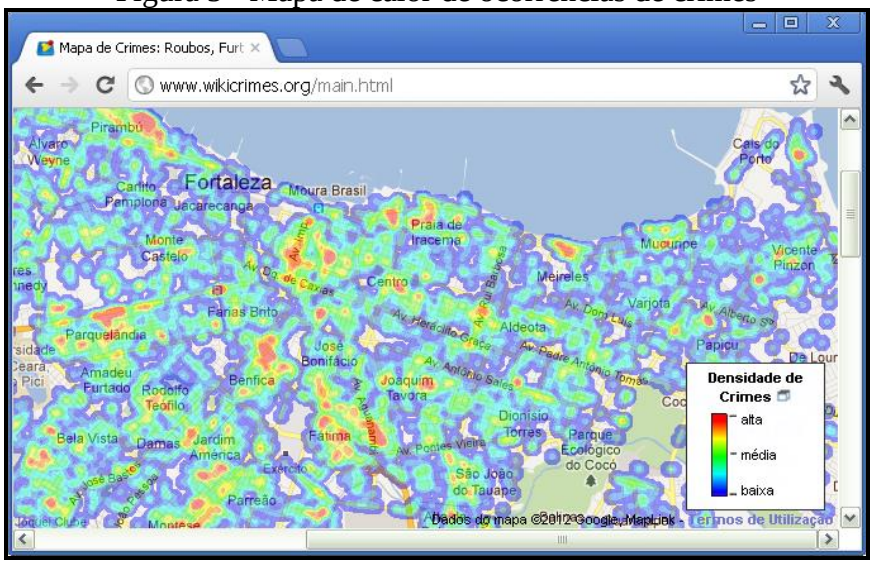


Fonte: Medeiros (2012).

Existem vantagens em se utilizar mapas de calor, uma delas é que quando há uma enorme concentração de pontos a ser analisada, as cores ajudam na interpretação (MEDEIROS, 2012). Outra vantagem é que os pontos extremamente próximos acabam se tornando apenas uma única interpretação quando visto todo o contexto do mapa, pois a intensidade dos pontos deixa a cor daquela área mais vermelha (ESRI, 2015).

Uma ferramenta que faz uma interpretação de mapa de calor baseado em pontos é o WikiCrimes (MEDEIROS, 2012). Essa ferramenta é colaborativa e mapeia as ocorrências de crimes em determinadas regiões do país (MEDEIROS, 2012). A Figura 5 é um exemplo de visão da ferramenta, na qual estão listadas as ocorrências da cidade de Fortaleza, no Ceará (MEDEIROS, 2012).

Figura 5 - Mapa de calor de ocorrências de crimes



Fonte: Medeiros (2012).

Mapas de calor também tem seu uso em espaços menores como é o caso das lojas de varejo, na qual o mapa ajuda a conhecer melhor o cliente e detectar melhorias na organização da loja (RODRIGUES, 2017). A Figura 6 demonstra como ficou o mapa de calor em uma loja da Bretaband em San Francisco, sendo que as áreas em vermelho são locais com maior circulação de clientes e as áreas em verde e azul apresentam os locais de baixa circulação de público (GRISWOLD, 2014).

Figura 6 - Demonstração mapa de calor em loja

Fonte: Griswold (2014).

A geração dos dados, nesse caso, foi feita através de câmeras e software de detecção da Prism Skylabs (GRISWOLD, 2014). Os dados contidos nas imagens do mapa de calor da

loja ajudam os varejistas a reorganizarem os leiautes de seus estabelecimentos de forma que melhore a venda de certos produtos (GRISWOLD, 2014).

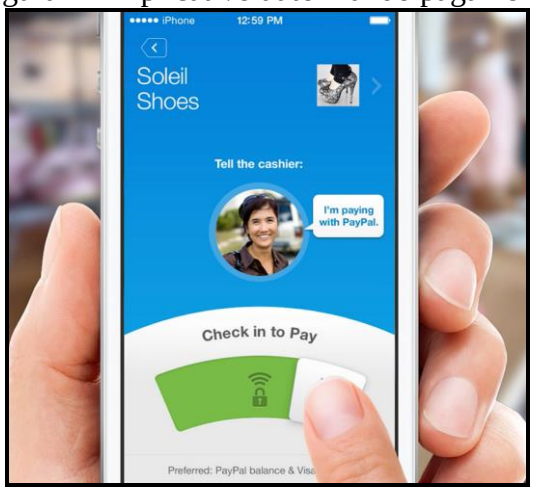
2.5 TRABALHOS CORRELATOS

Esta seção apresenta três trabalhos correlatos que possuem características semelhantes a este trabalho. A seção 2.5.1 detalha o PayPal Beacon (PAYPAL, 2009), que é uma função dentro do aplicativo padrão da empresa para realizar pagamentos. A seção 2.5.2 descreve o Microlocation (HANDCOM, 2014), um sistema de resgate de cupons focado para varejo com utilização de beacons. A seção 2.5.3 aborda o Launch Here (AWWAPPS, 2013), que utiliza a localização *indoor* baseado em *beacons* para lembretes básicos do dia a dia em seu lar. Por fim, a seção 2.5.4 correlaciona os trabalhos, apresentando as comparações entre os mesmos.

2.5.1 PAYPAL BEACON

O PayPal Beacon é um facilitador para o usuário na hora de realizar o seu pagamento. Com ele, o usuário precisa fazer *check-in* no estabelecimento (Figura 7) somente na primeira vez que o frequenta, e nas próximas vezes, sem precisar pegar a carteira ou o smartphone, basta fazer o pedido e o valor é descontado diretamente da conta do PayPal (VENTURA, 2013).

Figura 7 - Aplicativo autorizando pagamento

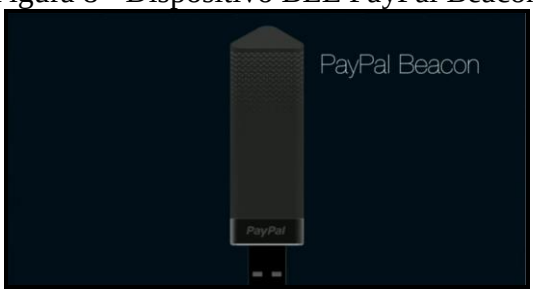


Fonte: Nanrata Mruthinti (2015).

O funcionamento é baseado no aparelho mostrado na Figura 8 que emite um sinal BLE identificando o estabelecimento. Quando o usuário entra na loja com o aplicativo do PayPal pré-instalado, ele terá de fazer *check-in* no local uma única vez. Esse procedimento é feito somente para garantir que o usuário realmente permita as cobranças de pedidos feitos por aquela loja. Após a conclusão do *check-in* o aparelho se conecta ao sinal e o usuário pode realizar todos os pagamentos naquele estabelecimento de forma simples e prática. O

pagamento é feito com uma confirmação verbal para o atendente que cadastra o pedido, o aplicativo autoriza e envia o recibo para o e-mail do usuário (VENTURA, 2013).

Figura 8 - Dispositivo BLE PayPal Beacon



Fonte: Nanrata Mruthinti (2015).

2.5.2 MICROLOCATION

A Microlocation é uma plataforma de *marketing* baseado em microlocalização com o intuito de interagir com consumidores próximos ao estabelecimento e também dentro dele. A interação se faz quando o consumidor está próximo a pontos do estabelecimento. Ele é notificado com mensagens contextualizadas dos produtos e promoções. Esse tipo de interação possibilita ações como *marketing* de proximidade, *call to action* no estabelecimento, mapas de circulação e estatísticas de *e-commerce* para o estabelecimento (MICROLOCATION, 2013).

O serviço é oferecido a partir de quatro produtos, o primeiro é uma plataforma *online* para o vendedor administrar os conteúdos das promoções, cupons e produtos enviados aos *smartphones*. O segundo é um aplicativo chamado Encart.es para a interação com o cliente. A Figura 9 possui um exemplo desses dois primeiros produtos, sendo que o primeiro se encontra na esquerda e o segundo na direita. O terceiro é uma Application Programming Interface (API) para customização desse aplicativo. Por fim, a empresa oferece *hardwares* para fazer a comunicação utilizando BLE (Figura 10) com os aparelhos e o software da loja (BERNARDO, 2014).

Figura 9 - Exemplificação de serviços

Fonte: Bernardo (2014).

Figura 10 - *Beacon* Microlocation

Fonte: Microlocation (2013).

A ideia do aplicativo disponibilizado pela Microlocation conforme exemplifica a tela do smartphone na Figura 9 é que o cliente entre na loja e conforme ande pelos corredores haja a interação dele com as promoções e/ou produtos oferecidos pela loja em seu smartphone. Essa interação ocorre quando são posicionados os beacons na loja conforme demonstrado na Figura 11. Através da distância de cada beacon o cliente pode ter acesso a promoções e divulgações dos produtos, exemplificado através da Figura 12, sendo basicamente uma versão digital dos panfletos. Com isso, o próprio dono das lojas podem obter informações geradas como quantidade de visualizações, quantos cupons foram pegos e quantas pessoas utilizaram os cupons de desconto adquiridos através dos beacons, assim como ilustra as informações de cada beacon na Figura 13, dando possibilidade de verificar quais estratégias de *marketing* funcionariam melhor naquele ambiente (BERNARDO, 2014).

Figura 11 - Posicionamento dos Beacons

Fonte: Microlocation (2014).

Figura 12 - Visão do cliente

Fonte: Microlocation (2014).

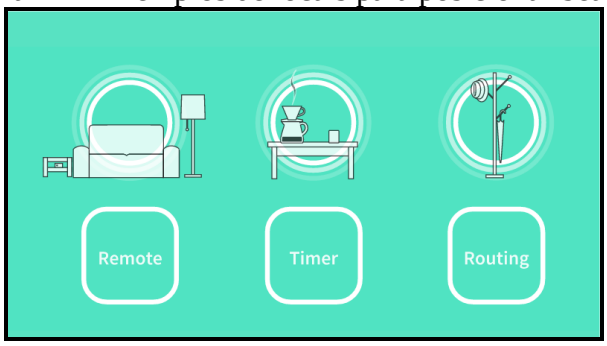
Figura 13 - Estatísticas geradas

Fonte: Microlocation (2014).

2.5.3 LAUNCH HERE

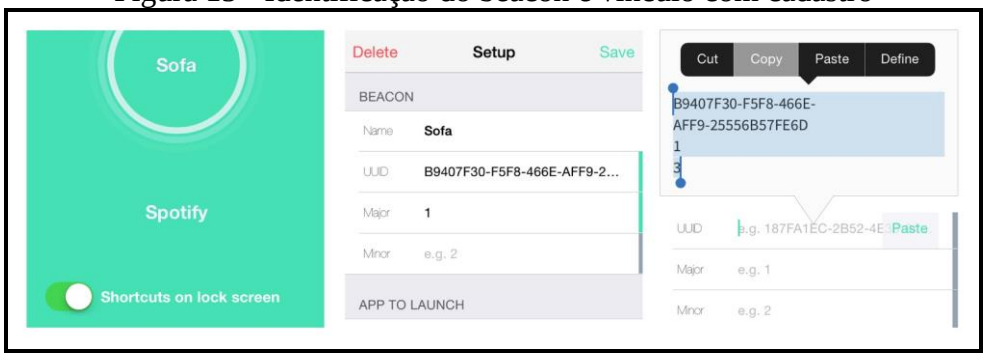
O Launch Here fornece uma maneira de vincular locais da casa à temporizadores e notificações. Para iniciar o uso, basta posicionar os beacons em locais onde regularmente é necessário ter acesso rápido a qualquer aplicativo, exemplificado pela Figura 14. Após isso, deve-se identificar o ID do beacon e vincular ao local que ele está posicionado, assim como demonstra a Figura 15 na qual o *beacon* está sendo configurado com o aplicativo customizável a ser executado (LAUNCH HERE, 2015).

Figura 14 – Exemplos de locais para posicionar beacons



Fonte: Launch Here (2015).

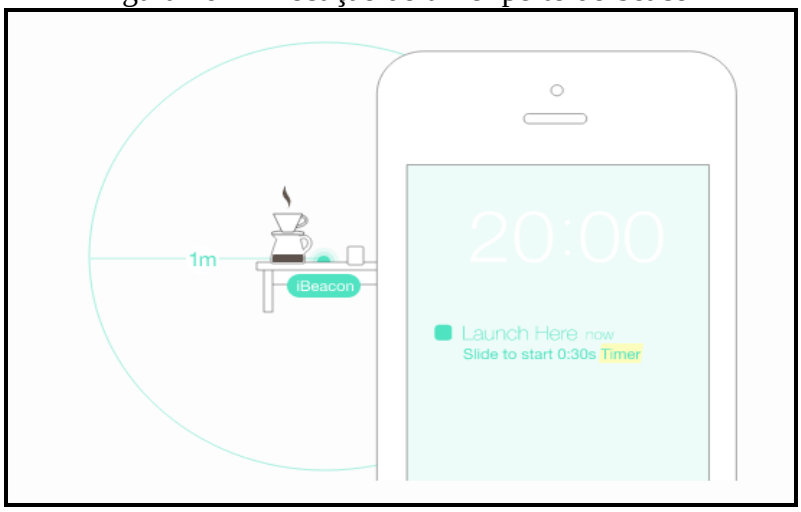
Figura 15 - Identificação do beacon e vínculo com cadastro



Fonte: Awwapps (2013).

Com a ferramenta, ainda é possível realizar uma configuração baseada em situações. Por exemplo, após um determinado tempo de espera perto do beacon é disparado um *timer*, após o tempo configurado o alarme é ativado conforme demonstra a Figura 16, ou pode-se utilizar a notificação mais simples é que é quando o usuário passe várias vezes no mesmo local ele ativa uma notificação. Dessa forma, ao se aproximar do *beacon*, o usuário pode ser lembrado de algo (AWWAPPS, 2013).

Figura 16 - Execução do timer perto do beacon



Fonte: Awwapps (2014).

2.5.4 Comparação entre os trabalhos correlatos

A seguir é demonstrado no Quadro 3 a comparação entre os trabalhos correlatos. Percebe-se, que todos os trabalhos demonstram um uso claro e objetivo de beacons vinculados a aplicativos pré-estabelecidos e com informações fixas por identificação dos mesmos.

Quadro 3 - Comparativo entre trabalhos relacionados

Caracterísitcas / Trabalhos correlatos	**PayPal Beacon (2009)**	**Microlocation (2014)**	**Launch Here (2013)**
Utiliza beacons como marcadores	X	X	X
Realiza ação após identificar beacon		X	X
Permite customizar ação realizada			X
Geração de informações estatísticas	X	X	
Armazena acessos e a localização do beacon		X	
Não necessita de aplicativo instalado			

Fonte: elaborado pelo autor.

Conforme tem-se no Quadro 3, os aplicativos apresentados atendem em partes as características relacionadas, sendo requisitos do trabalho desenvolvido. Sobre o quesito de utilizar beacons como marcadores *indoors* todos atendem com perfeição. Sobre as ações ou tomadas de decisões após reconhecer os beacons cada um atende em partes, sendo por execuções de aplicativos externos como é o caso do Launch Here ou por execuções fixas como é o caso dos demais correlatos, não é permitido customização direta das ações através da própria aplicação. Outro ponto a ser observado é que todos necessitam de aplicativo pré-instalado para poder operar e realizar suas tarefas.

3 DESENVOLVIMENTO

Neste capítulo são abordadas as etapas e detalhes do desenvolvimento da ferramenta, assim como suas especificações e implementação. Na seção 3.1 são enumerados os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) da ferramenta. Na seção 3.2 é apresentada a especificação do modelo e da ferramenta desenvolvida. Na seção 3.3 é detalhada a implementação da ferramenta, além dos principais algoritmos utilizados para a sua construção e a operacionalidade da implementação. Por fim, na seção 3.4 são apresentados os testes realizados e discussão dos resultados obtidos.

3.1 REQUISITOS

Nesta seção são listados os requisitos do trabalho desenvolvido, contemplando requisitos funcionais e não funcionais. O Quadro 4 apresenta os Requisitos Funcionais (RF). Já o Quadro 5 apresenta os Requisitos Não Funcionais (RNF).

Quadro 4 - Requisitos funcionais

Requisitos funcionais
RF01: A ferramenta deverá exibir ao usuário uma página web vinculada ao beacon.
RF02: A ferramenta deverá enviar data e hora de acesso do beacon ao servidor Web.
RF03: A ferramenta deverá disponibilizar serviços para recepção de data e hora de acesso dos beacons.
RF04: A ferramenta deverá disponibilizar serviços de consulta a páginas editadas/estruturadas para cada beacon.
RF05: A ferramenta deverá conter uma autenticação de usuário.
RF06: A ferramenta deverá permitir manter cadastro de usuários.
RF07: A ferramenta deverá permitir manter o cadastro de beacons.
RF08: A ferramenta deverá manter o histórico de acesso dos beacons.
RF09: A ferramenta deverá permitir vincular o beacon a um lugar físico num mapa.
RF10: A ferramenta deverá permitir editar uma página HTML para exibição de informações e componentes web.
RF11: A ferramenta deverá permitir a pré-visualização da página estruturada.

Fonte: elaborado pelo autor.

Quadro 5 - Requisitos não funcionais

Requisitos não funcionais
RNF01: A ferramenta deverá ser desenvolvida na linguagem de programação Java.
RNF02: A ferramenta deverá ser desenvolvida utilizando o *framework* Spring.
RNF03: A ferramenta deverá ser disponibilizada através de serviços no protocolo HTTP.
RNF04: A ferramenta deverá se comunicar com banco de dados PostgreSQL.
RNF05: A ferramenta deverá manter a estrutura das páginas HTML dinâmicas no banco de dados em formato JSON.
RNF06: A ferramenta deverá utilizar o *framework* AngularJS para interpretar o JSON e gerar páginas HTML.

Fonte: elaborado pelo autor.

3.2 ESPECIFICAÇÃO

Esta seção aborda a especificação da ferramenta desenvolvida, assim como os seus detalhamentos e diagramações. Na seção 3.2.1 é apresentado o diagrama de casos de uso, assim como são detalhados os Casos de Uso (UC) mais relevantes. Na seção 3.2.2 é apresentada a especificação da ferramenta desenvolvida, através do diagrama de classes.

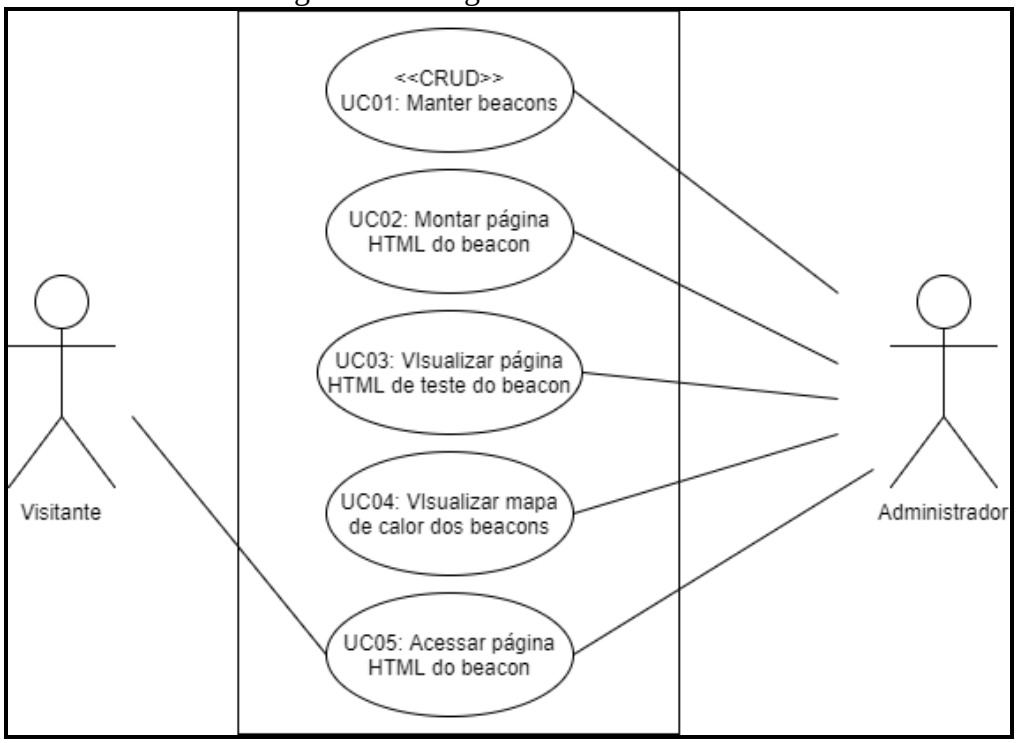
3.2.1 Casos de uso

A Figura 17 apresenta o diagrama de casos de uso da ferramenta. No diagrama de casos de uso são representados os atores que possuem um papel relevante na ferramenta, assim como sua interação com o sistema, sendo eles: visitante e administrador.

O ator visitante representa o usuário final que utiliza a URL disponibilizada pelo beacon para visualização e iteração com a ferramenta. Ele possui privilégios de acessar e visualizar a página vinculada ao beacon acessado.

O ator administrador tem como objetivo realizar a configuração e montagem do conteúdo que o usuário comum poderá visualizar e interagir dentro da ferramenta. Assim, ele possui privilégios de vínculo de URL com os beacons, montagem das páginas HTML de cada beacon e visualização do mapa de calor gerado a partir dos acessos das páginas HTML. Desta forma, este usuário representa também o pilar de gerenciador de conteúdo dentro da ferramenta.

Figura 17 - Diagrama de casos de uso



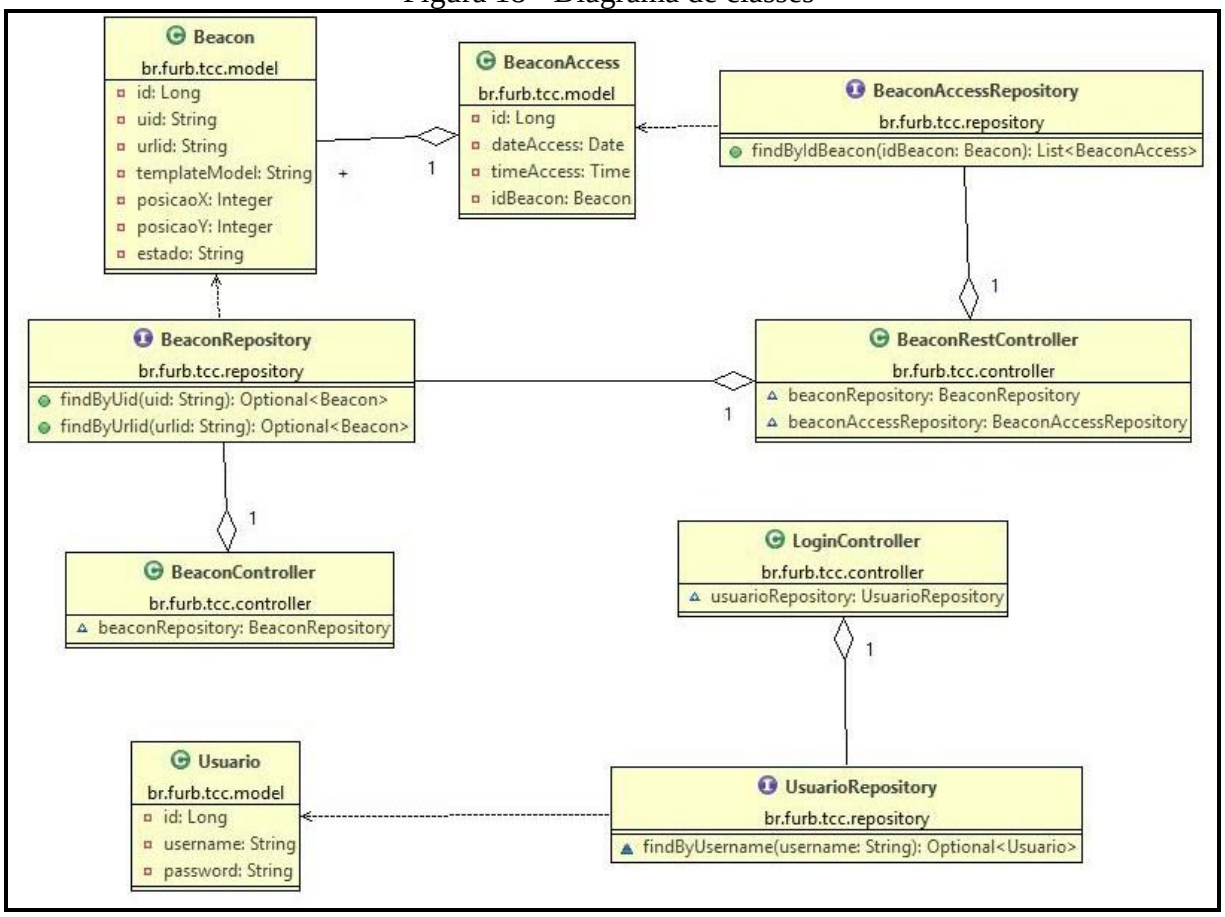
Fonte: elaborado pelo autor.

Os principais casos de uso são manter beacons, montar página HTML do beacon, visualizar mapa de calor dos beacons e acessar página HTML do beacon. Os detalhamentos dos casos de usos citados podem ser visualizados no Apêndice A.

3.2.2 Diagrama de classes

A Figura 18 apresenta o diagrama de classes que abrange o domínio da ferramenta construída.

Figura 18 - Diagrama de classes



Fonte: elaborado pelo autor.

A classe `Beacon` é a principal classe do modelo da aplicação e é responsável por representar um beacon cadastrado pelo usuário administrador e acessado pelo usuário comum, sendo utilizado como referência aos acessos realizados. Essa classe possui relacionamentos com a classe `BeaconAccess` que representa os acessos daquele beacon. Esses acessos são a base de informações para a geração do mapa de calor. A classe `Usuario` representa o usuário que é o administrador da ferramenta. Ela é responsável por garantir e validar o administrador para o gerenciamento dos dados dos beacons.

Além das classes de modelagem citadas, há as classes que controlam a persistência dos dados no banco de dados, esses são: `BeaconRepository`, `BeaconAccessRepository` e

`UsuarioRepository` que gerenciam todo o acesso e busca de informações dos beacons, acessos aos beacons e de usuários respectivamente. Outras classes importantes de se citar são as `controllers` que gerenciam toda a requisição, alteração ou inserção de informações dos beacons e de seus acessos da camada `view` da ferramenta.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as técnicas e ferramentas utilizadas para a construção desta ferramenta, as principais rotinas e seus códigos-fontes desenvolvidos, assim como a operacionalidade da implementação. Essa seção refere-se à ferramenta como todas as partes tecnológicas do projeto, incluindo a ferramenta desenvolvida, o *back-end* da aplicação e também o banco de dados utilizado. Junto às técnicas e ferramentas utilizadas é apresentada a arquitetura da aplicação, através do Diagrama de Componentes da UML.

3.3.1 Técnicas e ferramentas utilizadas

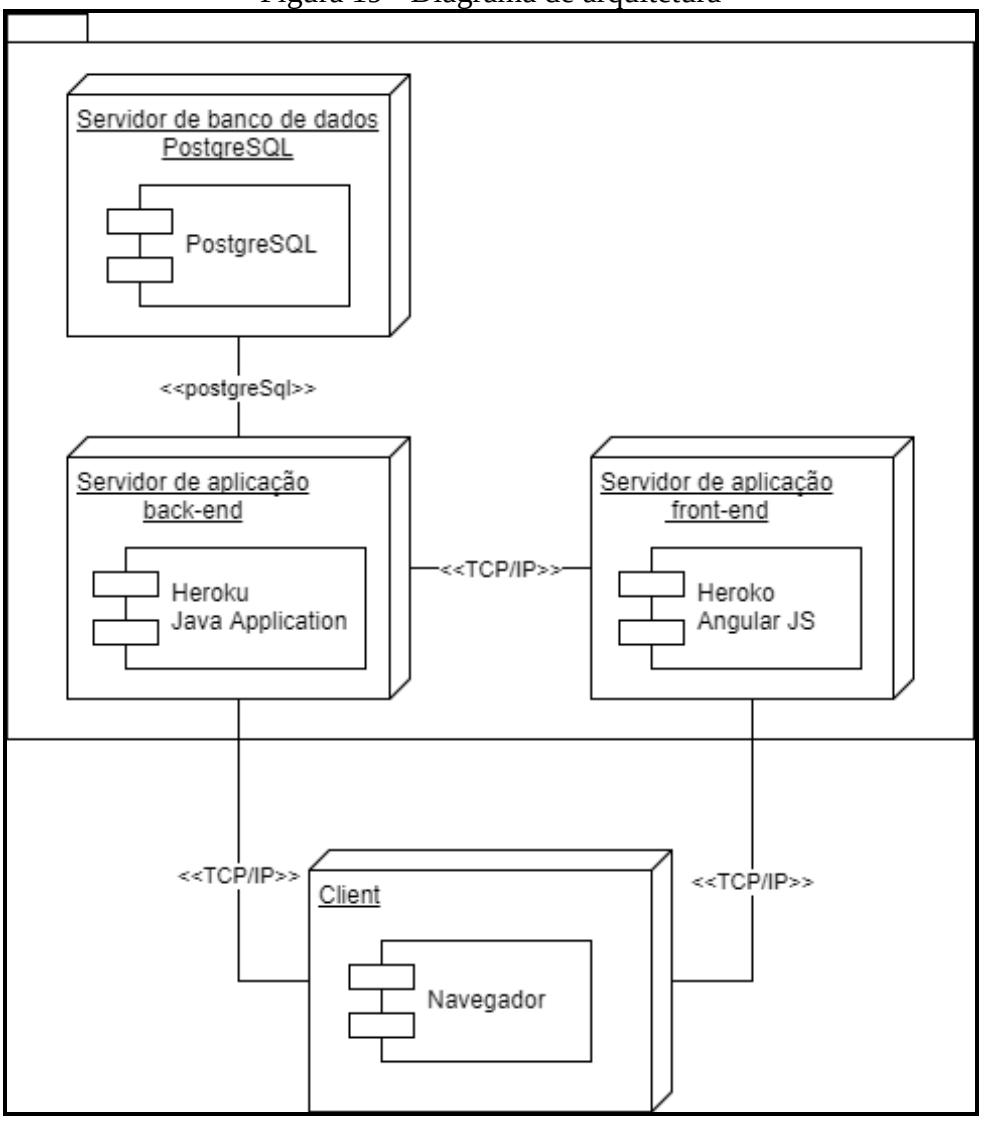
A ferramenta foi desenvolvida utilizando a linguagem de programação Java juntamente com Spring *framework*, detalhado na seção 3.3.1.1, bem como com Java Server Pages (JSP), *taglibs* e JavaScript para a parte de cadastros e geração de mapa de calor. Já para a parte de geração de página HTML e acesso da página para o usuário final foi utilizado o Angular JS que é detalhado melhor na seção 3.3.1.2. Na construção da comunicação do servidor foi utilizada também a linguagem de programação Java com auxílio do Spring framework. O banco de dados utilizado para manipulação dos dados foi o banco relacional PostgreSQL.

Com base nessas ferramentas, a Figura 19 apresenta o diagrama de arquitetura da ferramenta construída. Nesse diagrama é ilustrado cada módulo da ferramenta, sendo estes o servidor de banco de dados PostgreSQL, o servidor da aplicação *back-end*, o servidor da aplicação *front-end* e o cliente.

O módulo servidor de banco de dados PostgreSQL contém o serviço de banco de dados PostgreSQL, responsável por hospedar e manipular toda a estrutura do banco de dados relacional da ferramenta. Esse módulo é diretamente ligado com o servidor da aplicação *back-end*. No módulo servidor da aplicação *back-end* tem-se a aplicação do Heroku Java Application, no qual é localizado o servidor principal da ferramenta e também toda a inteligência para gerenciamento e geração de mapa de calor da ferramenta. No módulo servidor da aplicação *front-end* tem-se a estrutura web construída para a aplicação funcionar em navegadores web, sendo o componente Angular JS responsável pela geração dinâmica de

todo o *front-end* para o usuário final. Ainda, tem-se o módulo cliente, responsável pela execução da aplicação no navegador do dispositivo.

Figura 19 - Diagrama de arquitetura



Fonte: elaborado pelo autor.

3.3.1.1 Spring framework

Spring é um *framework open source* criado por Rob Johnson em 2002 (GENTIL, 2016). O *Spring* é composto de módulos, cada um com sua função e particularidade (CARVALHO, 2006).

O principal módulo é o `spring core` que é responsável por mapear os componentes através de anotações, conforme pode ser visto na Figura 20, a anotação `@Component` faz o mapeamento da classe `Cliente`. Esse módulo é responsável por manter apenas uma instância *singleton* de cada componente mapeado, distribuindo seu uso nos dependentes (GENTIL, 2016). Outro módulo importante é o `spring dao` responsável pela abstração e interação com

o JDBC, além da manipulação dos dados persistidos que também é feito através de anotações conforme ilustra a Figura 21. Nesse código-fonte é utilizada a anotação `@Repository` para o módulo registrar a classe `ClienteDao` como um gerenciador de dados da classe `Cliente` no banco de dados da aplicação (CARVALHO, 2006).

Figura 20 - Exemplo anotação de componente

```
package br.com.cliente.entity;

import org.springframework.stereotype.Component;

@Component
public class Cliente {

	private String nome;

	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
}
```

Fonte: Allet (2016)

Figura 21 - Exemplo de anotação para repositório

```
package br.com.cliente.dao;

import org.springframework.stereotype.Repository;
import br.com.cliente.entity.Cliente;

@Repository
public class ClienteDao
{
	public void save(Cliente cliente) {
		System.out.println("Cliente salvo no Dao: "+cliente.getNome());

	}
}
```

Fonte: Allet (2016)

Os módulos do spring responsáveis pela comunicação e iteração com o usuário são respectivamente o `spring web` e `spring mvc`. Com o `spring web` é possível criar serviços web de forma bem simplificada e prática através de anotações. Conforme demonstrado na Figura 22 é utilizada a anotação `@Service` na classe `ClienteService` para mapeá-la como um serviço da aplicação (). Já o `spring mvc` disponibiliza todo o *framework* para desenvolvimento web.

Figura 22 - Exemplo anotação serviço web

```
package br.com.cliente.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import br.com.cliente.dao.ClienteDao;
import br.com.cliente.entity.Cliente;

@Service
public class ClienteService {

        @Autowired
        ClienteDao clienteDAO;
        @Autowired
        Cliente cliente;

        public void save() {
                cliente.setNome("Leonel Messi");
                System.out.println("Cliente foi manipulado no Service");
                clienteDAO.save(cliente);
        }
}
```

Fonte: Allet (2016)

3.3.1.2 Angular JS

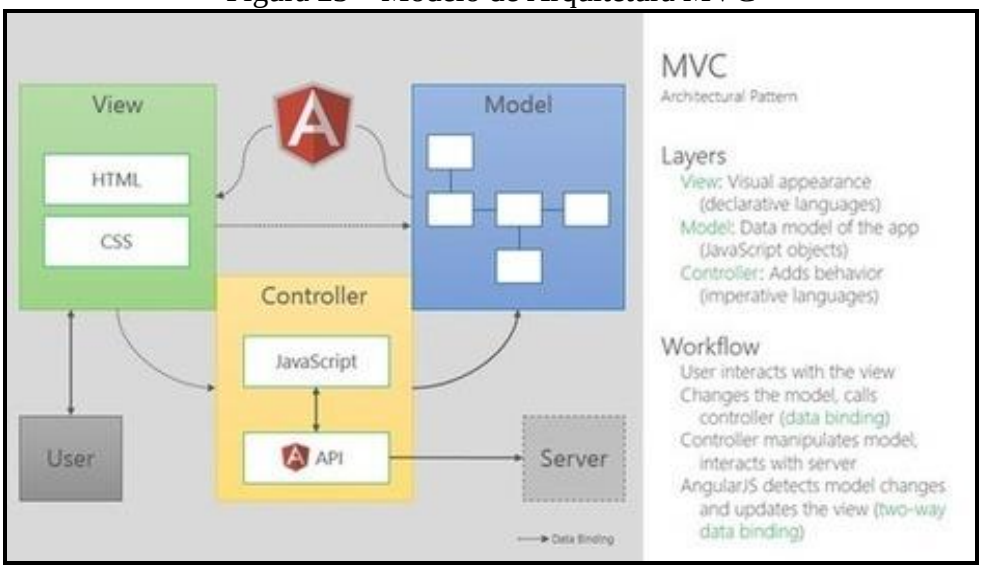
Angular JS é um *framework* JavaScript baseado na arquitetura Model View Whatever (MVW), foi criado e está sendo mantido pela Google (FEITOSA, 2015). Model View WhatEver é o termo utilizado quando o *framework* consegue trabalhar em diversas arquiteturas, como no caso do Angular JS que é capaz de utilizar Model View Presenter (MVP), Model View Viewmodel (MVVM) e Model View Controller (MVC) (RAUH, 2015). O Angular JS veio para complementar o HTML, que por sua vez transformou páginas que antes eram estáticas em páginas mais maleáveis e dinâmicas (BARBOSA, 2016).

A flexibilidade dada pelo *framework* possibilita a criação de componentes customizados e de elementos para uso e reaproveitamento em *templates*, possibilitando a fácil modularização e reuso dos códigos-fontes (BARBOSA, 2016). Outra grande vantagem do Angular JS é a facilidade de controlar e injetar *templates* e *controllers* dependendo da navegação do usuário (FEITOSA, 2015).

O Angular JS se encontra na camada de *controller*. É nessa camada fica toda a lógica da aplicação e também da comunicação com o servidor, já que o *framework* é totalmente desacoplado do servidor (BARBOSA, 2016). Como exemplo é utilizado o modelo de arquitetura MVC conforme ilustra a Figura 23, o Angular JS se encontra na camada *controller*

atuando juntamente com o JavaScript da página, entretanto ele mantém as camadas *view* e *model* sincronizadas através das diretivas disponibilizadas pelo *framework* em ambas as camadas (FEITOSA, 2015).

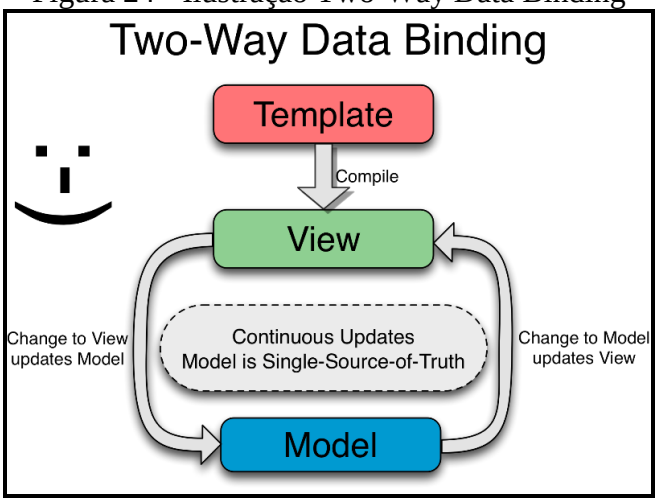
Figura 23 – Modelo de Arquitetura MVC



Fonte: Barbosa (2016).

Pode-se dizer que a maior vantagem do Angular JS é o chamado *Two-Way Data Binding*. Essa *feature* facilitou o trabalho de sincronização dos dados entre a *view* e o *controller* (FEITOSA, 2015). Através de diretivas e expressões é possível alterar o HTML da página dinamicamente. Conforme ilustra a Figura 24, após a interpretação do *framework* baseado no *template* executado, ele mantém uma constante atualização das camadas *view* e *controller* para operacionalidade das expressões e diretivas, sem a necessidade de ficar atualizando os valores manualmente nas camadas com a preocupação de consistência da informação exibida e manipulada.

Figura 24 - Ilustração Two-Way Data Binding



Fonte: Barbosa (2016).

3.3.2 Desenvolvimento da ferramenta

O desenvolvimento da ferramenta ocorreu em quatro etapas, cada uma com suas validações. Na primeira etapa apresentada na seção 3.3.2.1, foi construída toda a modelagem de banco de dados juntamente com a tela de cadastro do beacon. Na segunda etapa detalhada melhor na seção 3.3.2.2, foi realizada toda a implementação do gerador de página HTML juntamente com o visualizador da página vinculada ao beacon. Na terceira etapa explicada na seção 3.3.2.3, o foco se tornou a comunicação e ligação das duas outras etapas realizadas anteriormente, na qual foram criados os serviços para o vínculo com os beacons e consulta de estruturas das páginas HTML. Na quarta e última etapa vista melhor na seção 3.3.2.4, foi trabalhado na segurança no gerenciamento dos dados e a geração do mapa de calor.

3.3.2.1 Modelagem e cadastro da ferramenta

Toda a modelagem de dados foi feita utilizando anotações em classes Java, de modo que as tabelas do banco de dados e seus atributos fossem criados e atualizados de forma automática pela aplicação. Na Figura 25 é demonstrado parte do código-fonte da classe beacon e seus atributos, no qual a anotação `@Entity` na linha 12 sinaliza aquela classe como sendo uma entidade de banco, juntamente com a anotação `@Table`. Essa anotação, disponível na linha 13, confere a qual tabela a entidade é mapeada, sendo que nesse caso foi mapeada a tabela `beacon` do banco de dados. Outra anotação utilizada foi a que realiza a ligação de atributos da classe a campos da tabela como é o caso das linhas que contém a anotação `@Column`. Também foi utilizada anotações para campos específicos como é o caso da linha 16 que contém a anotação `@Id` que indica o `id` único da tabela e a linha 28, com a anotação `@Lob` que sinaliza o tipo do campo como `LOB` no banco de dados.

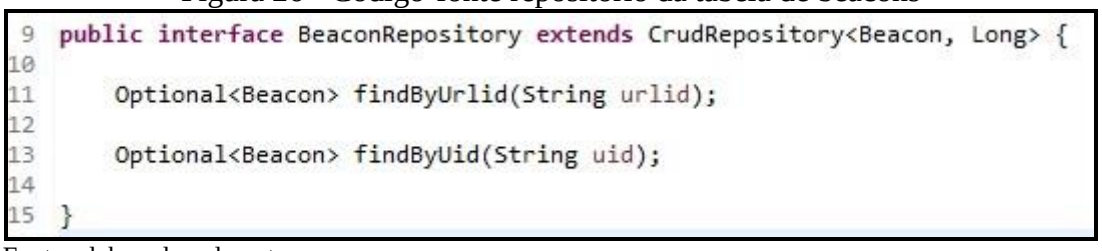
Para tratar a busca e gravação dos dados no banco foi criada uma `interface` e estendido da `interface CrudRepository` do `spring dao`. A `interface CrudRepository` contém todos os métodos e abstração de configuração de banco e mapeamento necessário já prontos. Na Figura 26 é demonstrado como é realizado esse passo, na linha 9 é realizado o `extends` da interface comentada, sinalizando a entidade que será manipulada, sendo nesse caso a classe `Beacon`. Além disso, é indicado qual é o tipo do `id` único da tabela que no caso do `Beacon` é `Long`. A Figura 26 ainda contém os métodos de busca. Na linha 11 há o método que faz a busca pelo atributo `urlid` e na linha 13 o método que busca pelo `uid` do `Beacon`.

Figura 25 - Código-fonte modelagem da tabela de beacons

```
12 @Entity
13 @Table(name = "beacon")
14 public class Beacon {
15
16     @Id
17     @GeneratedValue(strategy = GenerationType.AUTO)
18     @Column(name = "id", updatable = false, nullable = false)
19     private Long id;
20
21     @Column(name = "uid")
22     private String uid;
23
24     @Column(name = "urlid")
25     private String urlid;
26
27     @Column(name = "template_model")
28     @Lob
29     private String templateModel;
30
31     @Column(name = "posicao_x")
32     private Integer posicaoX;
33
34     @Column(name = "posicao_y")
35     private Integer posicaoY;
36
37     @Column(name = "estado")
38     private String estado;
```

Fonte: elaborado pelo autor.

Figura 26 - Código-fonte repositório da tabela de beacons



```
 9 public interface BeaconRepository extends CrudRepository<Beacon, Long> {
10
11     Optional<Beacon> findByUrlid(String urlid);
12
13     Optional<Beacon> findByUid(String uid);
14
15 }
```

Fonte: elaborado pelo autor

Outra tabela modelada foi a de acessos dos beacons. Na Figura 27 é demonstrado parte do código-fonte da classe `BeaconAccess` e seus atributos. Na linha 16 é feito o mapeamento com a anotação `@Table` para a tabela `beacon_access`. As anotações `@Column` também são utilizadas para ligação com os campos em banco de dados, assim como é feito com a tabela de `beacon`, porém nesse caso na linha 31 existe a anotação `@JoinColumn` que simboliza em termos de SQL a clausula `join`. Na Anotação `@JoinColumn` é informado qual o campo na tabela de origem que se deseja realizar o relacionamento e ele identifica a tabela e o campo de relacionamento através da declaração no código-fonte que nesse caso é a entidade `Beacon` que está mapeada como sendo a tabela `beacon`, tendo o campo `id` como chave. A partir dessas informações ele carrega e entidade por reflexão ao utilizar a classe `BeaconAccess`. Para busca e gravação dos dados para a classe `BeaconAccess` foi utilizada a mesma estratégia usada com

a classe `Beacon`, sendo que foi criada uma `interface` que estende a `interface` `CrudRepository`, como é demonstrado na Figura 28 na linha 10. Outro ponto é na linha 12 que foi implementado um método que busca todos os acessos de um determinado *beacon*. O método retorna uma lista de entidade `BeaconAccess` tendo como parâmetro somente o `id` do `beacon` desejado, de modo que toda parte de carregamento da entidade `Beacon` e `BeaconAccess` é feito pelo *framework*.

Figura 27 - Código-fonte modelagem da tabela de acessos dos beacons

```
15 @Entity
16 @Table(name = "beacon_access")
17 public class BeaconAccess {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     @Column(name = "id", updatable = false, nullable = false)
22     private Long id;
23
24     @Column(name="date_access")
25     private Date dateAccess;
26
27     @Column(name="time_access")
28     private Time timeAccess;
29
30     @ManyToOne
31     @JoinColumn(name="id_beacon")
32     private Beacon idBeacon;
```

Fonte: elaborado pelo autor.

Figura 28 - Código-fonte repositório da tabela de acesso dos beacons

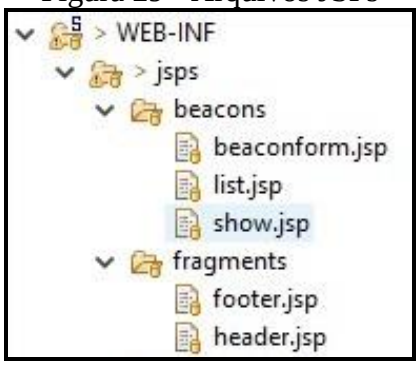
```
10 public interface BeaconAccessRepository extends CrudRepository<BeaconAccess, Long> {
11
12     public List<BeaconAccess> findByIdBeacon(Beacon idBeacon);
13
14 }
```

Fonte: elaborado pelo autor.

Para iteração com o usuário, demonstrando os dados gravados em banco de dados e disponíveis para alteração, foi desenvolvida a camada *view* inteiramente em JSP. A Figura 29 mostra todos os arquivos necessários para exibição, controle e validação dos valores de cadastro dos beacons, sendo que os arquivos `footer.jsp` e `header.jsp` estão presentes em toda a página exibida. O arquivo `footer.jsp` corresponde ao rodapé com as informações das páginas e o arquivo `header.jsp` ao cabeçalho com título e menu da ferramenta. O arquivo `beaconform.jsp` é responsável pela inserção de um novo beacon e pela alteração de beacons já cadastrados, ele cria um formulário com os campos para exibição e alteração. Já o arquivo `list.jsp` fica com a responsabilidade de exibir uma lista dos beacons cadastrados e exibição de botões com as opções disponíveis para os beacons, como alterar, deletar, gerar página

HTML, etc. Por fim, o arquivo `show.jsp` é responsável por uma exibição simples dos beacons cadastrados.

Figura 29 - Arquivos JSPs



Fonte: elaborado pelo autor.

A Figura 30 demonstra parte do código-fonte do formulário de edição do cadastro de beacon. Na linha 54 é indicado o tipo de formulário como sendo horizontal através da indicação da classe `form-horizontal` no atributo `class`. Além disso, também é indicado o tipo de mensagem emitida ao final do processo do formulário que será um `post` para a URL indicada que é informado nos atributos `method` e `action` respectivamente. Também na linha 54 é informado no atributo `modelAttribute` o nome do atributo de modelo, que nesse caso é `beaconForm`, que terá os campos do formulário e existirá nas camadas *view*, *model* e *controller*, criando assim um atributo único e atualizado nas três camadas.

Na Figura 30 também é demonstrado na linha 58 a criação do campo `uid` do beacon como um input criado na linha 62 e seu *label* na linha 60. A mesma coisa ocorre para o campo `urlid` especificado na linha 72 e *label* na linha 70. O campo `estado` do beacon é uma lista de opções pré-estabelecidas sendo elas `ativo` e `inativo`, para isso foi criado uma lista que é utilizada no formulário. Para alteração desse do campo `estado` foi criado um *combobox* na linha 82 e vinculado a lista pré-definida na linha 84, isso faz com que *combobox* tenha apenas os tipos sugeridos pela lista.

No cadastro dos beacons é possível escolher a localização baseado na planta baixa aonde o beacon se encontra instalado. Para a escolha do local foi adotado o esquema de *point-click*, no qual o usuário que está cadastrando o beacon clica na posição exata do beacon na planta baixa interna do local para pegar a posição do mapa. O algoritmo demonstrado na Figura 31 realiza a captura da posição do clique. Através da linha 37 e linha 36 ele captura a posição *x* e posição *y* respectivamente, para atribuir aos componentes em tela para exibição e gravação da posição.

Figura 30 - Código-fonte do formulário do cadastro de beacon

```
<form:form class="form-horizontal" method="post" modelAttribute="beaconForm" action="${beaconActionUrl}">

    <form:hidden path="id" />

    <spring:bind path="uid">
        <div class="form-group ${status.error ? 'has-error' : ''}">
            <label class="col-sm-2 control-label">ID</label>
            <div class="col-sm-10">
                <form:input path="uid" type="text" class="form-control " id="uid" placeholder="uid" />
                <form:errors path="uid" class="control-label" />
            </div>
        </div>
    </spring:bind>

    <spring:bind path="urlid">
        <div class="form-group ${status.error ? 'has-error' : ''}">
            <label class="col-sm-2 control-label">URL</label>
            <div class="col-sm-10">
                <form:input path="urlid" class="form-control" id="urlid" placeholder="urlid" />
                <form:errors path="urlid" class="control-label" />
            </div>
        </div>
    </spring:bind>

    <spring:bind path="estado">
        <div class="form-group ${status.error ? 'has-error' : ''}">
            <label class="col-sm-2 control-label">Estado</label>
            <div class="col-sm-5">
                <form:select path="estado" class="form-control">
                    <form:option value="NONE" label="--- Select ---" />
                    <form:options items="${estadoList}" />
                </form:select>
                <form:errors path="estado" class="control-label" />
            </div>
            <div class="col-sm-5"></div>
        </div>
    </spring:bind>
```

Fonte: elaborado pelo autor.

Figura 31 - Algoritmo de captura da posição do beacon na planta baixa

```
function salvarPosicao(event) {
    document.getElementById('posicaoX').value = event.offsetX;
    document.getElementById('posicaoY').value = event.offsetY;
}
```

Fonte: elaborado pelo autor.

3.3.2.2 Gerador de página HTML

O gerador de página HTML foi desenvolvido para abstrair ao usuário todo conceito de HTML, CSS, etc., sendo que um leigo poderia montar uma página para exibição de componentes. O jeito mais prático de realizar esse tipo de trabalho é usando o conceito *drag-and-drop* que é o movimento de arrastar e soltar componentes visuais.

Para realizar essa tarefa foi utilizado um *framework* chamado *angular-drag-and-drop-lists* que é baseado no Angular JS. A Figura 32 mostra uma lista dos componentes que se inicia na linha 8, que podem ser utilizados na ferramenta. Essa lista é definida pela própria ferramenta, igualmente os atributos alteráveis de cada componente que ficam em uma lista interna de cada um. Esses atributos de componentes são fornecidos pela ferramenta, como por exemplo, na linha 8 até a linha 15 há a estrutura do componente `label` do HTML, junto com ele temos os `fields` ou atributos editáveis que se iniciam na linha 12 até a linha 14, que é

apenas o `texto` exibido no componente. Diferente do componente de `image` do HTML que inicia na linha 17 e vai até a linha 28, tendo com ela outros dois atributos que o componente `label` não contém, sendo eles o `caminho` em que a imagem se encontra e o `link` para o qual a imagem encaminha ao ser selecionada.

Figura 32 - Lista de componentes pré-definidos

```
5   modelsNull = {
6       selected: null,
7       uid: "",
8       templates: [{
9               name: 'Rótulo',
10              type: "label",
11              id: 1,
12              fields: [{
13                      description: "Texto",
14                      value: ""
15              }]
16          },
17          {
18              name: 'Imagem',
19              type: "img",
20              fields: [{
21                      description: "Caminho",
22                      value: ""
23              },
24              {
25                      description: "Link",
26                      value: ""
27              }]
28          },
29          {
30              name: 'Container 1',
31              type: "container",
32              id: 4,
33              fields: [{
34                      description: "Titulo",
35                      value: ""
36              }],
37              columns: [
38                  []
39              ]
40          },
```

Fonte: elaborado pelo autor.

A Figura 33 demonstra o código-fonte para montagem dos componentes HTML. Os componentes ativos ou escolhidos pelo usuário ficam armazenados na lista `list` que é percorrida na linha 64 através da diretiva do `ng-repet` do Angular JS. A cada componente na lista encontrado é feita a chamada da inclusão do item através da diretiva `ng-include` também do Angular JS. Após encontrar o componente, ele busca nos *script*s de inserção de *template* dos componentes na Figura 34 algum que seja do componente atual e o inclui na área de edição. Os *scripts* de *template* dos componentes são pré-estabelecidos pela ferramenta desenvolvida e para cada componente disponibilizado existe um *script* de como ele deve ser inserido na página HTML. Na linha 29 e linha 30 da Figura 34 é feito o uso dos `fields` do

componente de `image` que foi declarado na Figura 32, através dessas diretivas que os valores e propriedades dos componentes são alterados.

Figura 33 - Código-fonte repetição na lista de componentes

```
62      <div class="col s7">
63          <div class="row">
64              <div ng-repeat="(zone, list) in models.dropzones" class="col s12">
65                  <div class="dropzone box box-yellow editor">
66                      <h3>Editor: {{zone}}</h3>
67                      <div ng-include="'list.html'"></div>
68                  </div>
69              </div>
70          </div>
71      </div>
```

Fonte: elaborado pelo autor.

Figura 34 - Código-fonte montagem pré-estabelecida

```
16  <script type="text/ng-template" id="container.html">
17      <div class="container box box-blue">
18          <h3 ng-hide="item.id == ''">{{item.id}}</h3>
19          <div class="column{{item.columns.length}}" ng-repeat="list in item.columns" ng-include="'list.html'"></div>
20          <div class="clearfix"></div>
21      </div>
22  </script>
23
24  <script type="text/ng-template" id="label.html">
25      <div class="item">{{item.fields[0].value}}</div>
26  </script>
27
28  <script type="text/ng-template" id="img.html">
29      <a href="{{item.fields[1].value}}">
30          <img src="{{item.fields[0].value}}">
31      </a>
32  </script>
```

Fonte: elaborado pelo autor.

Para a edição, teste e exibição da página foram criadas rotas através do *router* do Angular JS. Essas rotas redirecionam para *templates* e *controllers* diferentes dependendo do mapeamento realizado, é possível também inserir parâmetros para cada rota definida. Na Figura 35 são demonstradas as rotas da ferramenta, na linha 4 é mapeado que caso a URL não tenha nenhum parâmetro o redirecionamento acontece para a página `nested.html` (linha 5). Fazendo uso do *controller* `NestedListsControler`, essa rota é utilizada para testes de usabilidade da ferramenta de montagem de página. Na linha 8 é mapeada a rota utilizada para edição da página HTML vinculada ao beacon. O parâmetro `:uid` é utilizado para realizar o vínculo da página editada ao beacon com esse uid. Na linha 9 e na linha 10 ocorre a ligação com o *template* e o *controller* respectivamente. Na linha 12 é realizado o mapeamento da última rota que é a rota responsável pela geração do mapa de calor. A rota também contém o parâmetro `:uid` que indica qual página HTML gerada deverá ser carregada para exibição, pois essa rota é apenas para visualização da página e não para edição dela. Na linha 13 é feita a ligação com o *template* de exibição `vis.html` e na linha 14 é feita a ligação com o *controller* `NestedListsControllerVis` responsável pela exibição e carregamento da página.

A linha 16 serve para caso não for nenhuma das rotas mapeadas ele irá encaminhar para a rota da linha 17 que no caso é a primeira rota mapeada.

Figura 35 - Rotas da ferramenta

```
1   angular.module("demo", ["ngRoute", "dndLists"])
2       .config(function ($routeProvider) {
3           $routeProvider
4               .when('/', {
5                   templateUrl: '/nested/nested.html',
6                   controller: 'NestedListsController'
7               })
8               .when('/:uid', {
9                   templateUrl: '/nested/nested.html',
10                  controller: 'NestedListsController'
11              })
12              .when('/:uid/vis/', {
13                  templateUrl: '/nested/vis.html',
14                  controller: 'NestedListsControllerVis'
15              })
16              .otherwise({
17                  redirectTo: '/'
18              });
19      }).run(['$rootScope', function ($rootScope) {
20          $rootScope.models =  {};
21      }]);
```

Fonte: elaborado pelo autor.

3.3.2.3 Serviços

Os serviços criados têm como objetivo fazer a conexão do gerador de páginas HTML com o banco de dados, além da geração de registros de acessos a visualização dessas páginas. A Figura 36 mostra o mapeamento da requisição de busca do *template* utilizado na página para montagem dinâmica dela. Na linha 74 é informado o caminho `GetTemplate` que fornece acesso a esse serviço. Além disso, é informado o tipo de requisição que nesse caso é um `GET`, pois há somente busca de dados. Essa requisição tem como parâmetro somente o UID do beacon. Através do UID é realizada a busca utilizando o `beaconRepository` na linha 76, caso for encontrado algo no banco de dados ele retorna na linha 79 o *template* gravado no banco de dados, caso contrário retorna uma mensagem negativa de um *template* na linha 81

Figura 36 - Mapeamento requisição de template

```
73      @Transactional
74      @RequestMapping(value = "/GetTemplate", method = RequestMethod.GET)
75      public String getTemplate(@RequestParam String uid) {
76          Optional<Beacon> beacon = beaconRepository.findByUid(uid);
77
78          if (beacon.isPresent())
79              return beacon.get().getTemplateModel();
80
81          return "{\"resposta\" : \"none\"}";
82      }
```

Fonte: elaborado pelo autor.

Na Figura 37 é demonstrado o uso e consumo do serviço `GetTemplate`. Na linha 93 é informado qual o tipo de requisição que será feita, que no caso do serviço `GetTemplate` é `GET`. Logo mais na linha 94 é atribuído a URL em qual o serviço se encontra juntamente com o parâmetro de UID que é informado ao final dela. Após obter a resposta do servidor é verificado a consistência da mensagem nas linhas 98, 99 e 100 para averiguar se a resposta do servidor é válida, caso a resposta for válida é atribuída a variável `models`, na linha 102. A resposta do servidor é puramente a estrutura JSON do objeto `models`. Sendo assim, a busca e armazenamento dele em banco de dados é feita em formato JSON, pois assim é realizado somente uma atribuição aos atributos de controle como é o caso do atributo `models`.

Figura 37 - Requisição e atribuição da reposta do template

```
92          var req = {
93                  method: 'GET',
94                  url: 'http://$SERVIDOR/GetTemplate?uid=' + $routeParams.uid,
95              }
96              $http(req).then(function(response){
97                  console.log(response.data);
98                  if ((response.data != 'none') && (response.data != '') &&
99                      (response.data != null) && (response.data != '') &&
100                     (response.data != undefined)) {
101                     $scope.models = response.data;
102                 } else {
103                     $scope.models = modelsNull;
104                     $scope.models.uid = $routeParams.uid;
105                 }
106             }, function(response){
107                 $scope.models = modelsNull;
108                 $scope.models.uid = $routeParams.uid;
109             });
110     } else {
111         $scope.models = modelsNull;
112     }
```

Fonte: elaborado pelo autor.

Outro serviço disponibilizado foi o de registro de acessos das páginas dos beacons, ele é demonstrado na Figura 38. O mapeamento dele é feito na linha 38 e é disponibilizado através da primitiva `AccessBeacon` que é uma requisição `POST`. O serviço `AccessBeacon` tem como parâmetro na linha 39 somente o UID do beacon, sendo que através dele é feita a busca do beacon na linha 40 utilizando o `beaconRepository`. Após atestar na linha 42 que ele existe no banco de dados, é gerado um registro com a data atual e hora atual do acesso (linhas 44 e 45 respectivamente). Assim que é realizado essa montagem do registro, é feita a inserção dele em banco através do `beaconAccessRepository` na linha 48 e dado o retorno da requisição positivamente na linha 50. Caso o beacon não for encontrado, é retornado na linha 53 uma resposta sinalizando que a URL acessada não pertence a nenhum beacon cadastrado.

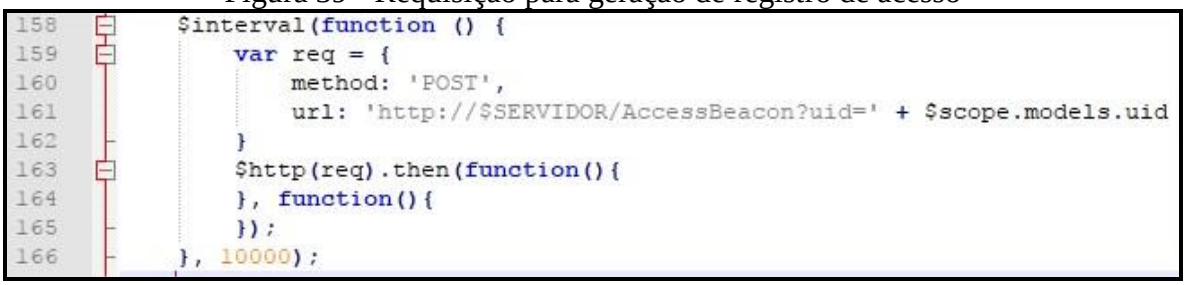
Figura 38 - Mapeamento do registro de acesso do beacon

```
37    @Transactional
38    @RequestMapping(value = "/AccessBeacon", method = RequestMethod.POST)
39    public String newAccessBeacon(@RequestParam String uid) {
40        Optional<Beacon> beacon = beaconRepository.findByUid(uid);
41
42        if (beacon.isPresent()) {
43            BeaconAccess beaconAccess = new BeaconAccess();
44            beaconAccess.setDateAccess(Date.valueOf(LocalDate.now()));
45            beaconAccess.setTimeAccess(Time.valueOf(LocalTime.now()));
46            beaconAccess.setIdBeacon(beacon.get());
47
48            beaconAccessRepository.save(beaconAccess);
49
50            return "{\"resposta\" : \"Ok\"}";
51        }
52
53        return "{\"resposta\" : \"Url sem vínculo a Beacon\"}";
54    }
```

Fonte: elaborado pelo autor.

Na Figura 39 é demonstrado o uso do serviço `AccessBeacon`, sendo que na linha 160 é informado o tipo de requisição do serviço como `POST` e logo abaixo na linha 161 é informado a URL que o serviço está disponível. Essa requisição ocorre a cada 10.000 milissegundos, ou seja, a cada 10 segundos da página em visualização é feita uma requisição ao serviço para gerar um registro de acesso. Esse tempo está informado na linha 166 e faz parte dos parâmetros da diretiva `$interval` do Angular JS.

Figura 39 - Requisição para geração de registro de acesso



```
158    $interval(function () {
159        var req = {
160            method: 'POST',
161            url: 'http://$SERVIDOR/AccessBeacon?uid=' + $scope.models.uid
162        }
163        $http(req).then(function(){
164        }, function(){
165        });
166    }, 10000);
```

Fonte: elaborado pelo autor.

Para fins de atualização e armazenamento da estrutura de montagem da página HTML, foi disponibilizado o serviço `UpdateTemplate`. Esse serviço é responsável por gravar em banco de dados o *template* da página HTML do beacon. Na Figura 40 é demonstrado o registro do serviço, de modo que na linha 57 é informada a primitiva `UpdateTemplate` que é por qual o beacon se faz acessível. O tipo de requisição é definido como `POST`, pois irá consumir informações e também qual é o tipo de informação que ele irá consumir que é definido como JSON através do tipo `MediaType.APPLICATION_JSON_VALUE`. Na linha 58, o parâmetro UID do beacon receberá a estrutura da página e o JSON passado como parâmetro na *String*. Na linha 59 é feita a busca do beacon através da UID informada e verificado sua existência no banco de dados na logo após na linha 61. Assim que for constatado que o

beacon está cadastrado é atribuído na linha 63 o JSON informado ao campo de `templateModel` do beacon e realizado sua gravação no banco na linha 65 fazendo uso do `beaconRepository`. Caso a gravação ocorra corretamente é retornado na linha 67 uma reposta positiva, porém se o UID informado não está vinculado a nenhum beacon cadastrado na ferramenta é retornado na linha 70 uma reposta com a situação ocorrida.

Figura 40 - Mapeamento gravação de template da página do beacon

```
56    @Transactional
57    @RequestMapping(value = "/UpdateTemplate", method = RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE)
58    public String updateTemplate(@RequestParam String uid, @RequestBody String model) {
59        Optional<Beacon> beacon = beaconRepository.findByUid(uid);
60
61        if (beacon.isPresent()) {
62            Beacon beaconUpdate = beacon.get();
63            beaconUpdate.setTemplateModel(model);
64
65            beaconRepository.save(beaconUpdate);
66
67            return "{\"resposta\" : \"Ok\"}";
68        }
69
70        return "{\"resposta\" : \"UID sem vínculo a Beacon\"}";
71    }
```

Fonte: elaborado pelo autor.

Na Figura 41 é demonstrado o uso do serviço `UpdateTemplate`. Na linha 122 é informado o tipo de requisição como sendo `POST` e logo abaixo na linha 123 é informada a URL em que o serviço está disponibilizado. Para realizar o envio do JSON da estrutura da página, é atribuído a variável `modelString` na linha 120 à estrutura `models` do escopo da página que já é uma estrutura JSON por si só. Para envio desse JSON foi informado na linha 126 no `header` da requisição o tipo de conteúdo como sendo um `application/json` e atribuído o JSON ao valor `data` da requisição.

Figura 41 - Envio de post para gravação de template da página do beacon

```
119         $rootScope.models = angular.copy($scope.models);
120         var modelString = JSON.stringify($scope.models);
121         var req = {
122             method: 'POST',
123             url: 'http://$SERVIDOR/UpdateTemplate?uid=' +
124                  $rootScope.models.uid,
125             headers: {
126                 'Content-Type': "application/json"
127             },
128             data: modelString
129         }
130         $http(req).then(function(){
131         }, function(){
132         });
```

Fonte: elaborado pelo autor.

3.3.2.4 Autenticação e mapa de calor

Para garantir a segurança à informação dos dados dos beacons foi implementado na última etapa a autenticação da ferramenta. A autenticação ocorre somente na parte da

ferramenta que há o cadastro e configuração das páginas HTML dos beacons, sendo que somente para a visualização da página HTML montada não é necessário autenticação.

Para controlar os usuários foi criado uma modelagem específica para esse fim. Ela segue o mesmo modelo das já criadas na etapa de cadastros. A Figura 42 mostra o mapeamento da tabela na linha 11 em que é feita a ligação da entidade com a tabela `users`. Além disso, é realizado o mapeamento dos campos `id`, `username`, e `password` do usuário nas linhas 16, 19 e 22 respectivamente. Na Figura 43 na linha 9 é feito a extensão da `interface CrudRepository` para ter acesso a toda a abstração de conexão e manipulação com o banco de dados. Ainda, na linha 11 foi criado o método `findByUsername` que retorna a entidade de usuário que contém aquele `username` informado.

Figura 42 - Classe de modelagem da tabela de usuários

```
10 @Entity
11 @Table(name = "users")
12 public class Usuario {
13
14     @Id
15     @GeneratedValue(strategy = GenerationType.AUTO)
16     @Column(name = "id", updatable = false, nullable = false)
17     private Long id;
18
19     @Column(name = "username")
20     private String username;
21
22     @Column(name = "password")
23     private String password;
```

Fonte: elaborado pelo autor.

Figura 43 - Interface de repositório da tabela de usuários

```
 9 public interface UsuarioRepository extends CrudRepository<Usuario, Long> {
10
11     Optional<Usuario> findByUsername(String username);
12
13 }
```

Fonte: elaborado pelo autor.

Afim de controlar todo acesso a ferramenta na área de configurações e cadastros foi criado um interceptador das requisições. A Figura 44 demonstra o uso do interceptador, nas linhas 16, 17 e 18 é verificado se o caminho que está sendo acessado é os caminhos de autenticação, caso não for ele verifica a seção do usuário na linha 22 através dos *cookies* e caso não houver nenhuma seção ativa na linha 26 é feito o encaminhamento da requisição para a tela de autenticação da ferramenta.

Figura 44 - Interceptador de requisição para autenticação

```
8  public class AutorizadorInterceptor extends HandlerInterceptorAdapter {
9
10     @Override
11     public boolean preHandle(HttpServletRequest request,
12             HttpServletResponse response, Object controller)
13             throws Exception {
14
15         String uri = request.getRequestURI();
16         if (uri.endsWith("loginForm") ||
17             uri.endsWith("efetuaLogin") ||
18             uri.contains("resources")) {
19             return true;
20         }
21
22         if (request.getSession().getAttribute("usuarioLogado") != null) {
23             return true;
24         }
25
26         response.sendRedirect("loginForm");
27         return false;
28     }
29 }
```

Fonte: elaborado pelo autor.

A autenticação ocorre no *controller* responsável pela autenticação. Foi implementado o *controller* `LoginController` com a responsabilidade de realizar a autenticação na ferramenta. Na Figura 45 é demonstrado o código-fonte do *controller* e os serviços de autenticação mapeados, de modo que na linha 22 é mapeado a página inicial da ferramenta para realizar o encaminhamento e na linha 26 tem-se o retorno do `login`. Na linha 29 é realizado as consistências e validações da autenticação no qual é verificado se o usuário realmente existe (linha 32) e se a senha é compatível (linha 33). Caso houver sucesso na autenticação é encaminhado na linha 35 para o `menu` da ferramenta, caso contrário é retornado a tela de autenticação para mais tentativas.

Figura 45 - *Controller* da área de autenticação

```
@Controller
public class LoginController {

    @Autowired
    UsuarioRepository usuarioRepository;

    @RequestMapping("/")
    public String loginForm(Model model) {
        Usuario usuario = new Usuario();
        model.addAttribute("usuarioForm", usuario);
        return "login";
    }

    @RequestMapping(value = "/efetuaLogin", method = RequestMethod.POST)
    public String efetuaLogin(Model model, Usuario usuario, HttpSession session) {
        Optional<Usuario> usuarioOptional = usuarioRepository.findByUsername(usuario.getUsername());
        if ((usuarioOptional.isPresent()) &&
            (usuarioOptional.get().getPassword().equals(usuario.getPassword()))) {
            session.setAttribute("usuarioLogado", usuario);
            return "menu";
        }
        return "redirect:loginForm";
    }
}
```

Fonte: elaborado pelo autor.

Na Figura 46 é exibido o serviço responsável por fazer o compilado dos dados de acesso dos beacons e interpretá-los para uma estrutura JSON. Na linha 92 é utilizado o `beaconRepository` para realizar a busca dos beacons cadastrados. Após isso é realizado na linha 93 a busca dos acessos de cada beacon encontrado. Com os acessos do beacon é verificado o valor máximo de acessos e caso ele for o maior, é feito o armazenamento temporário dele. Cada beacon é atribuído a uma sub-estrutura do JSON que contém o `posX`, `posY` e `value` que são a posição *x* onde o beacon se encontra na planta baixa, a posição *y* onde o beacon se encontra na planta baixa e o número de acessos a página vinculada aquele beacon. Após ter contabilizado cada beacon do mapa, é retornado toda a estrutura montada juntamente com o valor máximo de acessos que houve em um único beacon.

Figura 46 - Serviço para contagem de acessos

```
    @Transactional
    @RequestMapping(value = "/GetAllAccessBeacon", method = RequestMethod.GET)
    public String getAllAccessBeacon() {
        String baseFormat =
                "{ \"posX\" : \"%d\", "
                + "\"posY\" : \"%d\", "
                + "\"value\" : \"%d\" },";

        beaconRepository.findAll().forEach(beacon -> {
            List<BeaconAccess> allAccess = beaconAccessRepository.findByIdBeacon(beacon);

            int access = allAccess.size();
            if (maxValue < access)
                maxValue = access;

            retorno += String.format(baseFormat, beacon.getPosicaoX(), beacon.getPosicaoY(), access);
        });

        String retornoLocal = "{\"max\" : \"" + maxValue + "\", \"access\" : [ "
                              + retorno.substring(0, retorno.length() - 1) + "]}";
        retorno = "";
        maxValue = 0;

        return retornoLocal;
    }
```

Fonte: elaborado pelo autor.

O mapa de calor foi gerado com a utilização de um *framework* chamado *simple-heat*. Esse *framework* é baseado em JavaScript e utiliza o *canvas* para desenhar o mapa de calor na página HTML. A Figura 47 demonstra o *script* utilizado para a alimentação de informação do mapa de calor. Na linha 174 é realizada a requisição para o serviço `GetAllAccessBeacon` para obtenção dos valores de acesso dos beacons. Nas linhas 178, 179 e 180 é feita a atribuição individual de cada ponto do mapa de calor inserindo os valores da posição *x*, posição y e quantidade de acesso respectivamente de cada um. Após os dados de acessos serem atribuídos, é realizado na linha 183 a atribuição da massa de dados dos acessos e do valor máximo que significará a área mais quente do mapa. Nesse caso foi utilizado o maior valor de acesso de um beacon multiplicado por 1,5, pois com os testes efetuados a visibilidade dos pontos ficou mais clara com essa configuração. Assim que todos os dados foram inseridos, a linha 191 inicia o algoritmo do *framework* para desenhar o mapa de calor.

Figura 47 - Script para geração do mapa de calor

```
161    <script>
162        function httpGet(theUrl) {
163            var xmlHttp = new XMLHttpRequest();
164            xmlHttp.open( "GET", theUrl, false ); // false for synchronous request
165            xmlHttp.send( null );
166            return xmlHttp.responseText;
167        }
168
169        window.requestAnimationFrame = window.requestAnimationFrame ||
170                                       window.mozRequestAnimationFrame ||
171                                       window.webkitRequestAnimationFrame ||
172                                       window.msRequestAnimationFrame;
173
174        var response = JSON.parse(httpGet('http://$SERVIDOR/GetAllAccessBeacon'));
175
176        var data = [];
177        for (i = 0; i < response.access.length; i++) {
178            data.push([response.access[i].posX,
179                       response.access[i].posY,
180                       response.access[i].value]);
181        }
182
183        var heat = simpleheat('canvas').data(data).max(response.max * 1.5),
184            frame;
185
186        function draw() {
187            heat.draw();
188            frame = null;
189        }
190
191        draw();
192    </script>
```

Fonte: elaborado pelo autor.

3.3.3 Operacionalidade da implementação

Para demonstrar a operacionalidade da ferramenta será exemplificado os passos para o cadastro e uso de um novo beacon. Após isso, será detalhada a Figura 56 que exibe o diagrama de atividades de um usuário comum da ferramenta.

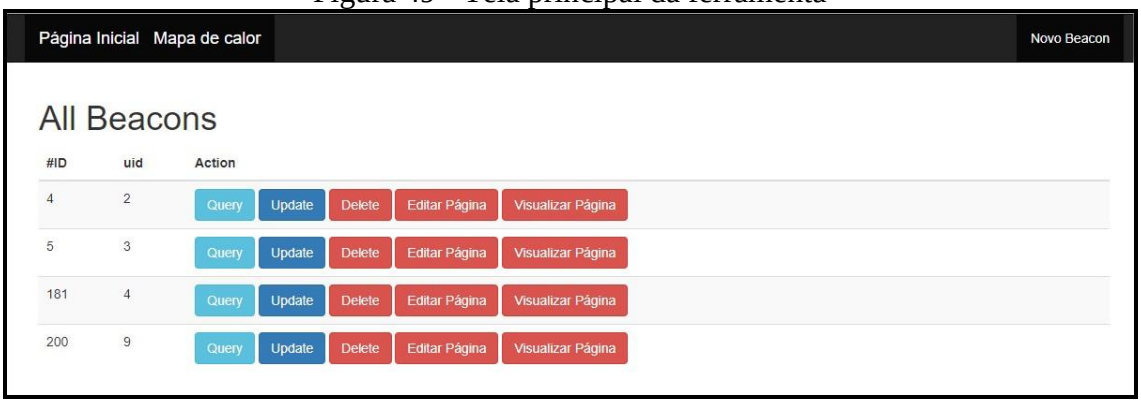
Para iniciar o cadastro dentro da ferramenta é necessário que o usuário seja autenticado como administrador. A Figura 48 demonstra a tela de autenticação da ferramenta pelo qual o usuário terá de se autenticar.

Figura 48 - Tela de autenticação

Fonte: elaborado pelo autor.

Após realizar o passo da autenticação a ferramenta irá direcionar para a tela principal demonstrada na Figura 49. Nela contém os beacons já cadastrados (parte central), acesso ao mapa de calor gerado pelos beacons (no menu superior), a tela de cadastro de novo beacon (menu superior direito), além das opções de atualizar e deletar o beacon e as opções de vinculadas ao gerador de página HTML.

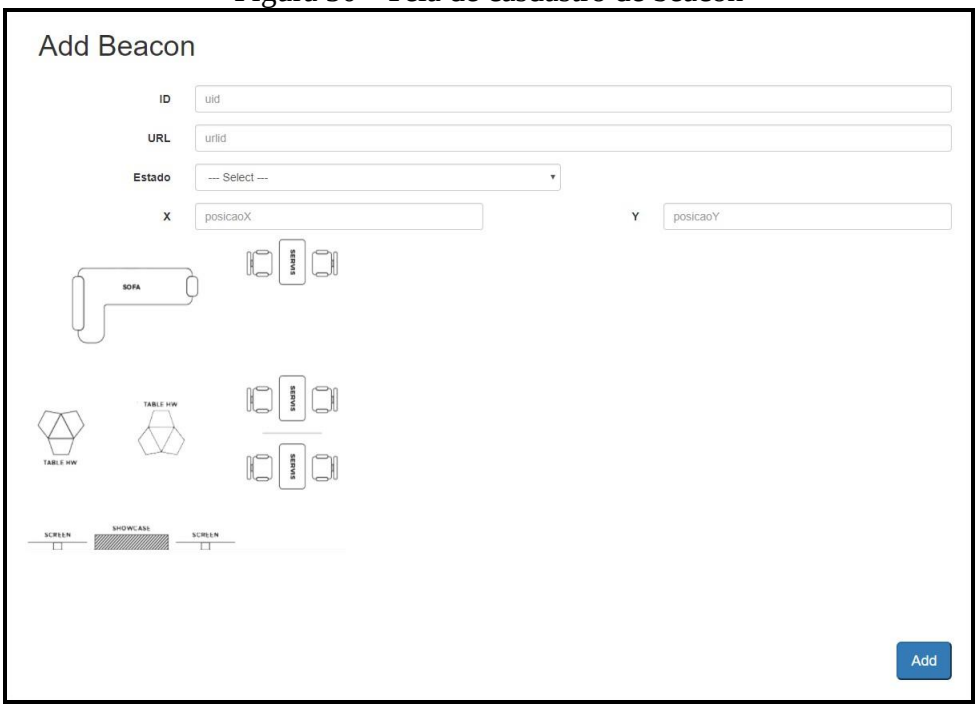
Figura 49 - Tela principal da ferramenta



Fonte: elaborado pelo autor.

Ao clicar no botão `Novo Beacon` desta tela (localizado no canto superior direito), é aberta a tela da Figura 50 que é responsável tanto pelo cadastro de novos beacons quanto a edição de valores dos beacons já cadastrados. A tela contém todos os campos condizentes ao beacon como seu ID e URL, assim como os condizentes a sua localização no espaço interno. Após preencher os campos e escolher na planta baixa o local do beacon, deve-se clicar no botão `Add` para que o beacon seja adicionado e esteja pronto para ser utilizado.

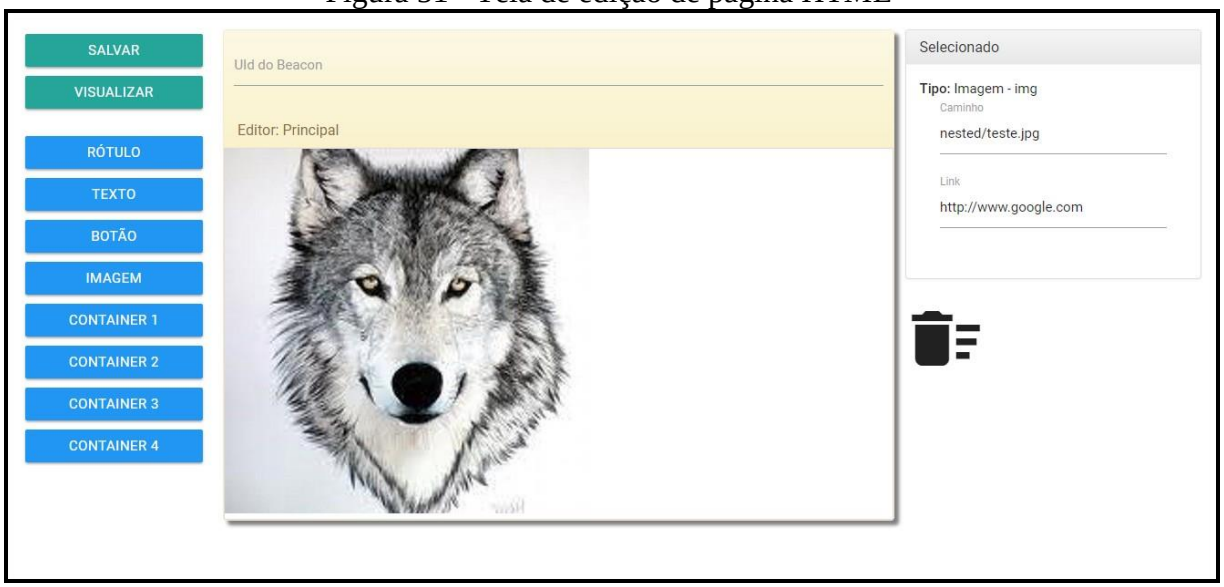
Figura 50 - Tela de casdastro de beacon



Fonte: elaborado pelo autor.

Ao voltar a página inicial, pode-se encontrar o beacon recém incluído e escolher a opção `Editar página` para realizar a montagem da página HTML do beacon. A tela demonstrada na Figura 51 é uma página recém montada apenas com uma imagem de exemplo.

Figura 51 - Tela de edição de página HTML

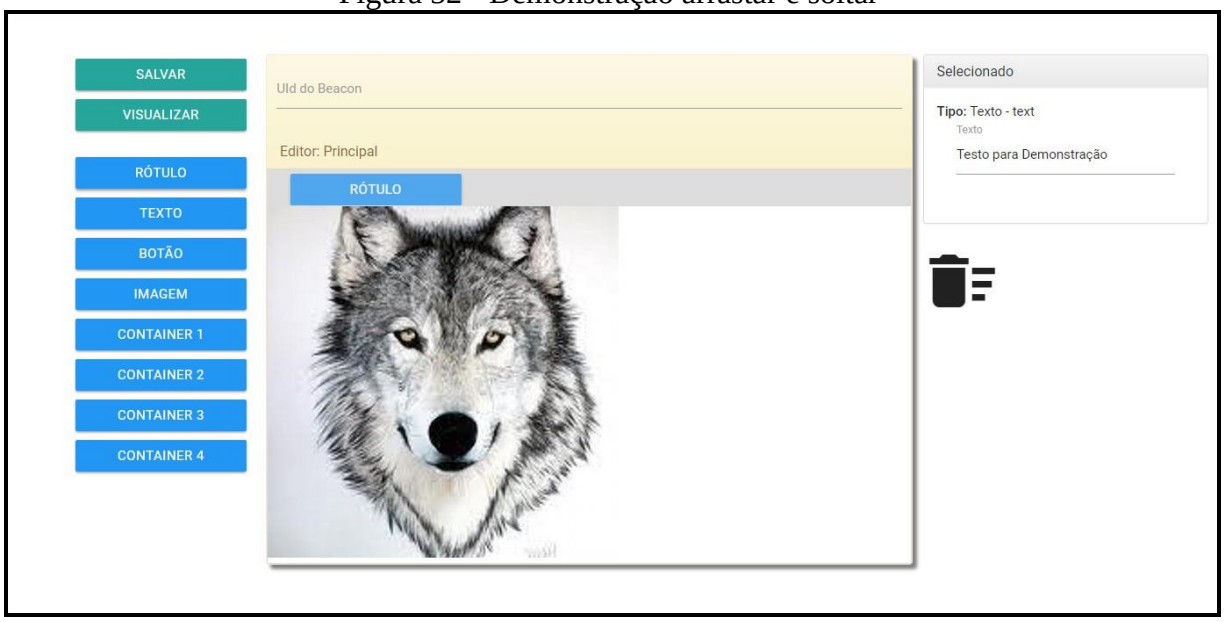


Fonte: elaborado pelo autor.

A Figura 52 demonstra o uso do arrastar e soltar comentado na implementação da ferramenta. No contexto da figura, o clique foi feito no botão `Rótulo` e foi arrastado até onde é demonstrado na Figura 53. Quando o componente é solto, ele é inserido no contexto da

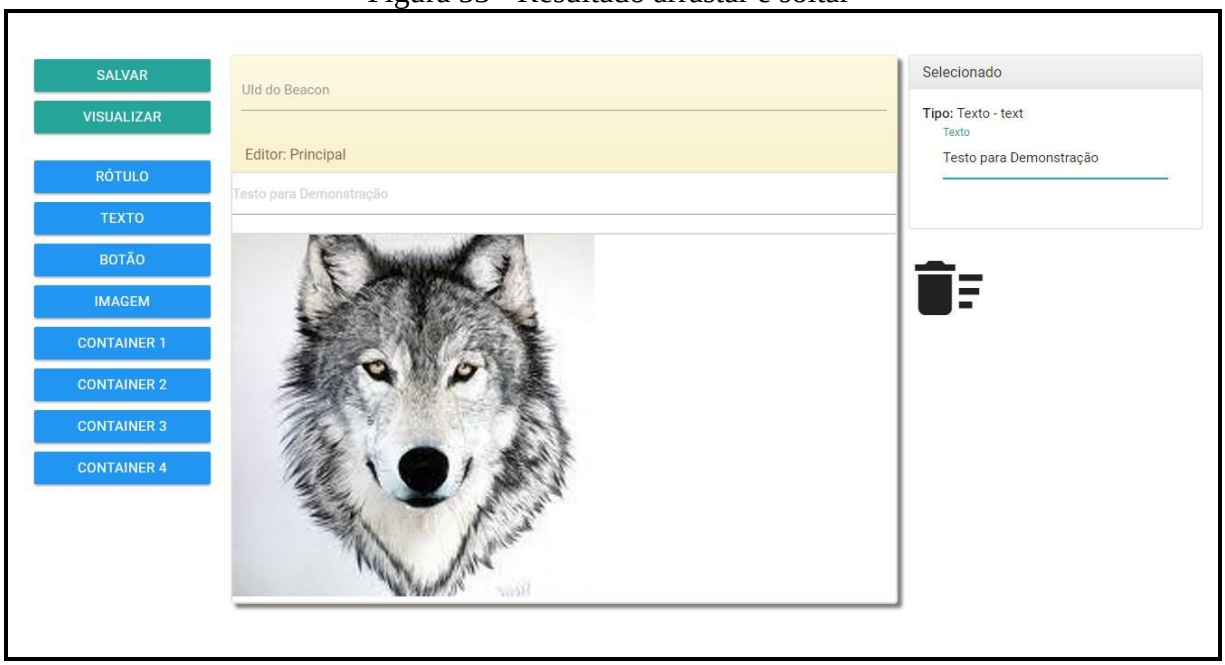
página. Também é possível arrastar o componente da área de edição da página até a lixeira para retirar ele do contexto da página.

Figura 52 - Demonstração arrastar e soltar



Fonte: elaborado pelo autor.

Figura 53 - Resultado arrastar e soltar



Fonte: elaborado pelo autor.

Com essas etapas prontas a ferramenta já estará preparada para a recepção e geração dos registros de acesso do beacon recém cadastrado. Os beacons utilizados para a ferramenta precisam ser do modelo Eddystone, pois a URL que ele emite é a mesma que está cadastrada na ferramenta.

Na Figura 54 e na Figura 55 é apresentado o beacon adquirido através da Hering em parceria com a FURB para a realização dos testes da ferramenta. Esse modelo segue o padrão Eddystone e a URL emitida é configurada pelo próprio aplicativo da fabricante.

Figura 54 - Beacon eddystone trás

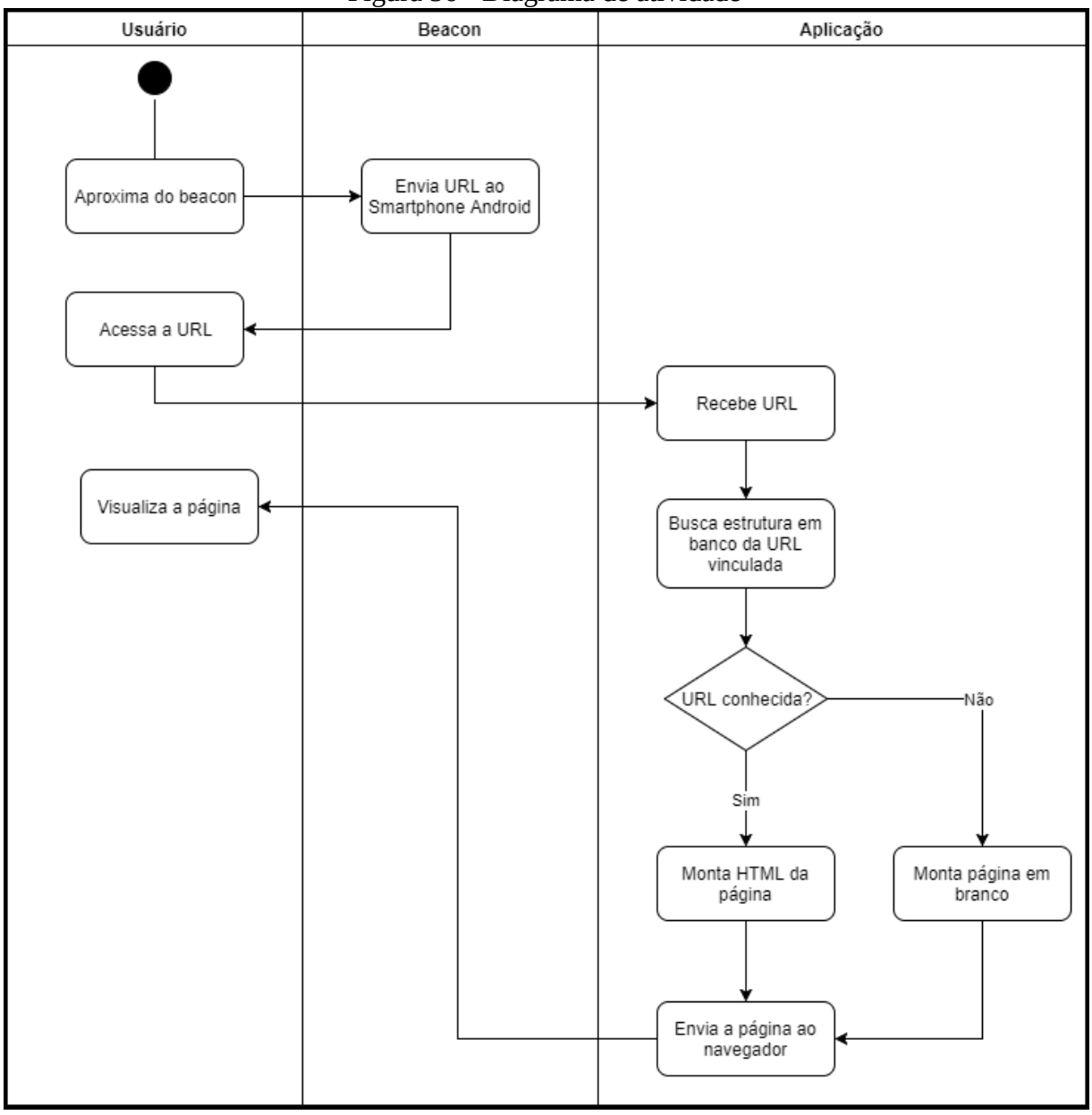
Fonte: elaborado pelo autor.

Figura 55 - Beacon eddystone frente

Fonte: elaborado pelo autor.

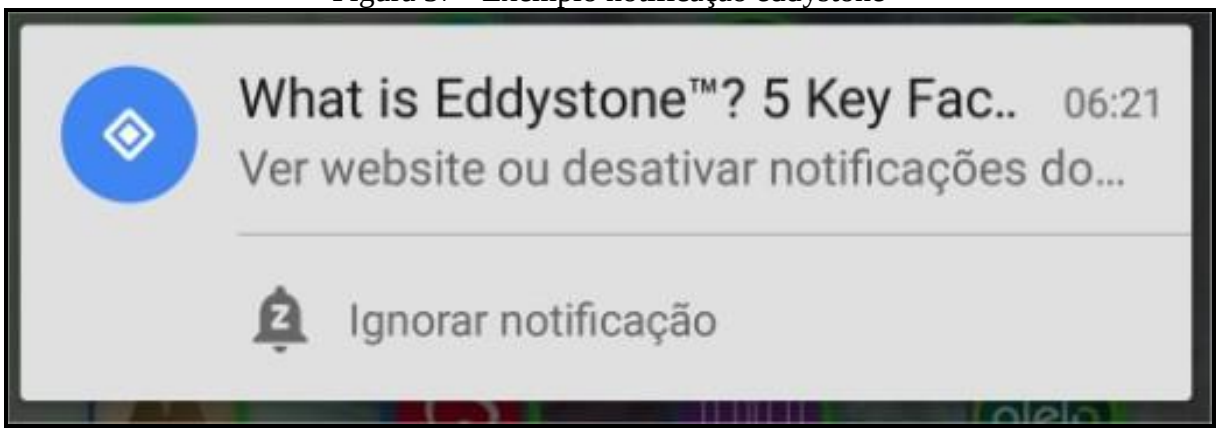
A Figura 56 exibe o diagrama de atividade que descreve o processo que ocorre para acessar a página HTML do aplicativo. Primeiramente o usuário necessita de um dispositivo Android, sendo que o mesmo precisa estar com o bluetooth ligado e estar a uma distância na qual o sinal do bluetooth emitido pelo beacon funcione. Essa distância depende de fabricante para fabricante. Após estar no alcance do sinal irá aparecer uma notificação silenciosa semelhante à Figura 57 no dispositivo Android, ou seja, a notificação é apenas visual sem emissão de som para o usuário. Assim que aparecer é possível acessar a URL ali informada apenas clicando na notificação.

Figura 56 - Diagrama de atividade



Fonte: elaborado pelo autor.

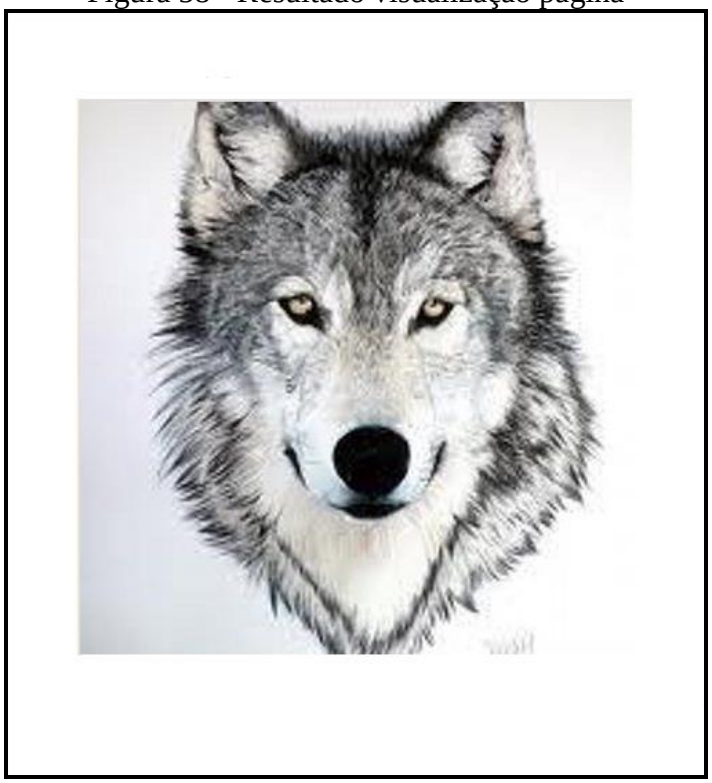
Figura 57 - Exemplo notificação eddystone



Fonte: elaborado pelo autor.

Ao acessar a URL vinculada ao beacon será aberta a página da Figura 58 que foi configurada anteriormente. Após acessar a página, os registros de acesso serão criados e começarão a aparecer no mapa de calor do local.

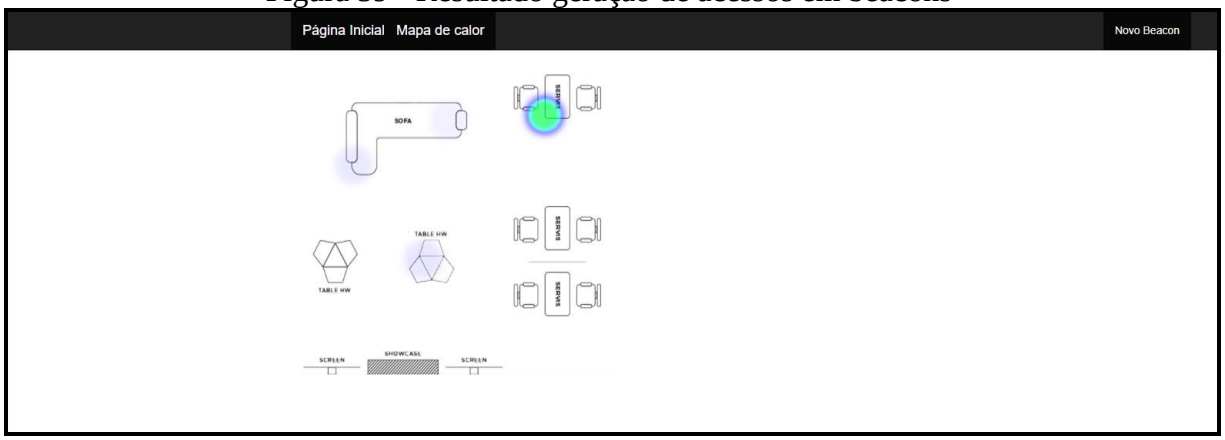
Figura 58 - Resultado visualização página



Fonte: elaborado pelo autor.

Para visualizar o resultado do mapa de calor dos acessos realizados nas páginas dos beacons, basta ir na tela inicial da ferramenta como demonstrado na Figura 49 e clicar no botão com título `Mapa de calor` na parte superior da ferramenta. Ao realizar a ação a ferramenta irá abrir a tela do mapa de calor e gerar ele a partir dos dados de localização do beacon com os acessos realizados a ele. Quanto mais verde a cor do local mais acessos ele tem registrado conforme exemplifica a Figura 59.

Figura 59 - Resultado geração de acessos em beacons



Fonte: elaborado pelo autor.

3.4 RESULTADOS E DISCUSSÕES

Esta seção é dedicada a mostrar os experimentos realizados com a ferramenta. Na seção 3.4.1 é apresentada a comparação de trabalhos correlatos com o trabalho desenvolvido. A seção 3.4.2 apresenta os experimentos e resultados da aplicação.

3.4.1 COMPARAÇÃO DE TRABALHOS CORRELATOS

Nesta seção é realizada uma comparação entre os trabalhos correlatos apresentados na seção 2.5 e a ferramenta final desenvolvida. No Quadro 6 são apresentadas e comparadas algumas características dos trabalhos correlatos e do trabalho desenvolvido. É percebido que este trabalho contempla todas as características destacadas. A característica “Utiliza beacons como marcadores” e “Geração de informação estatísticas” são essenciais para a comparação. Dentre esses trabalhos somente o Launch Here (2013) não cumpre com ambas as características. A característica “Realiza ação após identificar beacon” não é cumprida pelo trabalho desenvolvido, pois quem faz essa identificação é o próprio Android nativamente, já que a ferramenta não está instalada no dispositivo. Dentre todos os trabalhos, apenas o trabalho desenvolvido não necessita de aplicativo instalado para operar.

Quadro 6 - Comparativo entre trabalhos relacionados

Caracterísitcas / Trabalhos correlatos	**PayPal Beacon (2009)**	**Microlocation (2014)**	**Launch Here (2013)**	**Trabalho desenvolvido**
Utiliza beacons como marcadores	X	X	X	X
Realiza ação após identificar beacon		X	X	
Permite customizar ação realizada			X	X
Geração de informações estatísticas	X	X		X
Armazena acessos e a localização do beacon		X		X
Não necessita de aplicativo instalado				X

Fonte: elaborado pelo autor.

3.4.2 Experimento realizado com a ferramenta

Os testes da ferramenta foram realizados durante o mês de novembro de 2017. Nos testes a ferramenta foi hospedada na nuvem na plataforma Heroku. Para execução da ferramenta foi utilizado o navegador web Google Chrome e um dispositivo Asus Zenfone2 com Android 6.0.1. Além disso, foram utilizados os beacons adquiridos pela Hering para realização dos testes.

Os testes foram focados em validar as funcionalidades da ferramenta e adquirir dados para avaliar a performance da ferramenta nos quesitos: tempo de resposta da ferramenta para o dispositivo e tempo de execução da geração do HTML dinamicamente. Conforme demonstra o Quadro 7, o tempo de montagem da página HTML se manteve constante independentemente do modo de uso de rede de celular, porém o mesmo não pode se dizer do tempo de resposta da requisição do *template* da página, quando a execução ocorreu estando com o dispositivo em rede móvel de celular, a resposta foi levemente mais elevada devido a latência existente na rede.

Quadro 7 - Resultado tempos de resposta

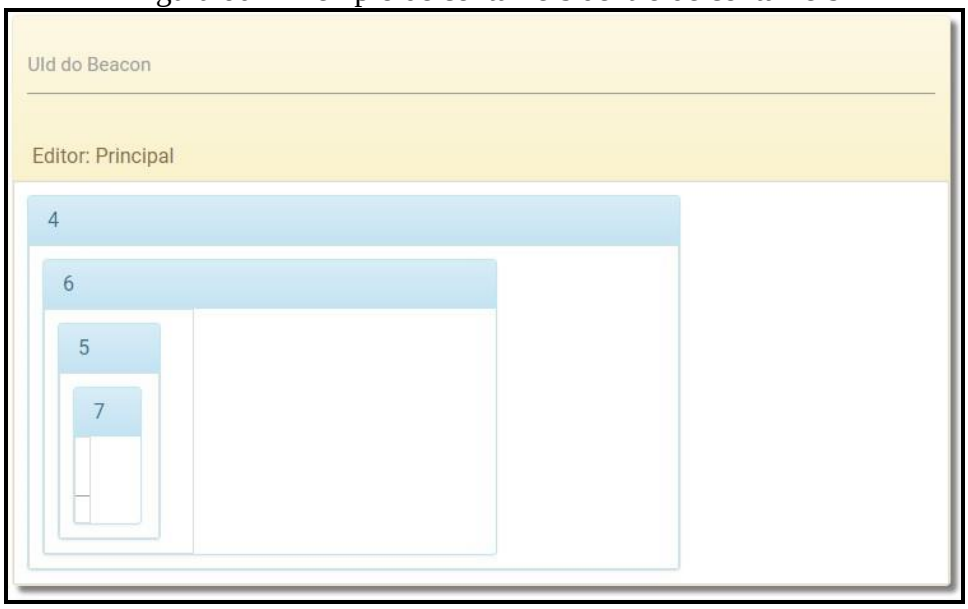
Tentativa	**Tipo de Rede**	**Resposta requisição template (s)**	**Tempo Montagem HTML (s)**
1	Móvel	0,824	1,459
2	Móvel	1,002	1,379
3	Móvel	0,832	1,414
4	Wifi	0.511	1,338
5	Wifi	0.436	1,479
6	Wifi	0.424	1,434

Fonte: elaborado pelo autor.

Foram realizados testes na montagem de páginas HTML com os componentes existentes atualmente na ferramenta. Foi verificado que a funcionalidade de arrastar e soltar

atende as expectativas, porém quando se trata de mover os componentes por dentro dos *containers*, que por sua vez contém *containers* conforme exemplifica a Figura 60, nesse caso a estabilidade do ponteiro é levemente afetada.

Figura 60 - Exemplo de containers dentro de containers



Fonte: elaborado pelo autor.

Por parte da geração do mapa de calor os testes demonstraram que o *framework* utilizado juntamente com a interpretação correta dos acessos e mapeamento da localização dos beacons tiverem seus resultados alcançados. A precisão do mapa de calor estava sempre condizente com os uso e acessos realizados.

Referente ao gerenciamento dos dados dos beacons e segurança na manipulação das informações, a ferramenta demonstrou estabilidade e confiabilidade nas informações salvas em banco de dados, além também da autenticação criar uma camada de controle de quem realmente tem acesso a alterar os dados dos beacons cadastrados. Houve testes realizados na autenticação do usuário referente a sessão ativa. Nesse ponto houve casos em que a sessão realmente não foi salva, por exemplo, quando se utiliza aba anônima pelo Google Chrome. Nesse caso sempre que foi iniciado algum processo pela ferramenta ele pede a autenticação, pois realmente a sessão não é salva.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma ferramenta capaz de utilizar comunicação com beacons e trabalhar de modo dinâmico com eles sem necessitar instalar aplicativo no dispositivo do usuário final. Desta forma facilita a adesão do usuário a utilizar a ferramenta e de aderir a novas experiências tecnológicas.

Inicialmente tentou-se desenvolver uma aplicação utilizando IONIC *framework*, porém ao alinhar expectativas com a representante da Hering descobriu-se que a necessidade era outra. Dessa forma, a implementação foi repensada e reorganizada de tal maneira que tivesse tempo hábil para finalização da ferramenta atual.

O objetivo de gerar página HTML dinamicamente foi atendido de forma que a ferramenta atual de montagem permite muita expansão e customização. Além de ser de fácil utilização até para pessoas que não conhecem muito bem componentes *web*. Os *frameworks* utilizados como apoio e estendidos auxiliaram no andamento da implementação da ferramenta. Outros objetivos alcançados com sucesso foram os de mapeamento dos usos dos beacons através dos registros de acessos de cada página gerada, além da estrutura dinâmica para interpretação e exibição dessas páginas. Através desses objetivos atendidos foi atendido também o objetivo de visualização dos becons mais visitados através da interpretação dos registros gerados em um mapa de calor.

A ferramenta desenvolvida está um pouco crua e tem muito o que evoluir, sendo que todo o algoritmo e inteligência do gerador de página HTML dinâmica pode ser estendido a quaisquer componentes web e a configuração por propriedade deles também está bem extensível. Logo, a ferramenta teve êxito na conclusão e está apta a ser estendida.

Por fim, este trabalho deixa uma contribuição tecnológica, que é o uso e iteração de beacons eddystone com dispositivos Android sem necessidade de aplicação instalada. Quanto à contribuição científica, este trabalho deixa as soluções encontradas para montagem de página HTML sem conhecimento da linguagem, além da fundamentação teórica sobre os temas relacionados.

4.1 EXTENSÕES

Algumas sugestões para possíveis extensões ou melhorias à aplicação desenvolvida são:

a) permitir a possibilidade de cadastrar mais de um local;
b) permitir vincular os beacons aos locais novos;
c) permitir visualiza vários mapas de calor de vários locais;

d) adicionar novos componentes web:
    a. botão com evento de clique;
    b. campo input para armazenamento de valor;
e) integrar software que gerencia URLID do beacon com a ferramenta;
f) tornar a montagem de página HTML parte da ferramenta principal;
g) Correção de erros.

REFERÊNCIAS

ALCARÁ, Adriana Rosecler; CHIARA, Ivone Guerreiro di; TOMAÉL, Maria Inês. **Tipos de compartilhamento de informação e do conhecimento no ambiente P&D**, João Pessoa, v. 20, n. 2, 2010. 15 p. Disponível em: <http://www.ies.ufpb.br/ojs2/index.php/ies/article/view/3876/4361>. Acesso em: 28 mar. 2017.

ALECRIM, Emerson. **Tecnologia Bluetooth**: o que é e como funciona?. São Paulo, 2017. Disponível em: <https://www.infowester.com/bluetooth.php>. Acesso em: 25 maio 2017.

ALLET, André. **Introdução prática ao Spring Framework com uso de Anotações**. [S.I], 2016. Disponível em: <https://www.devmedia.com.br/introducao-pratica-ao-spring-framework-com-uso-de-anotacoes/27859>. Acesso em: 18 nov. 2017.

AMARIZ, Luiz Carlos. *Bluetooth*. [S.I], 2016. Disponível em: <https://www.infoescola.com/tecnologia/bluetooth/>. Acesso em: 23 nov. 2017.

AWWAPPS. **Introducing Launch Here**. [S.I], 2013. Disponível em: <http://blog.awwapps.com/page4/>. Acesso em: 28 mar. 2017.

BABU, Pavithra. **Eddystone vs Physical Web vs iBeacon**: Why Eddystone will Rule the Beacon Space. [S.I], 2016. Disponível em: <https://blog.beaconstac.com/2016/08/eddystone-vs-physical-web-vs-ibeacon-why-eddystone-will-rule-the-beacon-space/>. Acesso em: 23 nov. 2017.

BARBOSA, Tadeu. **AngularJS**: Uma introdução. [S.I], 2016. Disponível em: <http://blog.dtidigital.com.br/angular-js-uma-introducao/>. Acesso em: 18 nov. 2017.

BERNARDO, Kaluan. **Microlocation usa tecnologia da Apple para criar marketing interativo**. [S.I], 2014. Disponível em: <https://startupi.com.br/2014/02/microlocation-usa-tecnologia-da-apple-para-criar-marketing-interativo/>. Acesso em: 28 mar. 2017.

BORGES, Francisco. **Criar aplicativos compatíveis com Beacons**. [S.I], 2016. Disponível em: <https://pt.yeeply.com/blog/criar-aplicativos-compativeis-beacons/>. Acesso em: 27 maio 2017.

CARNEIRO, Conrado. **Beacon**: o que é e quais suas utilizações mais inusitadas. Ouro Preto, 2016. Disponível em: <http://usemobile.com.br/conheca-beacon/>. Acesso em: 28 mar. 2017.

CARVALHO, Marlon. **Spring Framework**: Introdução. [S.I], 2006. Disponível em: <https://imasters.com.br/artigo/4497/java/spring-framework-introducao/?trace=1519021197&source=single>. Acesso em: 18 nov. 2017.

DORES, Kelly. **Varejo investe em tecnologia para medir fluxo e taxa de conversão**. 2016. Disponível em: <http://propmark.com.br/mercado/varejo-investe-em-tecnologia-para-medir-fluxo-e-taxa-de-conversao>. Acesso em: 18 nov. 2017.

ENDEAVOR. **Beacon**: o GPS que ajuda sua marca a localizar as melhores oportunidades. [S.I], 2015. Disponível em: <https://endeavor.org.br/beacon/>. Acesso em: 23 nov. 2017.

ESRI. **Mapas de Calor (Heat Maps)**. [S.I], 2015. Disponível em: <https://doc.arcgis.com/pt-pt/maps-for-sharepoint/arcgis-map-web-part/heat-maps.htm>. Acesso em: 24 nov. 2017.

FARNELL. **The Bluetooth Evolution**. [S.I], 2017. Disponível em: <http://uk.farnell.com/the-bluetooth-evolution>. Acesso em: 24 nov. 2017.

FEITOSA, Gabriel. **AngularJS**: Introdução. [S.I], 2015. Disponível em: <http://gabrielfeitosa.com/iniciando-com-angularjs/>. Acesso em: 18 nov. 2017.

GASIOREK, Agnieszka. **What is Eddystone™**: 5 Key Facts About the New Open Beacon Format From Google. 2016. Disponível em: <https://kontakt.io/blog/what-is-eddystone/>. Acesso em: 23 nov. 2017.

GENTIL, Efraim. **Introdução ao Spring Framework**. [S.I], 2016. Disponível em: <https://www.devmedia.com.br/introducao-ao-spring-framework/26212>. Acesso em: 18 nov. 2017.

GOOGLE. **Camada de mapa de calor**. [S.I], 2017. Disponível em: <https://developers.google.com/maps/documentation/javascript/heatmaplayer?hl=pt-br>. Acesso em: 24 nov. 2017.

GRISWOLD, Alison. **These Heat Maps Show How Retailers Track You As You Shop**. 2014. Disponível em: <http://www.businessinsider.com/how-retailers-track-shoppers-in-heat-maps-2014-1>. Acesso em: 18 nov. 2017.

HANDCOM. **Handcom**. [S.I], 2014. Disponível em: <http://www.handcom.com.br/>. Acesso em: 02 abr. 2017.

KONTAKT.IO. **What is a beacon?**. [S.I], 2017. Disponível em: <https://kontakt.io/beacon-basics/what-is-a-beacon/>. Acesso em: 27 maio 2017.

LAGE, Amarilis. **5 coisas que você precisa saber sobre os beacons**. Rio de Janeiro, 2016. Disponível em: <http://revistagalileu.globo.com/Caminhos-para-o-futuro/Desenvolvimento/noticia/2016/09/5-coisas-que-voce-precisa-saber-sobre-beacons.html>. Acesso em: 27 maio 2017.

LAUNCH HERE. **Launch Here**. [S.I], 2015. Disponível em: <http://launchhere.awwapps.com/>. Acesso em: 28 mar. 2017.

LIFELINK. **O que é o rastreamento pessoal**: Veja quais são as diferentes utilidades desse sistema. [S.I], 2016. Disponível em: <http://www.lifelink.com.br/rastreamento-pessoal/>. Acesso em: 18 nov. 2017.

MEDEIROS, Anderson. **Introdução aos Mapas de Kernel**. 2012. Disponível em: <http://www.andersonmedeiros.com/mapas-de-kernel-parte-1/>. Acesso em: 18 nov. 2017.

MICROLOCATION. **Marketing de proximidade**: quando a tecnologia encontra o negócio. Juiz de Fora, 2013. Disponível em: <http://www.microlocation.com.br/blog/Inovação/18/Marketing de proximidade: quando a tecnologia encontra o negócio>. Acesso em: 28 mar. 2017.

NANRATA MRUTHINTI. **PAYPAL**. [S.I], 2011. Disponível em: <http://www.namratam.com/paypal/>. Acesso em: 28 mar. 2017.

OLIVEIRA, Felipe. **Novas tecnologias identificam clientes em lojas e sugerem produtos**. 2015. Disponível em: <http://www1.folha.uol.com.br/mercado/2015/01/1575995-novas-tecnologias-identificam-clientes-em-lojas-e-sugerem-produtos.shtml>. Acesso em: 18 nov. 2017.

PAIVA, Fernando. **Adoção de beacons aumentará com Eddystone e Bluetooth 5, prevê Zebra**. [S.I], 2016. Disponível em: <http://www.mobiletime.com.br/27/06/2016/adocao-de-beacons-aumentara-com-eddystone-e-bluetooth-5-preve-zebra/443182/news.aspx>. Acesso em: 24 nov. 2017.

PAYPAL. **PayPal**. [S.I], 2009. Disponível em: <https://www.paypal.com/br/home>. Acesso em: 02 abr. 2017.

PESSOA, Leandro. **Introdução ao Bluetooth Smart (BLE)**. [S.I], 2016. Disponível em: <https://www.embarcados.com.br/bluetooth-smart-ble/>. Acesso em: 24 nov. 2017.

RAUH, Stephan. **Model-View-Whatever**. [S.I], 2015. Disponível em: <https://www.beyondjava.net/blog/model-view-whatever/>. Acesso em: 25 nov. 2017.

RODRIGUES, Fabio. **Mapa de Calor na loja**: Entenda a jornada do seu cliente. [S.I], 2017. Disponível em: <https://novida.com.br/blog/mapa-de-calor-na-loja/>. Acesso em: 24 nov. 2017.

SETTE, Sarah Bezerra Galindo et al. A importância do compartilhamento de informações para construção do conhecimento de extensionistas universitários e sua relevância para a comunidade. In: ENCONTRO DE INICIAÇÃO A DOCÊNCIA, 11., 2009, João Pessoa. **Anais**... João Pessoa: Universidade Federal da Paraíba, 2009. 2 p.

SILVA, Régis Barroso. **A Importância dos Sistemas de Informação para a Gestão das Empresas**. [S.I], 2011. Disponível em: <http://www.administradores.com.br/artigos/tecnologia/a-importancia-dos-sistemas-de-informacao-para-a-gestao-das-empresas/56331/>. Acesso em: 28 mar. 2017.

STAAB, Wayne. **Bluetooth 101**: Part IV. [S.I], 2013. Disponível em: <http://hearinghealthmatters.org/waynesworld/2013/bluetooth-101-part-iv/>. Acesso em: 24 nov. 2017.

TAVARES, Marcelo. **Cinco motivos para monitorar o fluxo de visitantes na loja**. [S.I], 2015. Disponível em: <http://onegociodovarejo.com.br/cinco-motivos-para-monitorar-o-fluxo-de-visitantes-na-loja/>. Acesso em: 18 nov. 2017.

TAVARES, Marcelo. **Monitoramento de visitantes ou contagem de fluxo no shopping**: Entenda as principais diferenças. [S.I], 2016. Disponível em: <http://www.abrasce.com.br/noticia/2049>. Acesso em: 18 nov. 2017.

TEIXEIRA, Fabrício. **Tudo o que você precisa saber para começar a brincar com iBeacons**. [S.I], 2014. Disponível em: <https://brasil.uxdesign.cc/tudo-o-que-você-precisa-saber-para-começar-a-brincar-com-ibeacons-fdf5847e640b>. Acesso em: 27 maio 2017.

VENTURA, Felipe. **PayPal Beacon**: fazendo pagamentos sem utilizar as mãos. [S.I], 2013. Disponível em: <http://gizmodo.uol.com.br/paypal-beacon-hands-free/>. Acesso em: 28 mar. 2017.

YAU, Nathan. **How to Make a Heatmap**: a Quick and Easy Solution. [S.I], 2010. Disponível em: <http://flowingdata.com/2010/01/21/how-to-make-a-heatmap-a-quick-and-easy-solution/>. Acesso em: 18 nov. 2017.

ZANINI, Michel. **Introdução ao Spring Web Services**. [S.I], 2006. Disponível em: <https://www.devmedia.com.br/introducao-ao-spring-web-services/13921>. Acesso em: 18 nov. 2017.

APÊNDICE A – Detalhamento dos casos de uso

A seguir são detalhados os casos de uso da ferramenta. No Quadro 8 é apresentado o caso de uso de Manter Beacons. No Quadro 9 é apresentado o caso de uso de Montar página HTML beacon. No Quadro 10 é apresentado o caso de uso de Visualizar mapa de calor dos beacons. No Quadro 11 é apresentado o caso de uso de Acessar página HTML do beacon.

Quadro 8 - Detalhamento do UC01

Caso de uso	Manter beacons.
Descrição	Este caso de uso descreve os procedimentos para a criação, exclusão e alteração do cadastro de beacons.
Pré-condição	A ferramenta executando em um navegador. Para a funcionalidade é necessário que o usuário esteja autenticado e que o mesmo seja administrador.
Pós-condição	Em caso de inserção, beacon passa a existir no mapa de calor. Em caso de exclusão, beacon não aparece no mapa de calor. Em caso de alteração, não há.
Cenário principal	1. Escolha opção: 1.1. Usuário escolhe variante "Inserir" 1.2. Usuário escolhe variante "Alterar" 1.3. Usuário escolhe variante "Excluir" **Variante 1.1: Inserir** 1.1.1. o administrador escolhe a funcionalidade de inserir beacon. 1.1.2. o sistema redireciona para a tela de cadastro de beacon. 1.1.3. o administrador preenche o campo de UID. 1.1.4. o administrador preenche o campo de URLID. 1.1.5. o administrador escolher a situação do beacon(ativo ou inativo). 1.1.6. o administrador seleciona uma posição na planta baixa do local onde o beacon ficará posicionado através de um clique. 1.1.7. o sistema armazena a posição selecionada e preenche os campos de posição X e posição Y automaticamente. 1.1.8. o administrador confirma a ação. 1.1.9. o sistema valida os dados do beacon. 1.1.10. o sistema grava os dados do beacon e insere ele no mapa de calor. **Variante 1.2: Alterar** 1.2.1. o administrador escolhe a funcionalidade de alterar do beacon. 1.2.2. o sistema redireciona para a tela de cadastro de beacon com os dados do beacon escolhido carregados. 1.2.3. o administrador altera qualquer dado do beacon. 1.2.4. o administrador confirma a ação. 1.2.5. o sistema valida os dados do beacon. 1.2.6. o sistema grava os dados do beacon. **Variante 1.3: Excluir** 1.3.1. o administrador escolhe a funcionalidade de excluir do beacon. 1.3.2. o sistema abre a tela de confirmação. 1.3.3. o administrador confirma a ação clicando em 'sim'. 1.3.4. o sistema deleta os dados do beacon e apaga ele do mapa de calor.
Caminhos alternativos	1.1.8 o sistema verifica que algum campo não foi preenchido e avisa o administrador. 1.3.3 o administrador nega a ação clicando em 'não'. 1.3.4 o sistema não apaga o beacon.

Fonte: elaborado pelo autor.

Quadro 9 - Detalhamento do UC02

Caso de uso	Montar página HTML beacon.
Descrição	Este caso de uso descreve os procedimentos para a montagem de uma página HTML de um beacon.
Pré-condição	A ferramenta executando em um navegador. Para a funcionalidade é necessário que o usuário esteja autenticado e que o mesmo seja administrador. É necessário que o beacon desejado esteja cadastrado.
Pós-condição	A página HTML montada fica disponível para visualização
Cenário principal	1. o administrador clica no botão "Editar página" do beacon desejado. 2. o sistema redireciona para a tela de edição de página HTML. 3. o sistema carrega a estrutura da página pré-editada anteriormente. 4. o administrador escolhe o componente no lado esquerdo da tela. 5. o administrador arrasta o componente para a área de edição. 6. o administrador edita as propriedades do componente. 7. o sistema aplica as novas propriedades ao componente. 8. o administrador clica no botão salvar. 9. o sistema grava a estrutura da página do beacon.
Caminhos alternativos	2. o sistema carrega uma página de página vazia. 6. o administrador arrasta o componente para a lixeira 7. o sistema apaga o componente da tela. 8. o administrador volta a tela anterior. 9. o sistema não grava nada e mantem a estrutura anterior se houver.

Fonte: elaborado pelo autor.

Quadro 10 - Detalhamento do UC04

Caso de uso	Visualizar mapa de calor dos beacons.
Descrição	Este caso de uso descreve os procedimentos para a visualização do mapa de calor gerado pelos acessos as páginas HTML dos beacons.
Pré-condição	A ferramenta executando em um navegador. Para a funcionalidade é necessário que o usuário esteja autenticado e que o mesmo seja administrador.
Pós-condição	O usuário administrador visualiza o mapa de calor.
Cenário principal	1. o administrador clica na opção de menu botão "Mapa de calor". 2. o sistema redireciona para a tela do mapa de calor dos beacons. 3. o sistema carrega os acessos de todos os beacons com situação "Ativado". 4. o sistema monta o mapa de calor acima da imagem da planta baixa do local. 5. o administrador visualiza o mapa de calor.
Caminhos alternativos	Não há.

Fonte: elaborado pelo autor.

Quadro 11 - Detalhamento do UC05

Caso de uso	Acessar página HTML do beacon
Descrição	Este caso de uso descreve os procedimentos para o acesso a página HTML do beacon.
Pré-condição	Usuário com dispositivo Android. A ferramenta executando em um navegador.
Pós-condição	O usuário acessa a página HTML do beacon.
Cenário principal	1. o visitante visualiza a notificação no dispositivo android resultando o beacon modelo eddystone. 2. o usuário acessa a notificação. 3. o dispositivo redireciona para a página HTML do beacon. 4. O usuário acessa a página HTML do beacon 5. O sistema gera um registro de acesso a página HTML.
Caminhos alternativos	Não há.

Fonte: elaborado pelo autor.