UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

ALLERGYBEACON: APLICATIVO PARA IDENTIFICAÇÃO DE ALERTA DE ALERGIAS DE PACIENTES VIA BEACONS

JUAN PETER NUNES

BLUMENAU
2016

JUAN PETER NUNES

ALLERGYBEACON: APLICATIVO PARA IDENTIFICAÇÃO DE ALERTA DE ALERGIAS DE PACIENTES VIA BEACONS

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

Profª Luciana Pereira de Araújo - Orientadora

BLUMENAU
2016

ALLERGYBEACON: APLICATIVO PARA IDENTIFICAÇÃO DE ALERTA DE ALERGIAS DE PACIENTES VIA BEACONS

Por

JUAN PETER NUNES

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:	Prof.ª Luciana Pereira de Araújo, Mestra – Orientadora, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB

Blumenau, 05 de dezembro de 2016

Dedico este trabalho aos meus familiares amigos, professores e todos que de alguma forma contribuíram para a realização deste.

AGRADECIMENTOS

A Deus, por todo seu amor.

À minha família, que mesmo distante sempre me apoiou.

Aos meus amigos, pelos conselhos, ajudas e empurrões.

Ao meu amigo Herbert Treis Neto que disponibilizou seus dispositivos *beacons* para fazer testes deste trabalho.

A minha orientadora, professora Luciana Pereira de Araújo que me auxiliou nessa jornada de forma positiva.

Nenhum vencedor acredita no acaso.

Friedrich Nietzsche

RESUMO

O presente trabalho descreve o desenvolvimento de um aplicativo móvel que visa auxiliar profissionais de saúde, apresentando as informações sobre alergias à medicamentos de pacientes por alerta utilizando dispositivos *beacons*. Este alerta depende de informações pré–cadastráveis por um administrador. O aplicativo retorna essas informações utilizando da tecnologia *Bluetooth Low Energy* (BLE) e é compatível com dispositivos *beacons*. Para seu desenvolvimento foi utilizado a ferramenta de desenvolvimento Delphi 10 Seattle com componentes FireDac, DataSnap e banco de dados MySQL. Como resultado se obteve um aplicativo capaz de se comunicar com informações cadastráveis e vinculadas a dispositivos *beacons* reais. Sendo assim, é possível a integração a um sistema legado onde as informações alérgicas de pacientes são existentes.

Palavras-chave: Beacon. Bluetooth. Delphi 10. Alergia. Medicamentos. DataSnap.

ABSTRACT

This paper describes the development of a mobile application that aims to assist health professionals, generating the allergic information to patients' medications by alerting using beacons devices in an agile way. This alert depends on information pre-registered by an administrator. The application returns this information using Bluetooth Low Energy (BLE) technology and is compatible with beacons. For its development the Delphi 10 Seattle development tool was used with components, FireDac, DataSnap and MySQL database. As a result, it obtained an application capable of communicating with information that can be registered and linked to real beacons in a fast and agile way. It is possible in the future to be integrated into a legacy system where patient allergy information is available.

Key-words: Beacon. Bluetooth. Delphi 10. Alergy. Drugs. DataSnap.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASBAI – Associação Brasileira de Alergia e Imunopatologia

BLE – Bluetooth Low Energy

CRUD – Create Read Update Delete

HTTP – HyperText Transfer Protocol

IDE – Integrated Development Enviroment

IOS – Iphone Operating System

IOT – Internet Of Things

IP – Internet Protocol

JSON – JavaScritp Object Notation

KBPS – Kilobit Per Second

M2M – Machine to Machine

MER – Modelo de Entidade e Relacionamento

MHz – MegaHertz

NFC – Near Field Communication

P2P – Peer to Peer

QR Code – Quick Response code

RF – Radio Frequency

RFID – Radio Frequency IDentification

SAMU – Sistema de Atendimento Móvel de Urgência

SQL – Stuctucred Query Language

SUS – Sistema Único de Saúde

TCP – Transmission Control Protocol

UUID – Universally Unique IDentifier

WIFI – WIreless FIdelity

WPAN – Wireless Personal Area Network

SUMÁRIO

1 INTRODUÇÃO

As informações contidas nas prescrições médicas são importantes para facilitar a comunicação entre os profissionais de saúde e paciente (NÉRI et al. 2011). Essas informações se tornaram ainda mais importantes com a visibilidade dada pelos sistemas de informação. A combinação de computadores com as informações médicas e os dados eletrônicos dos pacientes podem melhorar muito a qualidade e o apoio a decisões inerentes (RODRIGUES FILHO et al., 2001).

Conforme ASBAI (2012, p. 1), "dados apontam que 30% da população brasileira possui algum tipo de reação alérgica". Essas alergias do paciente normalmente são de difícil acesso por estarem em um prontuário de papel, escrito manualmente, ou em sistemas que contemplam a informação junto com todos os demais dados e consultas do paciente. Esta informação, quando está registrada, normalmente é de difícil acesso pois não está atrelada a uma tecnologia que permita sua visualização no momento da prescrição medicamentosa (NÉRI et al, 2011).

Embora já existam formas do profissional de saúde visualizar as alergias do paciente, estas formas ainda não são tão eficientes (NÉRI et al., 2011). Isso pode ser afirmado, pois, se a informação for registrada no prontuário de papel, em situações de risco na qual o medicamento deve ser administrado em um curto intervalo de tempo, pode não ser possível consultar este prontuário. Em outras situações, quando a informação é registrada em um sistema, sendo ele desktop, *web*, a informação está misturada com outras informações do paciente. Essa carga excessiva de informações nos sistemas atuais de gestão acaba não facilitando a usabilidade da gestão assistencial do profissional de saúde. Outro exemplo são as aplicações móveis no qual o paciente pode cadastrar sua própria alergia (CASTRO, 2015). Porém, em caso de o paciente estar impossibilitado de falar ou seu smartphone estiver bloqueado, os aplicativos existentes não auxiliarão o profissional de uma maneira fácil.

As informações sobre alergia do paciente são de alto nível de periculosidade devido a tratar da vida do paciente, pois está vulnerável a possíveis choques anafiláticos (ASBAI, 2012). Essas informações são ainda mais importantes em casos de leito hospitalar, pois em alguns casos os pacientes não conseguem se expressar para passar essa informação ao médico ou enfermeira que irá prescrever e aplicar determinado medicamento. Segundo Néri et al. (2011, p. 313), "O processo de prescrição é complexo e permeado por erros. Os erros de prescrição são geralmente multifatoriais e originários de falhas ativas ou condições que induzem ao erro". Uma pesquisa realizada em 2000, indica que em termos mundiais, morrem

entre 44 mil e 98 mil pessoas por danos decorrentes de erros médicos e cerca de 7 mil destas mortes são consideradas mortes por erros de medicação (KOHN; CORRIGAN; DONALDSON; 2000).

Na oportunidade de facilitar o acesso a estas informações, este trabalho apresenta o desenvolvimento de uma aplicação móvel que centraliza a informação da alergia do paciente e apresente de maneira facilitada ao profissional de saúde. A aplicação é voltada para a identificação de alergias a medicamentos em pacientes que se encontram em leitos hospitalares. Ao identificar o leito, a aplicação apresenta o paciente e suas deficiências alérgicas (quando houver). Desta maneira, a aplicação apoia a prescrição e administração do medicamento apresentando ao profissional a informação sobre suas alergias de forma facilitada. Para a identificação do leito, a aplicação se comunica através da tecnologia *beacons*. Com eles, tem-se o objetivo de identificar o leito por proximidade do dispositivo móvel ao leito. Dessa forma, este trabalho aborda uma aplicação na área de Internet das Coisas (IoT), através da *bluetooth* entre o aplicativo e os *beacon*s, dentro de um ambiente hospitalar.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma aplicação móvel para apresentar ao profissional de saúde as alergias de um paciente ao se aproximar de seu leito, utilizando a tecnologia *beacons* através de um dispositivo móvel.

Os objetivos específicos do trabalho proposto são:

a) permitir a visualização de alergias de pacientes de forma facilitada;
b) testar a viabilidade de trabalhar com beacons através da plataforma Delphi 10 Seattle;
c) verificar o uso da tecnologia *beacon* em um experimento de laboratório.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos. O primeiro capítulo aborda a introdução e objetivos do trabalho. O segundo capítulo apresenta a fundamentação teórica responsável por conter as principais referências teóricas para desenvolvimento do trabalho. O terceiro capítulo aborda o desenvolvimento, onde é possível visualizar as ferramentas, levantamento de informações e a operacionalidade da aplicação. O quarto e último capítulo apresenta a conclusão do trabalho junto a suas possíveis extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda assuntos a serem apresentados nas seções a seguir. A seção 2.1 trata sobre o ambiente hospitalar em que pacientes não tem condições de falar, e os profissionais precisam verificar suas alergias. A seção 2.2 aborda sobre Internet das Coisas (IoT). Na seção 2.3 é abordado sobre a ferramenta Delphi 10 *S*eattle com foco no componente de *Internet of Things* (IOT). Formas de comunicação com o meio externo é apresentado na seção 2.4, incluindo as subseções Bluetooth, *beacon*, Near Field Communication (NFC) e Quick Response Code (QR Code). Por fim, a seção 2.5 apresenta os trabalhos correlatos e uma correlação comparativa entre eles.

2.1 AMBIENTE HOSPITALAR

Rodrigues Filho (2001, p. 105) afirma que "a combinação de computadores, redes de telecomunicações, informações médicas online e dados eletrônicos de pacientes pode melhorar a qualidade e as decisões inerentes ao cuidado de saúde, além de facilitar o acesso aos serviços disponíveis". Sendo assim, enfatiza-se que a automação de sistemas de prontuário do paciente contribui para melhorar o atendimento do paciente.

Segundo Cruciol et al. (2008, p. 192), "a prescrição exerce um papel fundamental no tratamento medicamentoso. Um primeiro fator que lhe confere tanta importância, principalmente no ambiente hospitalar, é a interligação que ela exerce entre toda a equipe de saúde". Muitas vezes, a etapa de orientação sobre os medicamentos acaba sendo omitida, e grande parte das dúvidas a respeito de uma prescrição médica incompleta ou mal grafada necessita ser revisada (CRUCIOL et al., 2008, p. 192). Há situações em que o paciente está incapaz de se comunicar e precisa ser medicado com rapidez, onde poderá haver riscos de reações alérgicas a medicações ministradas incorretamente, devido à urgência.

> A administração do medicamento ao paciente errado tem sido a causa de 5% dos erros relatados com medicamentos. A falta do número de registro do paciente no hospital e/ou do quarto e leito ocupados por ele pode contribuir para a ocorrência deste erro no ambiente hospitalar (CRUCIOL et al., 2008, p. 193).

Em algumas vezes, as informações sobre alergias constam no prontuário do paciente. Entretanto, essas informações podem estar incompletas e serem de difícil acesso ao profissional de medicina em uma situação de urgência. Na maioria dos hospitais brasileiros, o prontuário eletrônico ou de papel é mal organizado e de baixo padrão, funcionando de forma precária na maioria dos hospitais (RODRIGUES FILHO, 2001).

2.2 INTERNET DAS COISAS

Internet das coisas ou *Internet of Things* (IoT) consiste em poder conectar objetos como sensores, *tags Radio-Frequency IDentification* (RFID), *smarthphones*, computadores, *beacons* e até objetos de uso comum entre si (AFONSO; PEREIRA R.; PEREIRA M., 2015). Geralmente estes objetos estão relacionados a facilitar a vida do ser humano de modo que tome decisões de forma autônoma. Por exemplo, um smartphone pode enviar a informação a um sistema elétrico de uma casa que está se aproximando através de posicionamento de *Global Positioning System* (GPS) e acionar as luzes ou até mesmo a energia sem a necessidade de interferência humana. Nesse exemplo pode–se concluir que a IOT exerce vantagem em dois aspectos, primeiro na facilidade do acionamento da luz e segundo a sustentabilidade de energia com o total desligamento e acionamento do sistema da casa ao se aproximar.

Segundo Silva et al. (2016, p. 1), "O termo Internet das Coisas (IOT) surgiu em meados de 1999, a partir de ideias do pesquisador britânico, Kevin Ashton, do Instituto de Tecnologia de Massachusetts". Incialmente o termo IOT se referiu a etiquetar eletronicamente produtos de uma linha de produção de uma empresa utilizando identificadores de radiofrequência (SILVA et al., 2016). Silvia et al. (2016 p. 1), completa;

> Apesar de o termo estar presente desde 1999, essa ideia já está presente a algum tempo, não de forma tão clara, mas está como, por exemplo, nos caixas eletrônicos bancários, onde a máquina lê a identificação do cartão do usuário, envia as informações à central bancária, verifica o saldo e libera ou não o saque, sem a necessidade de interação humana, tudo de forma automatizada.

Afonso, Pereira e Pereira (2015, p. 184) alertam que o termo IoT "confunde-se com o termo Comunicações *Machine-to-Machine* (M2M *Communications*), que trata da comunicação máquina a máquina, entre dois ou mais dispositivos, sem um usuário final envolvido no processo da troca de informações". Desta maneira é entendível que IoT engloba a interligação de objetos, a maneira como os humanos interagem com eles e o ambiente. Já M2M é onde máquinas utilizam recursos de rede para se comunicar para efeitos de monitoração ou controle de um ambiente.

2.3 DELPHI 10 SEATTLE COMPONENTES IOT

O ambiente de programação Delphi é baseado na linguagem de programação *Object Pascal*, oriunda da linguagem Pascal, a qual foi projetada pelo Professor Niklaus Wirth, professor da Universidade de Zurique (Suíça) (MENEZES; MUNIZ, 2014). Segundo

Menezes e Muniz (2014) houve várias outras versões com mudanças impactantes para o desenvolvimento, uma delas ocorreu na versão XE6. Nesta versão foi identificada uma maneira mais rápida para desenvolver um único código nas plataformas Windows, Mac iOS e Android. Atualmente a versão atual do Delphi é a Delphi 10 Seattle ou Delphi RX.

O Delphi 10 Seattle é a ferramenta de desenvolvimento multiplataforma da Embarcadero Technologies. Com ela é possível desenvolver aplicações nativas para as plataformas Win32, Win64, Android, IOS32, IOS64 e OSX, a partir de um único código fonte (CARREIRO, 2015).

Nesta versão foi possível notar a variedade de componentes voltados a resolver problemas interligados a objetos comuns. Sendo possível identificar que a Embarcadero está procurando soluções voltadas para IoT.

> Embarcadero Technologies respira a vida nova em aplicações existentes, permitindo que os desenvolvedores possam conectar facilmente seus aplicativos para aparelhos, dispositivos, sensores, dados empresariais e serviços em nuvem. Construir soluções da Internet das coisas completas que proporcionam experiências inovadoras (EMBARCADERO, 2015).

Um dos componentes de destaque é o *BeaconFence* que permite desenvolver uma aplicação que forneça o rastreio de seu *beacon* sem a necessidade de utilizar o GPS. Ainda a fabricante garante que é possível criar mapas carregando uma imagem existente, como uma planta de escritório, e configurar o evento *OnZoneEnter* do componente para disparar alguma ação. Este evento é acionado quando o usuário entra em uma zona pré-definida na aplicação (EMBARCADERO, 2016).

2.4 FORMAS DE COMUNICAÇÃO COM MEIO EXTERNO

Esta seção aborda alternativas de tecnologias que permitem a interação entre objetos e humanos no ambiente em que se encontram. Foram levadas em consideração para o desenvolvimento deste trabalho, Bluetooth, QR Code e NFC.

2.4.1 BLUETOOTH

Segundo Menegotto (2015, p. 2), “Os sistemas de localização e microlocalização em ambientes fechados (*Indoor Positioning System*) têm sido programados utilizando diversas infraestruturas”. Dentre os sistemas que foram propostos, alguns são baseados em Rádio Frequência (RF) (*RADAR, PlaceLabs, Herecast, Ekahau, LandMarc, Moca*); outros são baseados em sinais de ultrassom (*Cricket Location System, Active Bat*) e há ainda os que utilizam sinais infravermelhos (*Active Badge Location System*) (MENEGOTTO, 2015).

Conforme Menegotto (2015), em anos recentes, smartphones tem sido equipados com funções específicas que permitem o desenvolvimento de sistemas para localização do usuário ou objetos. Algumas dessas funções são a leitura e triangulação de sinais RF das redes WIFI, assim como soluções combinadas com acelerômetros e outros sensores do aparelho (MENEGOTTO, 2015).

A definição de Bluetooth para Manganelli (2014, p. 23) consiste em "ser um padrão de rede sem fio ligado às redes pessoais sem fio, também conhecida como WPAN". O surgimento dessa rede aconteceu em meados de 1994 com a Ericsson estudando uma forma de comunicação entre aparelhos celulares e acessórios com sinais de rádio (MANGANELLI, 2014). A versão mais recente da tecnologia Bluetooth é chamada de Bluetooth 4.0 ou *Bluetooth Low Energy* (BLE). Sua principal característica condiz com seu nome, baixo consumo de bateria utilizando a frequência de 2,4 GHZ. Segundo Maio (2014, p. 3), BLE "está otimizada para ser usada em dispositivos que tenham que operar durante longos períodos de tempo sem que seja necessário substituir ou recarregar as suas baterias". Um exemplo de dispositivo operante na tecnologia BLE são os *beacons*.

2.4.1.1 BEACON

Beacons são dispositivos de proximidades em ambientes fechados que permitem localizar objetos ou pessoas que carreguem objetos (TEIXEIRA, 2014). Pode-se concluir que os *beacons* estão para ambientes fechados assim como os GPS estão para ambientes externos.

O *Estimote Beacon*, modelo utilizado para a realização desse trabalho, é considerado um *iBeacon* por se comunicar através do padrão tecnológico desenvolvido pela Apple (ROCHA et al., 2014). Na Figura 1 pode-se visualizar um *beacon* do modelo *Estimote* em visão explodida.

> O *Estimote Beacon* trata-se de um pequeno super-computador, possuindo um processador 32-bit ARM® Cortex M0 CPU com 256kB memória flash, acelerômetro, sensor de temperatura e 2.4 GHz Bluetooth 4.0 Smart, também conhecido como BLE ou Bluetooth Low Energy, envoltos por uma estrutura flexível feita de silicone. (ROCHA et al., 2014).

Figura 1 – *Beacon estimote* em visão explodida.

Fonte: Rocha (2014).

Ainda segundo Rocha et al (2014, p. 28), "O dispositivo *Estimote* utiliza a comunicação BLE através de um formato padronizado pela *Apple* no qual cada pacote transmitido consiste em cinco partes principais de informação: ".

a) prefixo iBeacon (9 bytes): string padronizada pela *Apple*;
b) UUID (16 bytes): utilizado para diferenciar um grande grupo relacionado de *beacons*;
c) major (2 bytes): utilizado para diferenciar pequenos grupos de *beacons*;
d) minor (2 bytes): identificador individual de cada *beacon*;
e) tx power (2 bytes): definido com a força do sinal a um metro de distância do aparelho;

Na Figura 2 está representado esquematicamente o pacote de dados transmitido.

Figura 2 – Representação pacote de dados transmitido pelo *Estimote beacon*.

Prefixo iBeacon (9 bytes)	UUID (16 bytes)	Major (2 bytes)	Minor (2 bytes)	TX Power (2 bytes)

Fonte: Rocha (2014).

O pacote segue inicialmente com o prefixo iBeacon, UUID, *major*, *minor*, *tx power*. Cada *Estimote beacon* vem com um UUID fixo e major e minor randomizados. É possível também alterar o UUID, *major* e *minor* do *Estimote beacon*, porém se faz necessário realizar uma autenticação pela fabricante em sua plataforma web chamada *Estimote Cloud* (STECZKIEWICZ, 2016).

2.4.2 NFC

Segundo Cinelli e Duarte (2013), *Near Field Communication* (NFC) é uma comunicação de transmissão de dados de forma segura e sem fio a distâncias curtas, geralmente menores que 10 centímetros.

> *Near Field Communication* é um subconjunto de padrões RFID, que estabelece uma comunicação de curto alcance sem fio que depende de modulação de radiofreqüência (RF), operando a 13,56 MHz com uma distância de trabalho até 10 centímetros suportando uma taxa máxima de dados de 424 kbps. (CINELLI; DUARTE, 2013).

As duas principais maneiras de se realizar a comunicação segundo Rocha et al. (2014), é o modo leitura e escrita que permite a dispositivos NFC lerem e escrever neles mesmo. A outra é o modo Peer to Peer (P2P) que permite que dispositivos troquem dados uns com os outros. Alguns exemplos desse tipo de comunicação são encontrados em transações monetárias com cartões de crédito e débito, tickets de estacionamento ou crachás.

Um ponto positivo para o NFC é sua segurança. Os aparelhos precisam estar muito próximos para o pareamento do aparelho ao receptor acontecer. É necessário fazer uso de criptografia e autenticação que pode ser solicitada pelo iniciador antes de iniciar a comunicação e a transferência de informações (PALMEIRA; FERNANDES, 2013, p. 19).

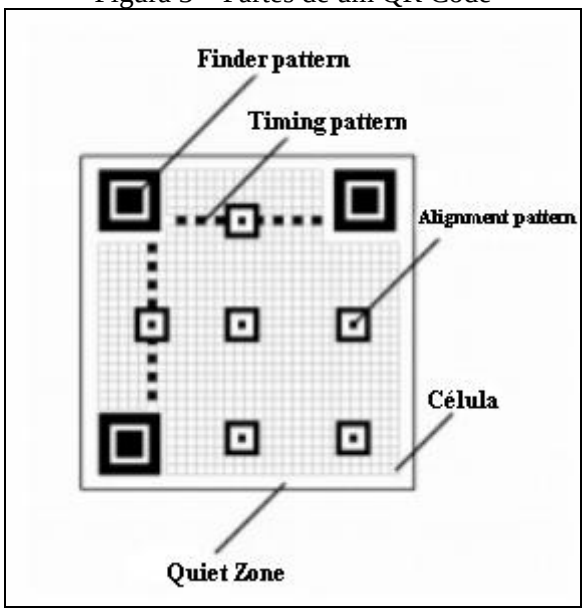
Pode-se concluir que o NFC se destaca para operações que dependam de até dois mecanismos simultaneamente, sendo um para o emissor e outro para o receptor.

2.4.3 QR CODE

O *Quick Response Code* (QR Code) é um código de barras bidimensional desenvolvido pela empresa Denso Wave, subsidiária da Toyota, com intuito de rastrear os veículos de produção. Tornou–se o código de barras mais famoso devido sua potencialidade estratégica (COPETTI; GHISLENI, 2013, p. 65). Segundo Rocha et al. (2014),

> Um QR Code possui uma capacidade máxima de armazenamento de informação de 4296 caracteres alfanuméricos, entretanto toda sua superfície não é utilizada apenas como área de dados, ela é subdivida em espaços menores voltados para facilitar a leitura, proporcionar outras funcionalidades e melhorar o seu desempenho, são os finder patterns, alignment patterns, timing patterns e a quiet zone.

Figura 3 – Partes de um QR Code



Fonte: Rocha (2014).

Na Figura 3 é possível visualizar as partes de um QR Code, que é composta por quatro partes; *finder pattern*, *alignment pattern*, *timing pattern* e a *quiet zone* (ROCHA et al. 2014). O *finder pattern*, representado pelos símbolos nas três extremidades da Figura 3, são responsáveis pela posição, tamanho e ângulo do QR Code. Assim cria a estrutura do QR Code para que possa ser lido de qualquer direção. Já o *alignment pattern* é responsável por auxiliar o software que vai decodificá-lo, corrigindo possíveis distorções. O *timing pattern* são linhas horizontais e verticais do código, também auxiliam na correção de distorções. Por último é apresentado a *quiet zone* que delimita o fim do QR Code.

Para leitura do código é necessário o uso de uma câmera, com algum aplicativo que contenha o leitor do código. Atualmente existem inúmeros aplicativos leitores devido a empresa Denso wave ter disponibilizado seu código fonte (KARASINSKI, 2013).

2.5 TRABALHOS CORRELATOS

Como trabalhos correlatos pode-se citar o aplicativo Alergia a Medicamentos (CASTRO, 2015), o sistema de gestão de saúde Philips Tasy (TASY, 2010), o sistema PRONTO que gere o Sistema Único de Saúde (SUS) de Blumenau (MATTOS et al., 2015) e os sistemas operacionais Android (HAMANN, 2016) e iOS (BIJORA, 2014) que possuem funcionalidades similares sobre informações de emergência.

2.5.1 ALERGIA A MEDICAMENTOS

O aplicativo Alergia a Medicamentos é um aplicativo disponível nas plataformas Android e iOS que tem como finalidade o cadastro e reconhecimento de medicamentos que potencialmente causam alergia para o usuário. O aplicativo sugere ao usuário que o paciente seja cadastrado logo ao acessá-lo, caso contrário, os medicamentos salvos poderão ser perdidos. Os medicamentos disponíveis no aplicativo já estão cadastrados em uma base de conhecimento e são exclusivamente comercializados no Brasil. Conforme Castro (2015), o aplicativo não substitui a opinião do médico, sendo apenas uma ferramenta de auxílio ao paciente alérgico e seu médico. O usuário cadastrado também pode cadastrar o nome e o telefone de seu médico. Nesta mesma tela também é disponibilizado o telefone do Serviço de Atendimento Móvel de Urgência (SAMU) previamente cadastrada (CASTRO, 2015).

Na Figura 4 são apresentadas as principais telas do aplicativo Alergia a Medicamentos. É possível visualizar sua tela de inicialização, tela de registro de medicamentos potencialmente alérgicos e a tela de informações úteis.

Figura 4 – Principais telas do aplicativo Alergia a Medicamentos

Fonte: Castro (2015).

A tela de registro de medicamentos potencialmente é onde pode–se vincular medicamentos ao usuário venha causar algum risco ao usuário. A quantidade de registros medicamentos a ser vinculado é ilimitado. O sistema indica um tempo de espera de dez a trinta segundos sobre a consulta do medicamento à sua base de dados local, fazendo com que o sistema não dependa do acesso à internet para consultar dados. Na tela de informações úteis

além de cadastrar o nome do médico do usuário o aplicativo disponibiliza três campos para cadastro de telefones de contato do mesmo.

2.5.2 TASY

O sistema Philips Tasy é um sistema de gestão de saúde desktop onde é possível realizar a gestão de diversas áreas hospitalares. Dentre as áreas destacam-se a área clínica, banco de sangue, laboratório e *homecare*. Nele é possível gerenciar tanto a parte assistencial de um hospital quanto a parte financeira do mesmo. Conforme Tasy (2010), o sistema permite realizar o cadastro das alergias de medicamentos do paciente junto a prescrição eletrônica. Este cadastro permite alertar o prescritor com a relação de possíveis alergias potencialmente prejudiciais ao paciente. As informações vinculadas a prescrição são bastante genéricas devido a cada caso e paciente, portanto é possível cadastrar diversas informações como medicamentos, materiais utilizados, altura, peso, exame, procedimentos realizados, coleta de sangue, entre outros (TASY, 2010).

2.5.3 PRONTO

O sistema PRONTO é um sistema de gestão de saúde que visa implementar o conceito de Prontuário Único do Paciente (PUP) aplicado na rede pública de saúde com foco em Blumenau (ARAÚJO; BERKENBROCK; MATTOS, 2014). O PRONTO é dividido em módulos, os quais são visualizados de acordo com o perfil do profissional que está o utilizando. Nas telas de consulta médica e de enfermagem, os profissionais possuem um espaço para a adição das alergias medicamentosas dos pacientes, além das demais informações referentes a própria consulta. Uma vantagem do preenchimento das alergias é que os medicamentos que são fornecidos pelo SUS são pré-cadastrados. Além disso, o PRONTO possui o espaço para a prescrição de medicamentos, dessa forma, quando o profissional preenche a receita médica com algum medicamento que o paciente é alérgico, o profissional já é notificado, evitando assim o erro na prescrição (MATTOS et al., 2015). Porém, apesar da prescrição controlar este fator, o PRONTO não é utilizado para o atendimento em leitos hospitalares, e se fosse, ainda assim teria que ser consultado o prontuário do paciente antes de realizar um procedimento medicamentoso.

2.5.4 INFORMAÇÕES DE EMERGÊNCIA (ANDROID & iOS)

O sistema operacional iOS da Apple, possui a capacidade de armazenar as informações alérgicas de seus usuários em seus dispositivos. Além da informação alérgica é possível

cadastrar o grupo sanguíneo, se o usuário é doador de órgãos, peso, altura e o telefone do médico para emergências (BIJORA, 2014). A opção existe a partir da versão oito do sistema operacional e as informações podem ser acessadas por qualquer pessoa com o dispositivo bloqueado. Na Figura 5 pode-se visualizar a opção das informações alérgicas do paciente através do sistema.

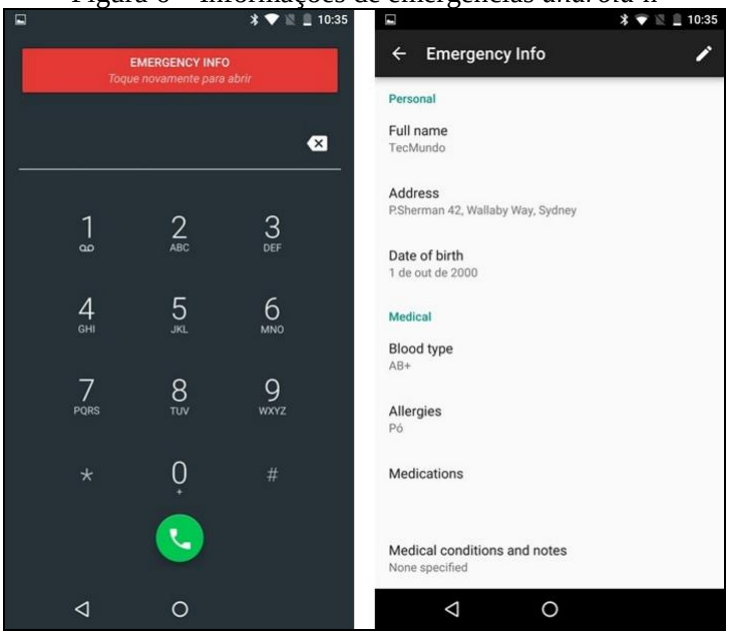
Figura 5 – Opção de visualização das informações alérgicas no iOS.

Fonte: Bijora (2014).

Para cadastrar as informações no iOS basta acessar o aplicativo Saúde, que é nativo do sistema e logo em seguida o menu ficha médica.

Já no Android existe uma função semelhante ao seu concorrente iOS. As informações armazenadas são endereço, nome, data de nascimento, tipo sanguíneo, alergias, medicação e notas. Essa funcionalidade está disponível a partir da versão 7.0 e também pode ser acessada com o dispositivo bloqueado (HAMANN, 2016). Na Figura 6 é apresentada a tela onde do sistema operacional bloqueado e as informações de emergência.

Figura 6 – Informações de emergências *android* n



Fonte: Hamann (2016).

As informações de emergência do sistema operacional Android são acessadas pelo aplicativo de configuração. É preciso selecionar o menu usuários e em seguida informações de emergência. Neste cadastro ainda é possível dizer se a informação deverá aparecer mesmo com dispositivo bloqueado ou não.

2.5.5 Comparativo dos trabalhos correlatos

Esta seção tem como objetivo realizar um comparativo entre os trabalhos correlatos e suas funcionalidades. As descrições e características, pontos fortes e fracos de cada trabalho é apresentado a seguir.

A seção 2.5.1 apresentou a aplicação Alergia a Medicamentos, onde é possível cadastrar as alergias do usuário. A aplicação tem como objetivo disponibilizar a informação alérgica de medicamentos em situações de emergência, quando o usuário está impossibilitado. Um ponto forte da aplicação é de que possui uma vasta base de dados de medicamentos já disponível de forma off-line. Um ponto negativo é de que usuário depois de cadastrar precisa consultar a própria informação cadastrada, dificultando a visualização desta informação. Desta forma a aplicação se mostra mais eficiente a terceiros que possam vir a ajudar o usuário em questão.

A seção 2.5.2 apresentou o sistema Tasy. Tasy é um sistema robusto que visa gerenciar diversos processos de um hospital de forma eficiente. A vasta gama de informação é um ponto forte do sistema, pois além da gestão assistencial o sistema fornece a gestão financeira de consultas, exames, procedimento cirúrgicos entre outros. Apesar disso, esta vasta gama de

informação se torna também um ponto negativo na usabilidade, pois em meio a toda ela o profissional deve ir atrás da informação para que possa encontrá-la. Este processo pode ser demorado dificultando o apoio a decisão do profissional.

A seção 2.5.3 apresenta o sistema PRONTO. O sistema visa garantir a gestão da saúde no Sistema Único de Saúde (SUS) de Blumenau. As prescrições do pronto possuem validações na hora de prescrever um medicamento quanto a alergia do paciente, garantindo com assertividade que o medicamento não irá apresentar risco a vida do paciente. Quanto a utilização do sistema, não é feita em leitos e se fosse o profissional teria de ir atrás da informação.

A seção 2.5.4 apresenta os sistemas operacionais móveis Android e iOS. Neles é possível informar as informações alérgicas do usuário da mesma forma que a aplicação Alergia a Medicamentos. Porém não existe uma base de dados com medicamentos, o campo é preenchido apenas uma breve descrição do medicamento. Um ponto positivo é sua disponibilidade caso o dispositivo esteja bloqueado, podendo auxiliar socorristas em casos de acidente por exemplo.

3 DESENVOLVIMENTO

Neste capítulo estão descritas as particularidades técnicas da aplicação. O capítulo está dividido em quatro subseções. A seção 3.1 apresenta o levantamento de informações. A seção 3.2 apresenta as especificações dos requisitos. Detalhes do desenvolvimento são apresentados na seção 3.3 que também é dividida nas subseções 3.3.1 técnicas e ferramentas utilizadas e 3.3.2 que apresenta a operacionalidade da aplicação. Por fim a seção 3.4 apresenta os resultados e discussões do desenvolvimento desta aplicação.

3.1 LEVANTAMENTO DE INFORMAÇÕES

Neste trabalho é apresentado o desenvolvimento de uma aplicação para dispositivos móveis, que possa auxiliar o profissional de saúde quanto a identificação de alergias de pacientes em leitos hospitalares. Em situações de emergência as informações necessárias para a tomada de decisão, às vezes, pode estar de difícil acesso, como em sistemas desktop ou prontuários de papel. O aplicativo permite que ao invés do médico ir atrás da informação armazenada em algum local, ele mesmo trará a informação ao profissional com assertividade.

Para o desenvolvimento da aplicação é utilizada a tecnologia *beacons* através da comunicação BLE. Está tecnologia foi escolhida para fins de testes dos componentes *beacons*. Outro fator que foi levado em consideração foi a distância de aproximação do dispositivo móvel ao leito do paciente. Este fator foi determinante para o descarte da tecnologia Near Field Communication (NFC) pois a mesma tem a limitação de distâncias até dez centímetros. Já a tecnologia QR Code foi descartada devido sua dependência de uso de câmera do dispositivo, obrigando o usuário a acessá-la e posicioná-la corretamente para escaneamento total da imagem do código de barras.

A aplicação permite o cadastro de medicamentos que possam causar algum tipo de reação alérgica ao paciente. Os cadastros essenciais para o funcionamento da aplicação dependem unicamente do profissional de saúde. A aplicação também permite cadastrar os leitos e seus pacientes internados, para que assim sejam cadastradas suas alergias e medicações de risco alérgico.

A infraestrutura necessária para o funcionamento correto da aplicação é relativamente simples, necessitando somente de um dispositivo *beacon* para cada leito e um dispositivo móvel com Android para o profissional de saúde. Também é obrigatório que este dispositivo móvel contenha driver de *bluetooth* e esteja ativado para sua comunicação com os *beacons*.

Dessa forma, a aplicação será construída com base nas seguintes ferramentas:

a) Delphi Seattle 10;

b) smartphone com sistema operacional Android;
c) banco de dados MySQL no Servidor;
d) comunicação com *beacons* via BLE.

3.2 ESPECIFICAÇÃO DOS REQUISITOS

Esta seção é composta pelas subseções 3.2.1 Requisitos Funcionais (RF), 3.2.2 Requisitos Não Funcionais (RNF), 3.2.3 diagrama de Casos de Uso (UC), 3.2.4 diagrama de classes, 3.2.5 diagrama de atividades e a seção 3.2.6 Modelo de Entidade Relacional (MER).

Para a modelagem do diagrama de classes, casos de uso e atividades foi utilizado a ferramenta Enterprise Architect (EA). A modelagem do Modelo de entidade Relacional (MER) foi criada a partir da ferramenta DBDesigner.

3.2.1 REQUISITOS FUNCIONAIS

O Quadro 1 apresenta os Requisitos Funcionais (RF) previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o caso de uso associado.

Quadro 1 – Requisitos Funcionais

Requisitos Funcionais	**Caso de Uso**
RF01: O sistema deve permitir manter usuários.	UC01
RF02: O sistema deve permitir manter leitos.	UC02
RF03: O sistema deve permitir manter medicamentos.	UC03
RF04: O sistema deve permitir manter as alergias do paciente.	UC04
RF05: O sistema deve permitir manter pacientes.	UC05
RF06: O sistema deve permitir manter *beacons*.	UC06
RF07: O sistema deve emitir um alerta sobre as alergias do paciente ao se aproximar do leito.	UC07
RF08: O sistema deve permitir consultar os pacientes.	UC08
RF09: O sistema deve permitir consultar os medicamentos e compostos alérgicos do paciente.	UC09
RF10: O sistema deve permitir consultar leitos.	UC10
RF11: O sistema deve permitir consultar as alergias.	UC11
RF12: O sistema deve permitir ao usuário consultar *beacons*.	UC12

Fonte: Elaborado pelo autor.

Pode-se afirmar que todos os requisitos funcionais estão ligados ao cliente da aplicação. A parte do servidor não é de uso aos usuários finais, porém precisa estar ativo para o funcionamento correto do cliente.

3.2.2 REQUISITOS NÃO FUNCIONAIS

O Quadro 2 apresenta os requisitos não funcionais previstos para o sistema.

Quadro 2 – Requisitos não funcionais

Requisitos Não Funcionais
RNF01: O sistema deve ser desenvolvido na ferramenta Delphi 10 Seattle.
RNF02: O sistema deve se comunicar com *beacons* via bluetooth.
RNF03: O sistema deve ser executado na plataforma Android.
RNF04: O sistema deve utilizar o banco de dados MySQL como base de dados.

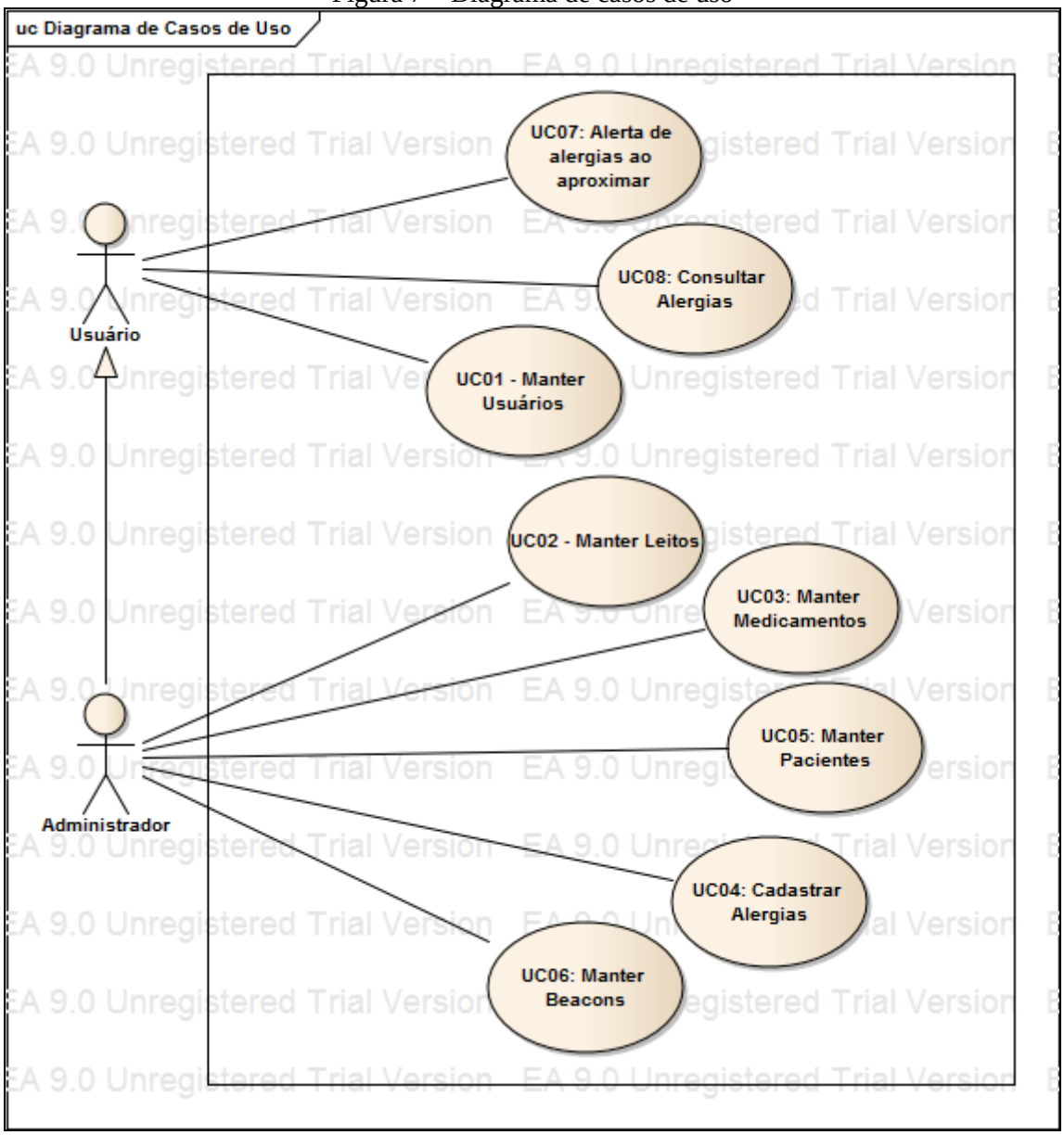
Fonte: Elaborado pelo autor.

O requisito não funcional RNF04 está relacionado a parte de servidor da aplicação. Já os requisitos RNF02 e RNF03 estão relacionados parte cliente. O requisito não funcional RNF01 está relacionado em ambas as partes (cliente e servidor).

3.2.3 DIAGRAMA DE CASOS DE USO

Nesta seção pode-se visualizar o diagrama de Casos de Uso (UC) do aplicativo. O diagrama apresenta dois atores sendo um denominado `usuário`, que se refere ao profissional de saúde e o outro denominado `administrador`, onde o mesmo tem acesso total ao aplicativo. O paciente, no contexto, não é usuário direto do aplicativo, mas terá de ser cadastrado se estiver internado e utilizando algum leito hospitalar. Todos os casos de uso pertencem apenas ao cliente. O usuário final não interage com o servidor. O diagrama de casos de uso é apresentado na Figura 7.

Figura 7 – Diagrama de casos de uso



Fonte: Elaborado pelo autor.

A seguir é demonstrado em subseções os casos de usos deste trabalho assim como suas especificações e detalhamentos. A seção 3.2.3.1 apresenta o caso de uso `Manter Beacons`. A subseção 3.2.3.2 pode-se visualizar o caso de uso `Manter Leitos`. A seção 3.2.3.3 apresenta o caso de uso Cadastrar Alergias. A seção 3.2.3.4 apresenta o caso de uso Alerta de alergias ao aproximar. Por fim a seção 3.2.3.5 apresenta o caso de uso Consultar alergias.

3.2.3.1 UC06 Manter Beacons.

Este caso de uso é responsável por manter cadastrado os dispositivos *beacons* e suas informações. No Quadro 3 é apresentado a descrição detalhada do caso de uso `Manter Beacons`.

Quadro 3 – Descrição detalhada do caso de uso manter *beacons* (UC06)

UC06	A aplicação deve permitir manter *beacons*.
Descrição	Permite aos usuários realizarem o cadastro de *beacons* para que possam vincular ao leito.
Ator	Administrador.
Pré-condição	Administrador autenticado no sistema; Bluetooth ativo; Acesso à internet.
Fluxo Principal	1. Administrador acessa o aplicativo; 2. Administrador acessa o menu Beacon; 3. Administrador clicar no botão adicionar; 4. Administrador informa a descrição do *beacon* e foto; 5. Administrador faz o scan do dispositivo *beacon*s próximos para obter seu UUID; 6. Administrador clicar em salvar.
Cenário Consulta	1. Realizar os passos 1 e 2 do fluxo principal; 2. Sistema apresenta lista de todos os *beacon*s.
Cenário Edição	1. Realizar passos 1 e 2 do fluxo principal; 2. Selecionar um *beacon* existente; 3. Sistema posiciona o *beacon* para edição; 4. Administrador realiza as alterações; 5. Administrador clicar em salvar.
Cenário Exclusão	1. Realizar passos 1 e 2 do fluxo principal; 2. Clicar no botão Del sobre o *beacon* da lista; 3. Sistema pergunta se deseja excluir o registro; 4. Administrador clicar em Sim; 5. Sistema exclui o registro.
Pós-condição	Administrador visualizou, alterou, incluiu ou exclui um *beacon*.

Fonte: Elaborado pelo autor.

Como detalhado no Quadro 3 o caso de uso possui um fluxo principal que se inicia com *login* do ator `administrador`. O `administrador` visualiza na tela de menus e acessa o cadastro de *beacon*. O `administrador` clica em adicionar e informa uma descrição ao registro identificando o *beacon*. Em seguida o `administrador` pode acionar a câmera através de um botão para captura de uma foto do dispositivo para armazenamento, porém essa

informação não é obrigatória. A próxima etapa é efetuar o escaneamento do UUID do dispositivo e preencher o campo UUID para finalizar o fluxo principal de inserção.

O caso de uso possui cenários de consulta que é acionado ao entrar no cadastro de *beacon*, pois é apresentada uma lista com os *beacons* já existentes. O cenário de edição pode ser acessado ao selecionar um item da consulta. A aplicação direciona para a tela de cadastro para atualização das informações do registro correspondente pelo `administrador`.

O cenário de exclusão é acionado pelo botão de descrição `Del` em todos os registros apresentados pela consulta. Assim que o botão é selecionado a aplicação questiona se o administrador deseja confirmar a exclusão.

3.2.3.2 UC02 Manter Leitos

O caso de uso `Manter Leitos` é onde pode–se armazenar as informações de um leito hospitalar na aplicação.

No Quadro 4 é apresentado a descrição detalhada do caso de uso `Manter Leitos`.

Quadro 4 – Descrição detalhada do caso de uso Manter Leitos (UC02)

UC02	O aplicativo deve permitir manter Leitos.
Descrição	Permite aos usuários realizarem o cadastro de leitos bem como vincular um paciente ou um dispositivo *beacon.*
Ator	Administrador.
Pré-condição	Administrador autenticado no sistema; Acesso à internet.
Fluxo Principal	1. Administrador acessa o aplicativo; 2. Administrador acessa o menu Leitos; 3. Administrador clicar no botão adicionar; 4. Administrador informa a descrição do leito, vincula paciente atual, vincula o *beacon*; 5. Administrador clicar em salvar.
Cenário Consulta	1. Realizar os passos 1 e 2 do fluxo principal; 2. Sistema apresenta lista de todos os leitos.
Cenário Edição	1. Realizar passos 1 e 2 do fluxo principal; 2. Selecionar um leito existente; 3. Sistema posiciona o leito para edição; 4. Administrador realiza as alterações;

	5. Administrador clicar em salvar.
Cenário Exclusão	1. Realizar passos 1 e 2 do fluxo principal; 2. Clicar no botão Del sobre o leito da lista; 3. Sistema pergunta se deseja excluir o registro; 4. Administrador clicar em Sim; 5. Sistema exclui o registro.
Pós-condição	Administrador visualizou, alterou, incluiu ou exclui um leito.

Fonte: Elaborado pelo autor.

O caso de uso tem um fluxo principal que se inicia com o `administrador` fazendo *login* na aplicação. Em seguida o `administrador` acessa o menu leito e clica no botão adicionar. A aplicação direciona para a tela de cadastro de leito para que o administrador preencha as informações correspondente ao leito hospitalar e clique em salvar. Antes da ação é possível vincular um paciente e um *beacon* ao leito. Ao clicar em salvar a aplicação consiste se as informações de *beacon* ou paciente são verdadeiras.

O caso de uso também possui cenário de consulta, alteração e exclusão. A consulta é acionada ao entrar no cadastro de *beacon* onde a aplicação monta uma lista com os registros existentes e seus respectivos nomes e foto, se existir. O cenário de edição é acionado ao selecionar algum registro da lista montada pelo cenário de consulta. O `administrador` realiza as alterações necessárias e salva o registro. Já o cenário de exclusão é acionado pelo botão com a descrição `Del` que está presente em todos os itens da lista de consulta. Ao clicar no botão a aplicação questiona se o registro deve ser excluído através de uma mensagem de confirmação.

3.2.3.3 UC04 Cadastrar Alergias

O caso de uso cadastrar alergias permite a inclusão da informação alérgica medicamentosa do paciente. No Quadro 5 é apresentado a descrição detalhada do caso de uso `Cadastrar Alergias`.

Quadro 5 – Descrição detalhada do caso de uso Cadastrar Alergias

UC04	O aplicativo deve permitir manter Alergias.
Descrição	Permite aos usuários realizarem o cadastro de Alergias bem como vincular um paciente, um medicamento e um tipo de reação.
Ator	Administrador.
Pré-condição	Administrador autenticado no sistema; Acesso à internet;

	Existir ao menos um paciente previamente cadastrado; Existir ao menos um medicamento previamente cadastrado; Existir ao menos um tipo de reação alérgica previamente cadastrada.
Fluxo Principal	1. Administrador acessa o aplicativo; 2. Administrador acessa o menu Alergia; 3. Administrador informa a descrição da alergia, vincula paciente, vincula um medicamento, seleciona um tipo de reação, digita alguma observação; 4. Administrador clicar em salvar.
Pós-condição	Administrador incluiu uma alergia.

Fonte: Elaborado pelo autor.

Inicialmente o ator `administrador` faz o *login* na aplicação. Em seguida acessa o menu `Alergia`. A aplicação posiciona na tela de cadastro da alergia, permitindo que seja informado o paciente, medicamento, tipo de reação, descrição da alergia e uma breve observação sobre a mesma. O `administrador` preenche as informações necessárias e clica em salvar. A aplicação válida os dados e emite uma mensagem dizendo que a alergia foi cadastrada com sucesso. Os campos obrigatórios são paciente, medicamento, descrição, e o tipo de reação alérgica. A aplicação não permite salvar caso algum desses dados não for preenchido emitindo uma mensagem alertando a falta do mesmo.

3.2.3.4 UC07 Alerta de alergias ao aproximar

O caso de uso alerta de alergias ao aproximar é o caso de uso de maior importância da aplicação e está atrelado aos atores usuário e administrador. No Quadro 6 é apresentado a descrição detalhada do caso de uso `Alerta de alergias ao aproximar`.

Quadro 6 – Descrição detalhada do caso de uso Alerta de alergias ao aproximar

UC07	O aplicativo deve permitir emitir o alerta de alergias ao usuário ao se aproximar.
Descrição	Permite aos usuários a visualizar um alerta na lista ao se aproximar do leito do paciente. Para isso o leito deve estar vinculado corretamente a um dispositivo *beacon*.
Ator	Administrador, Usuário.
Pré-condição	Usuário/Administrador autenticado no sistema; Paciente estar vinculado ao leito; Beacon estar vinculado ao leito; Bluetooth ativo; Acesso à internet.

Fluxo Principal	1. Usuário acessa o aplicativo; 2. Usuário acessa o menu Alerta; 3. Usuário posiciona o dispositivo móvel a menos de 1 metro do leito do paciente; 4. Aplicação lista o nome do paciente mais a descrição do leito no item da lista.
Pós-condição	Usuário visualizou o alerta.

Fonte: Elaborado pelo autor.

O fluxo principal começa após o `usuário` efetuar o *login* da aplicação. Em seguida o `usuário` acessa a tela de alerta. O `usuário` deve posicionar seu dispositivo móvel com a aplicação aberta a menos de um metro de distância do leito cadastrado. A aplicação válida se o *beacon* atrelado ao leito está cadastrado corretamente. Aplicação emite o alerta adicionando o nome do paciente do leito e a descrição na lista de alertas.

3.2.3.5 UC08 Consultar Alergias

O caso de uso consultar alergias permite a visualização da alergia medicamentosa do paciente através do alerta emitido pelo UC07. No Quadro 7 é apresentado a descrição detalhada do caso de uso `Consultar Alergias`.

Quadro 7 – Descrição detalhada do caso de uso consultar alergias

UC08	O aplicativo deve permitir emitir consultar as alergias após selecionar um alerta.
Descrição	Permite aos usuários acessar a informação alérgica a medicamentos do paciente ao selecionar um alerta do UC07.
Ator	Administrador, Usuário.
Pré-condição	Usuário/Administrador autenticado no sistema; Paciente deve estar vinculado ao leito; Beacon deve estar vinculado ao leito; Paciente deve possuir alergias cadastradas; Bluetooth ativo; Acesso à internet.
Fluxo Principal	1. Realizar os passos do Fluxo principal UC07 do 1 a 4; 2. Usuário seleciona o item da lista; 3. Sistema apresenta as informações do paciente, nome, idade, medidas e lista de alergias medicamentosas.
Fluxo	1. Realizar os passos 1 e 2 do fluxo principal; 2. Sistema verifica que o paciente não possui alergias cadastradas.

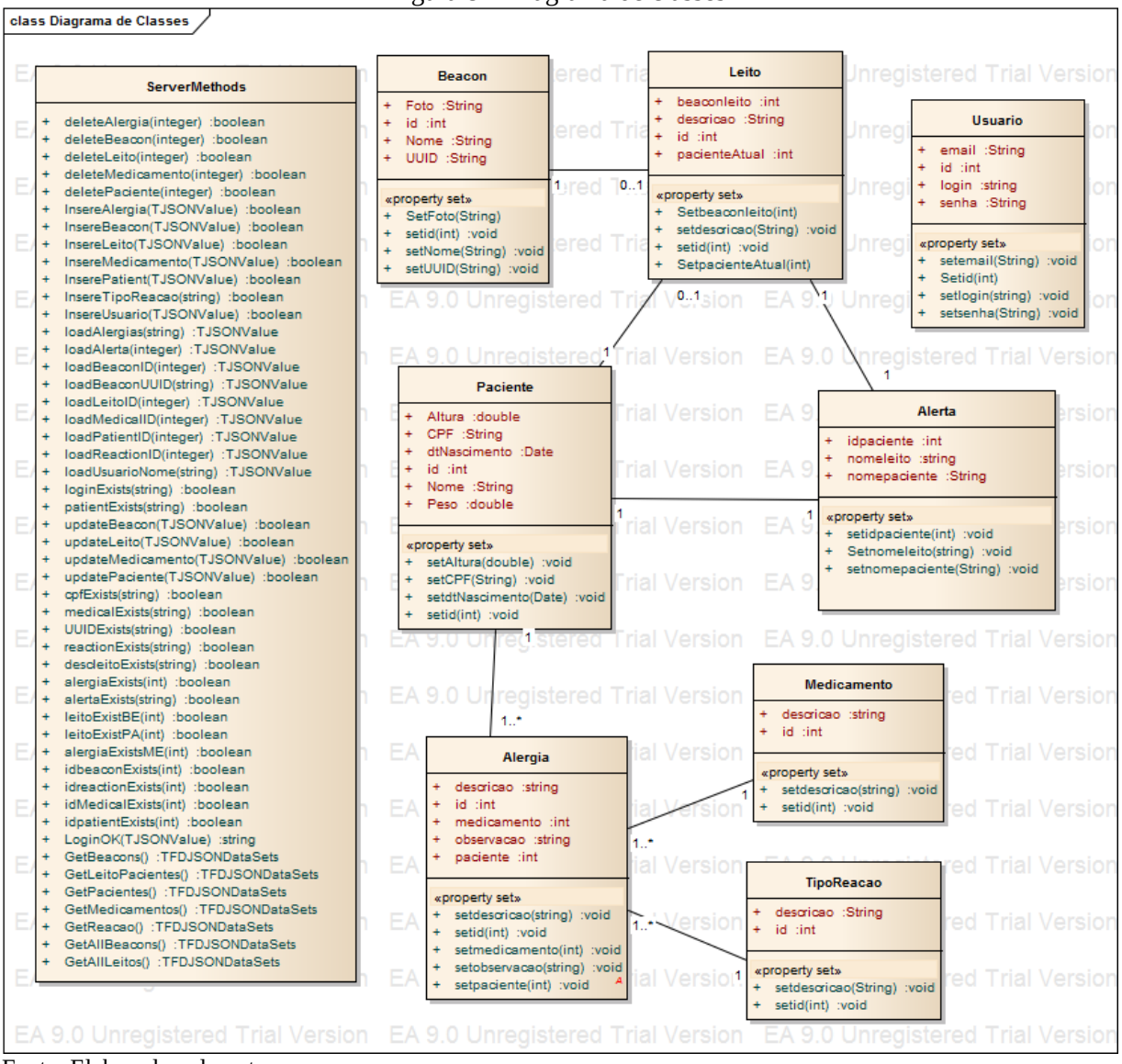
Alternativo	3. Sistema emite uma mensagem alertando que o paciente não possui alergias medicamentosas cadastradas.
Pós-condição	Usuário visualizou as informações alérgicas através do alerta.

Fonte: Elaborado pelo autor.

3.2.4 DIAGRAMA DE CLASSES

A Figura 8 apresenta o diagrama de classes do aplicativo, pode-se ter um melhor entendimento estrutural das classes. As classes foram criadas para melhor manuseio das informações carregadas da base de dados do Servidor.

Figura 8 – Diagrama de classes



Fonte: Elaborado pelo autor.

A classe `Beacon` possui atributos que identificam um dispositivo BLE *beacon*. Já a classe `Usuario` possui atributos que identificam um usuário na aplicação. A classe

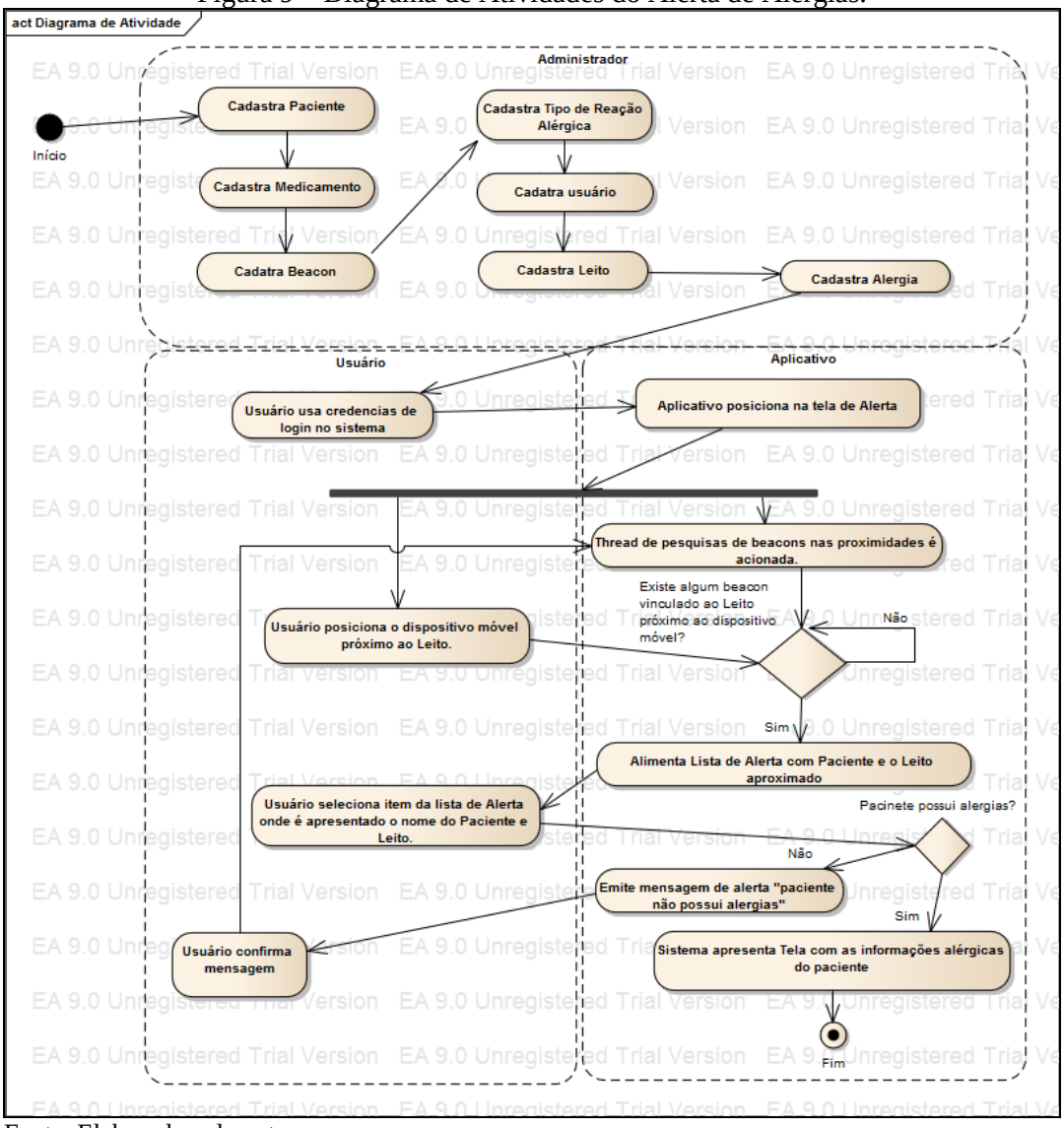
`Medicamento` representa um medicamento da aplicação. Já a classe `TipoReacao` representa as informações que identificam uma reação alérgica. A classe `Paciente` representa um paciente possível de ser utilizado. A classe `Leito` possui atributos que identificam um leito hospitalar. A classe `Alergia` possui atributos para manipular os dados da alergia medicamentosa do paciente. A classe `Alerta` foi criada com a intenção de manipular dados mais rapidamente, pois a mesma é utilizada quando é encontrado algum *beacon* próximo. Ela ajuda a trazer a informação caso o *beacon* deva ser considerado como alerta ou não. Pode-se perceber que seus atributos são os mesmo de outras classes, como o `idpaciente`, e `nomepaciente` que pode ser encontrado na classe `Paciente`.

A classe `ServerMethods` é onde é possível encontrar todos os métodos de validação e inserção, leitura, atualização e exclusão (CRUD), e manuseio do servidor com o banco de dados. Todos os métodos criados nesta classe são espelhados pelo servidor DataSnap na parte do cliente de forma que fica disponível para ser utilizado em ambas as partes. A geração desse espelhamento é feita de forma automática pela tecnologia DataSnap.

3.2.5 DIAGRAMA DE ATIVIDADES

Nesta seção é apresentado o diagrama de atividades do aplicativo, com o fluxo principal de funcionamento do aplicativo. É válido afirmar que quando a *thread* que procura por dispositivos do tipo *beacon* está ativa, ela permanece nesse estado até que o aplicativo seja encerrado. Na Figura 9 é possível visualizar o diagrama descrito neste parágrafo.

Figura 9 – Diagrama de Atividades do Alerta de Alergias.



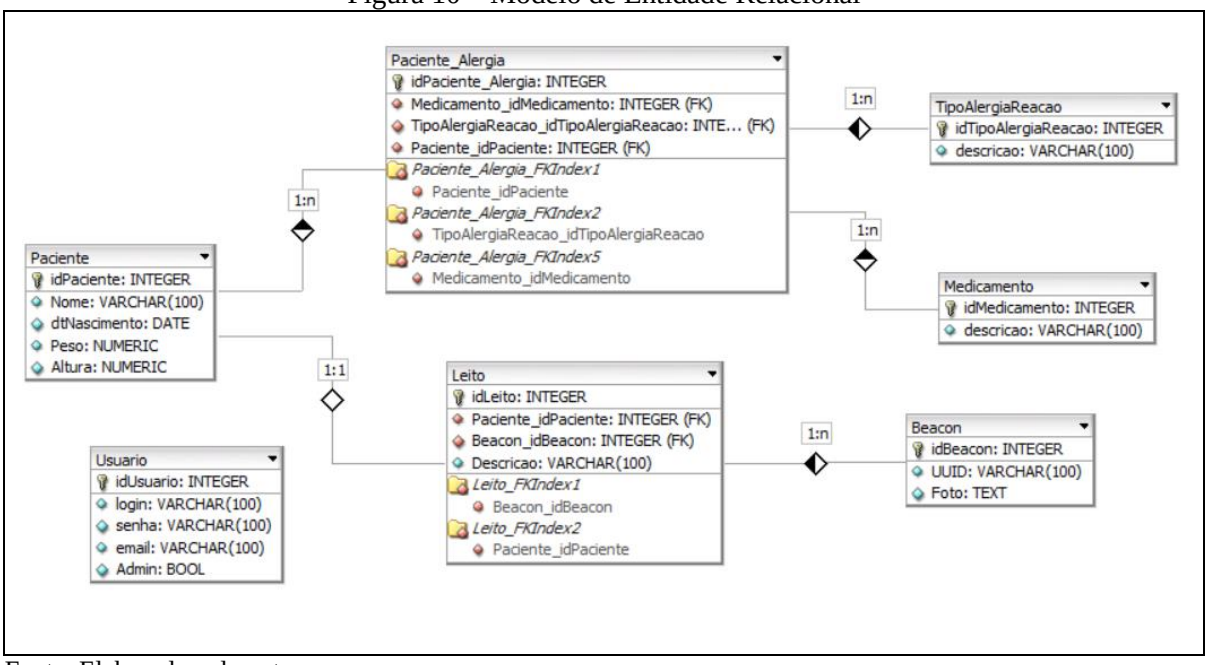
Fonte: Elaborado pelo autor.

Primeiramente o `administrador` alimenta base de dados cadastrando paciente, medicamento, *beacons*, tipo de reação alérgica, usuário, leito e as alergias dos pacientes. O usuário efetua *login* na aplicação que posiciona diretamente na tela de alerta. A aplicação inicia escaneamento de leitos vinculados por *beacons.* Com a interação física o usuário se aproxima do leito previamente cadastrado. A aplicação encontra o leito através do *beacon* e alimenta a lista de alertas com o nome do paciente e a descrição do leito. O `usuário` por sua vez visualiza a lista e clica no alerta. Aplicação válida se o usuário possui alergias a medicamentos cadastradas. Se a resposta for sim, o sistema apresenta as informações alérgicas. Se a resposta for não, o sistema emite uma mensagem alertando o usuário de que o paciente não possui informações alérgicas cadastradas e reinicia a *thread.*

3.2.6 MODELO DE ENTIDADE RELACIONAL

A Figura 10 demonstra o Modelo de Entidade Relacional (MER) onde é possível visualizar as tabelas, campos e seus relacionamentos utilizados no aplicativo. A informações detalhadas sobre o dicionário de dados podem ser visualizadas no apêndice A deste trabalho.

Figura 10 – Modelo de Entidade Relacional



Fonte: Elaborado pelo autor.

As tabelas foram criadas no banco de dados MySQL, que é a base de dados utilizada pelo aplicativo no servidor. A tabela `Paciente` apresenta as propriedades do paciente. Somente é possível um usuário administrador adicionar novos pacientes. A tabela `Usuario` apresenta as propriedades de usuário. Nela é feita a autenticação para entrar no aplicativo. O campo `Admin` indica se o usuário cadastrado é administrador ou não. Qualquer usuário pode se cadastrar no aplicativo, porém apenas um administrador pode cadastrar outro administrador.

A tabela `Beacon` armazena as propriedades de um dispositivo *beacon*. Nela é armazenado seu UUID como identificação. Apesar do nome da coluna ser UUID o sistema concatena a informação do *major* e *minor* do dispositivo ao salvar um registro.

A tabela `Leito` armazena as propriedades do leito. Também é de uso exclusivo dos administradores. Um registro na tabela `Leito` indica um leito hospitalar, nele é vinculado um registro da tabela `Beacon` e `Paciente`.

A tabela `Medicamento` é onde é armazenado a lista de descrições dos medicamentos do aplicativo. Os medicamentos posteriormente podem ser vinculados as alergias dos pacientes.

Já a tabela `Paciente_Alergia` mescla as informações de alergia a medicamentos, paciente e reação alérgica. Faz referências as tabelas `Paciente`, `Medicamentos` e `TipoAlergiaReacao`.

A tabela `TipoAlergiaReacao` é onde são armazenadas as possíveis reações alérgica de algum medicamento vinculado à tabela `Paciente_Alergia`. A tabela `TipoAlergiaReacao` serve apenas de apoio a decisão do profissional de saúde.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as ferramentas e componentes utilizados para desenvolvimento da aplicação. Em seguida são apresentados trechos de códigos fontes relevantes, e por fim, é detalhado a operacionalidade da aplicação.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para o desenvolvimento da aplicação foi utilizada a IDE Delphi 10 Seattle da empresa Embarcadero que utiliza linguagem Object Pascal. A escolha da IDE está relacionada a funcionalidades para com componentes *Bluetooth Low Energy* (BLE) como é o caso dos dispositivos *beacons*. Também foi levado em conta o desenvolvimento de aplicações móveis com a ferramenta Delphi 10 Seattle onde a interação com componentes visuais e programação tornam um ambiente produtivo como nas versões para desktop.

A ferramenta Delphi 10 Seattle, como pode ser visto na seção 2.3, possui componentes que facilitam o desenvolvimento com dispositivos *beacons*. Para realizar a comunicação entre o *beacon* e a aplicação foi utilizado o componente `TBluetoothLEDevice` pertencente a classe nativa da IDE `System.Bluetooth`. Esta classe interage diretamente com o drive Bluetooth do dispositivo móvel indicando a necessidade da ativação da opção de Bluetooth do dispositivo.

3.3.1.1 DESENVOLVIMENTO DA APLICAÇÃO

O componente `TBluetoothLEDevice`, de acordo com a Embarcadero (2014), representa um dispositivo remoto que suporta *Bluetooth Low Energy* (BLE). É possível descobrir os serviços que o dispositivo remoto fornece e obter informações detalhadas sobre eles, incluindo suas características. Os *beacons* interagem com dispositivos móveis através de BLE, e devido a isso o uso do componente `TBluetoothLEDevice` facilitou o desenvolvimento.

Para a identificação dos *beacons* desenvolveu-se uma *thread* para varrer os dispositivos *beacons* nas proximidades do dispositivo móvel. Esta varredura atualiza constantemente uma lista com dispositivos BLE que sejam do tipo *beacon*. A lista também atualiza as informações detalhadas dos dispositivos, pois conforme o usuário movimenta o dispositivo móvel a distância em que se encontra o *beacon* também é atualizada. Para o desenvolvimento da aplicação foi definido que o usuário deva estar a no mínimo um metro de distância do dispositivo para que o aplicativo reconheça sua existência. No Quadro 8 é possível visualizar um trecho do código da *thread* correspondente ao escaneamento dos *beacons*.

Quadro 8 - Trecho do código da *thread* de escaneamento

```
1075      LBeaconDevice := DecodeScanResponse;
1076      NewBeacon := 0;
1077      if BeaconDeviceList.Count > 0 then
1078      begin
1079        for I := 0 to BeaconDeviceList.count-1 do
1080          if ((BeaconDeviceList[I].GUID = LBeaconDevice.GUID) and (BeaconDeviceList[I].Major = LBeaconDevice.Major)
1081            and (BeaconDeviceList[I].Minor = LBeaconDevice.Minor)) then
1082          begin
1083            BeaconDeviceList[I] := LBeaconDevice;
1084            NewBeacon := I+1;
1085            Break;
1086          end;
1087      end;
1088      TThread.Synchronize(nil, procedure
1089      begin
1090        if (NewBeacon = 0) then
1091        begin
1092          BeaconDeviceList.Add(LBeaconDevice);
1093          if (LBeaconDevice.Distance <= 1) then // Distância menor que 1 metro
1094          begin
1095            UUID :=  LBeaconDevice.GUID.ToString;
1096            UUID :=  UUID.Replace('{','');
1097            UUID :=  UUID.Replace('}','');
1098            UUID := UUID+':'+LBeaconDevice.Major.ToString+':'+LBeaconDevice.Minor.ToString;
1099            addBeaconAlerta(UUID);
1100           end;
1101        end
1102        else
1103        begin
1104          UUID :=  LBeaconDevice.GUID.ToString;
1105          UUID :=  UUID.Replace('{','');
1106          UUID :=  UUID.Replace('}','');
1107          UUID := UUID+':'+LBeaconDevice.Major.ToString+':'+LBeaconDevice.Minor.ToString;
1108          if LBeaconDevice.Distance <= 1 then // Distância menor que 1 metro.
1109          begin
1110            if not JahExisteNaLista(UUID) then
1111              addBeaconAlerta(UUID);
1112          end
1113          else
1114          begin
1115            if JahExisteNaLista(UUID) then
1116              RemoveBeaconAlerta(UUID);
1117          end;
```

Fonte: Elaborado pelo autor.

A função `DecodeScanResponde`, apresentada na `linha um mil e setenta e cinco`, retorna um objeto do tipo `TBeaconDevice` onde é comparado à uma lista desse tipo de componente para informar se o dispositivo já foi encontrado previamente. Logo em seguida a variável `NewBeacon` do tipo inteiro, apresentada na `linha um mil e noventa`, identifica se há um novo *beacon* ou se já foi mapeado. No caso de ser um novo *beacon* o sistema adiciona o mesmo a lista `BeacondeviceList` e válida se sua distância do dispositivo móvel é menor ou

igual a um metro para criar o alerta. Em *beacon*s existentes na lista o sistema apenas válida a distância e se já não foi adicionado no alerta.

Caso o usuário selecione um alerta o sistema aciona o método `getAlergias` passando por parâmetro o `id` do paciente. O Quadro 9 pode-se visualizar o trecho de código que demonstra esta ação.

Quadro 9 – Consistência se paciente possui alergias

```
1218  procedure TFormAllergy.getAlergias(idP : integer);
1219  var
1220    ListAlergia : TListAlergia;
1221    Alergia     : TAlergia;
1222    Item        : TMetropolisUIListBoxItem;
1223    Paciente    : TPaciente;
1224    i,total     : Integer;
1225  begin
1226    if ClientModule1.ServerMethods1Client.alergiaExists(idP) then
1227    begin
1228      total := lbInfoAlergicas.Items.Count;
1229      for i := total-1 downto 0 do
1230      begin
1231        if lbInfoAlergicas.ItemByIndex(I) is TMetropolisUIListBoxItem then
1232        begin
1233          lbInfoAlergicas.RemoveObject(lbInfoAlergicas.ItemByIndex(I));
1234        end;
1235      end;
1236      Paciente :=  JSONtoPatient(ClientModule1.ServerMethods1Client.loadPatientID(idP));
1237      ListAlergia :=  JSONtoLAlergia( ClientModule1.ServerMethods1Client.loadAlergias(idP) );
1238      lbiInfoNome.ItemData.Text    := Paciente.nome;
1239      lbiInfoIdade.ItemData.Text   := Age(Paciente.dtNascimento).ToString+' anos';
1240      lbiInfoMedidas.ItemData.Text := 'Altura '+Paciente.altura.ToString()+ 'm Peso '+Paciente.peso.ToString()+'kg'
1241      try
1242        lbInfoAlergicas.BeginUpdate;
1243        for Alergia in ListAlergia.ListAlergia do
1244        begin
1245          Item :=  TMetropolisUIListBoxItem.Create(Self);
1246          Item.Height := 130;
1247          Item.TextSettings.WordWrap := True;
1248          Item.Title :=  'Alergia : '+Alergia.descricao;
1249          Item.SubTitle := 'Medicamento : '+
1250          JSONtoMedical(ClientModule1.ServerMethods1Client.loadMedicalID(Alergia.medicamento)).descricao;
1251          Item.Description := 'Reação : '+
1252          JSONtoReaction(ClientModule1.ServerMethods1Client.loadReactionID(Alergia.reacao)).desc+#13+
1253          Alergia.observacao;
1254          lbInfoAlergicas.InsertObject(6, Item);
1255        end;
1256        lbInfoAlergicas.EndUpdate;
1257      finally
1258        if Assigned(ListAlergia) then
1259          ListAlergia.Free;
1260      end;
1261      ChangeTabAlergiaPaciente.ExecuteTarget(Self);
```

Fonte: Elaborado pelo autor.

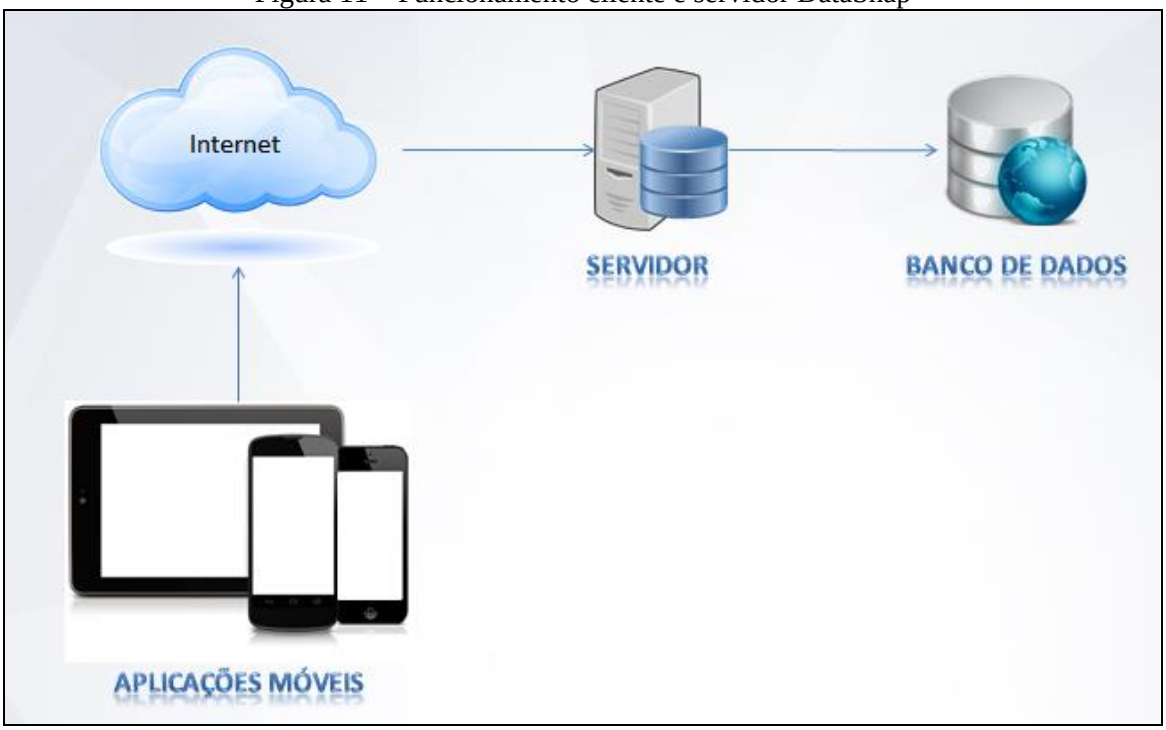
O método `getAlergias` é acionado quando selecionado um item da tela de alertas. Pode-se verificar que o servidor é chamado no método `alergiaExist`, na `linha um mil duzentos e vinte seis`, passando por parâmetro o atributo `id` do paciente selecionado. Caso este método retorne verdadeiro o método busca a lista de alergias do paciente no método `loadAlergias`, apresentado na `linha um mil duzentos e trinta e sete`, e retorna um componente interno do tipo `TListaAlergia`. Esta lista é uma coleção de componentes do tipo `TAlergia`. Logo em seguida, na `linha um mil duzentos e quarenta e três`, pode–se visualizar a instrução `for` passando por todos os itens da `ListaAlergia` alimentando assim um componente visual do tipo `TListViewItem`.

3.3.1.2 CLIENTE E SERVIDOR

Para o desenvolvimento das consistências de dados remotos do aplicativo foi utilizado o *framework* DataSnap disponível pela IDE Delphi 10 Seattle. Este *framework* facilita a criação de um ambiente cliente e servidor. Esta comunicação é feita através de uma classe de servidor de métodos criada automaticamente pela ferramenta.

Na Figura 11 pode-se visualizar o funcionamento da comunicação de um dispositivo móvel com o servidor DataSnap.

Figura 11 – Funcionamento cliente e servidor DataSnap



Fonte: Santos (2015).

Segundo Embarcadero (2014), o DataSnap fornece uma maneira do cliente se comunicar com segurança com o servidor, usando transferência *JavaScript Object Notation* (JSON) sobre os protocolos TCP/ IP ou HTTP. Dessa forma foi necessário compartilhar as classes de objetos no projeto do aplicativo, que é o cliente, e no projeto de servidor. O compartilhamento foi necessário para facilitar a troca de informação entre objetos do cliente servidor. O DataSnap não permite que a comunicação seja feita em objetos nativos do Delphi, como por exemplo `TUsuario`, da classe `Usuario`. Para isso é necessário converter uma classe nativa do Delphi em JSON para enviar ao cliente ou vice-versa. No Quadro 10 é apresentado o trecho de código onde é possível visualizar os métodos utilizados para conversão em JSON.

Quadro 10 – Trecho de código de conversão de classe usuário para JSON

```
39  function UsuarioToJSON(usuario : TUsuario): TJSONValue;
40  var
41    m: TJSONMarshal;
42  begin
43    if Assigned(usuario) then
44    begin
45      m := TJSONMarshal.Create(TJSONConverter.Create);
46      try
47        Exit(m.Marshal(usuario));
48      finally
49        m.Free;
50      end;
51    end
52    else
53      Exit(TJSONNull.Create);
54  end;
55
56
57  function JSONtoUsuario(Json : TJSONValue): TUsuario;
58  var
59    unm: TJSONUnMarshal;
60  begin
61    if Json is TJSONNull then
62      Exit(nil)
63    else
64    begin
65      unm := TJSONUnMarshal.Create;
66      try
67        Exit(unm.Unmarshal(Json) as TUsuario);
68      finally
69        unm.Free;
70      end;
71    end;
72  end;
```

Fonte: Elaborado pelo autor.

Os métodos `UsuarioToJSON` da `linha trinta e nove`, e o método `JSONtoUsuario` da `linha cinquenta e sete`, se repetiram perante as demais classes alterando apenas a classe nativa do objeto. A conversão facilitou a manipulação de dados em forma de objetos nos dois lados da aplicação (cliente e servidor). Desta forma foi possível adotar a prática de orientação a objeto. Na `linha quarenta e sete`, a função `Marshal` do componente do tipo `TJSONMarshal` transforma um objeto do tipo `TUsuario`, passado por parâmetro, em um `TJSONvalue`. Já na `linha sessenta e sete`, é possível notar que a função `Unmarshal` do componente `TJSONUnMarshal` faz o contrário, retorna um objeto do tipo `TUsuario` a partir de um objeto `TJSONValue` passado por parâmetro.

A conexão com banco de dados do servidor foi feita a partir dos componentes nativos da IDE chamados `FireDac`. Segundo a fabricante Embarcadero (2016), o `FireDac` é uma camada de acesso poderosa, mas fácil de usar, que suporta, abstrai e simplifica o acesso a dados, fornecendo todos os recursos necessários para construir aplicativos reais de alta carga. Em algumas situações no aplicativo fez-se necessário trazer informações para o administrador em forma de consultas. Nessas ocasiões utilizou-se recursos do Delphi para transformar os

dados de consultas em JSON e envia-los ao cliente. O Quadro 11 apresenta trecho de código utilizado para retornar as consultas no banco de dados do lado do servidor e transformá-las em objetos do tipo `TFDJSONDataSet`.

Quadro 11 – Trecho código te conversão de *querys* em `TFDJSONDatasets`

```
487   function TServerMethods1.GetAllBeacons: TFDJSONDataSets;
488   begin
489     FDAllBeacons.Active := false;
490
491     Result :=  TFDJSONDataSets.Create;
492
493     TFDJSONDataSetsWriter.ListAdd(Result, FDAllBeacons);
494   end;
495
496   function TServerMethods1.GetAllLeitos: TFDJSONDataSets;
497   begin
498     FDAllLeitos.Active := false;
499
500     Result :=  TFDJSONDataSets.Create;
501
502     TFDJSONDataSetsWriter.ListAdd(Result, FDAllLeitos);
503   end;
504
505   function TServerMethods1.GetBeacons: TFDJSONDataSets;
506   begin
507     FDBeacon.Active := false;
508
509     Result :=  TFDJSONDataSets.Create;
510
511     TFDJSONDataSetsWriter.ListAdd(Result, FDBeacon);
512   end;
513
514   function TServerMethods1.GetLeitoPacientes: TFDJSONDataSets;
515   begin
516     FDLeitoPaciente.Active := false;
517
518     Result :=  TFDJSONDataSets.Create;
519
520     TFDJSONDataSetsWriter.ListAdd(Result, FDLeitoPaciente);
521   end;
522
523   function TServerMethods1.GetMedicamentos: TFDJSONDataSets;
524   begin
525     FDMedicamento.Active := false;
526
527     Result :=  TFDJSONDataSets.Create;
528
529     TFDJSONDataSetsWriter.ListAdd(Result, FDMedicamento);
530   end;
```

Fonte: Elaborado pelo autor.

Os componentes `FDAllBeacons`, `FDAllLeitos`, `FDBeacon`, `FDLeitoPaciente` e `FDMedicamento` são do tipo `TFDQuery`. Este tipo de componente, disponível nos componentes `FireDac`, é capaz de executar consultas SQL no banco de dados. Pode–se observar o método `GetAllBeacons` na `linha quatrocentos e noventa e três`, onde o método `ListAdd` nativo do componente `TFDJSONDataSetsWrite` transcreve todo o resultado obtido pelo

componente `FDAllBeacons` no resultado da função. No Quadro 12 pode-se visualizar a chamada do método `GetAllBeacons` do servidor, pelo método `getAllBeacons` do cliente.

Quadro 12 - Chamada método `GetAllBeacons` no lado do cliente

```
1267  procedure TFormAllergy.getAllBeacons;
1268  var
1269    LDataSetList : TFDJSONDataSets;
1270    Item : TListViewItem;
1271    ms : TMemoryStream;
1272    base64 : TBase64;
1273  begin
1274    ClientModule1.FDMemAllBeacons.Close;
1275    LDataSetList := ClientModule1.ServerMethods1Client.GetAllBeacons();
1276    ClientModule1.FDMemAllBeacons.AppendData(TFDJSONDataSetsReader.GetListValue(LDataSetList, 0));
1277    ClientModule1.FDMemAllBeacons.Open;
1278    ClientModule1.FDMemAllBeacons.First;
1279    while not ClientModule1.FDMemAllBeacons.Eof do
1280    begin
1281      ListViewBeacon.BeginUpdate;
1282      Item :=  ListViewBeacon.Items.Add;
1283      Item.Height := 70;
1284      Item.Text   := ClientModule1.FDMemAllBeacons.FieldByName('nome').asString;
1285      Item.Detail := ClientModule1.FDMemAllBeacons.FieldByName('UUID').asString;
1286      Item.Tag    := ClientModule1.FDMemAllBeacons.FieldByName('idBeacon').AsInteger;
1287      if ClientModule1.FDMemAllBeacons.FieldByName('Foto').AsWideString <> '' then
1288      begin
1289        ms := TMemoryStream.Create;
1290        base64 := TBase64.Create;
1291        ms.LoadFromStream( base64.decodeToStream(ClientModule1.FDMemAllBeacons.FieldByName('Foto').AsWideString));
1292        Item.Bitmap.LoadFromStream(ms);
1293      end;
1294      ListViewBeacon.EndUpdate;
1295      ClientModule1.FDMemAllBeacons.Next;
1296    end;
1297  end;
```

Fonte: Elaborado pelo autor.

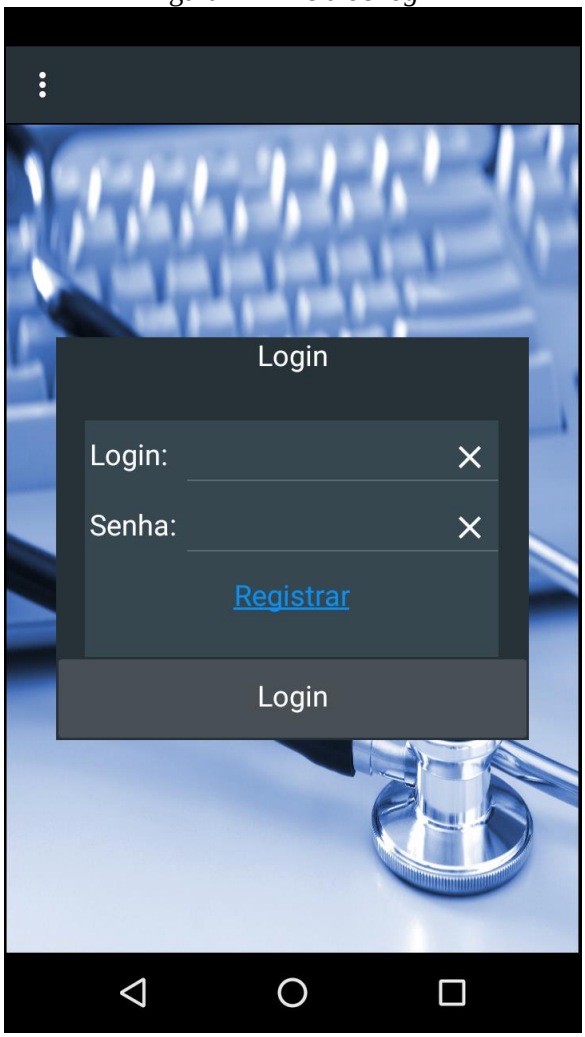
Este trecho de código representa a forma como a aplicação captura dados do servidor para montar a tela de consulta de *beacons*. A montagem é feita previamente ao acesso da tela.

O componente `FDMemAllBeacons`, apresentado na `linha um mil duzentos e setenta e quatro`, é do tipo `TFDMemTable` também nativo do pacote de componentes FireDac da IDE. Com ele é possível manipular dados em memória sem realizar uma conexão direta com banco de dados. Na `linha um mil duzentos e setenta e cinco` pode-se visualizar a chamada do método `GetAllBeacons` do servidor, alimentando o componente `LDataSetList` do tipo `TFDJSONDataSets`. Na `linha` seguinte ocorre a conversão inversa ao que aconteceu no lado do servidor. Os dados contidos no JSON são inseridos no componente `FDMemAllBeacons` para que possam ser manipulados como se estivesse utilizando uma *query* no lado do cliente. Da `linha um mil duzentos e setenta e nove` até `um mil duzentos e noventa e sete` é percorrido todos os dados dessa consulta e adicionado um componente do tipo `TListViewItem` para cada registro *beacon* retornado da consulta. Por fim é apresentada a tela de consultas de *beacons*.

3.3.2 OPERACIONALIDADE DA APLICAÇÃO

A aplicação inicialmente apresenta a tela de *login* para o usuário entrar no sistema. O campo `login` e o campo `senha` são sensíveis a letras maiúsculas e minúsculas. A senha é armazenada no banco de dados criptografada. A Figura 12 exibe a tela de *login* da aplicação.

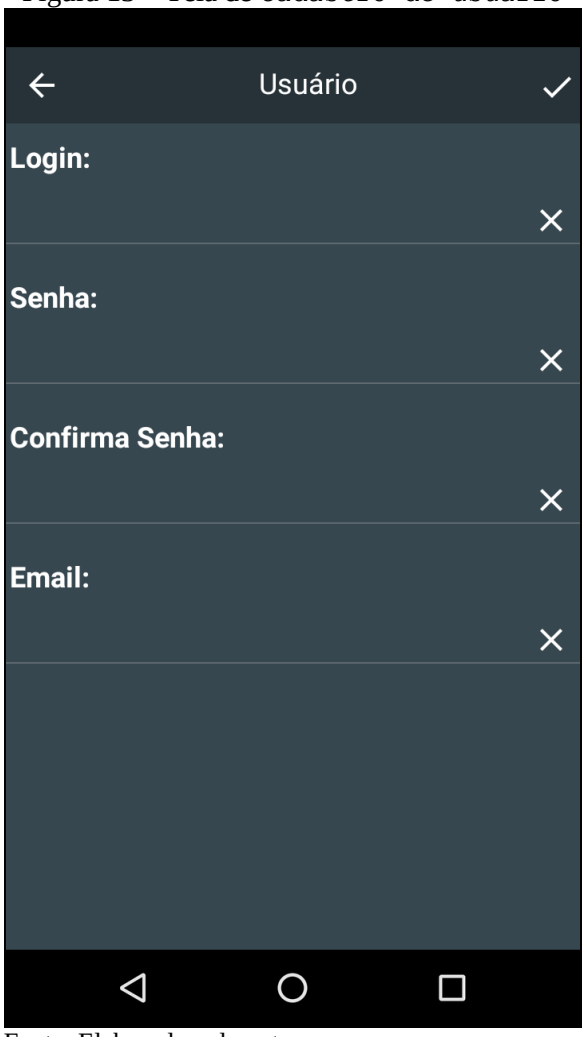
Figura 12 – Tela de *login*



Fonte: Elaborado pelo autor.

Ainda nesta tela é possível que o usuário faça um auto cadastro clicando no *link* `registrar`. Ao clicar neste *link* a aplicação direciona para a tela de `cadastro de usuário`. A Figura 13 apresenta a tela de `cadastro de usuário`.

Figura 13 – Tela de `cadastro de usuário`



Fonte: Elaborado pelo autor.

As informações de `login`, `senha` e `confirmação de senha` são obrigatórias para o cadastro de usuário. Os campos `senha` e `confirmação de senha` devem estar com mesmo valor ao salvar, caso contrário o sistema apresentará uma mensagem. O botão superior direito indica para salvar o registro de usuário. Já o superior esquerdo volta à página de *login*. Se ao clicar em salvar e o cadastro ocorrer corretamente, a aplicação emite uma mensagem informando que o cadastro foi efetuado com sucesso. Ao clicar no botão para voltar o campo *login* é automaticamente preenchido com o *login* do usuário salvo anteriormente. Caso tenha alguma divergência a aplicação emite uma mensagem alertando o usuário e mantêm no cadastro.

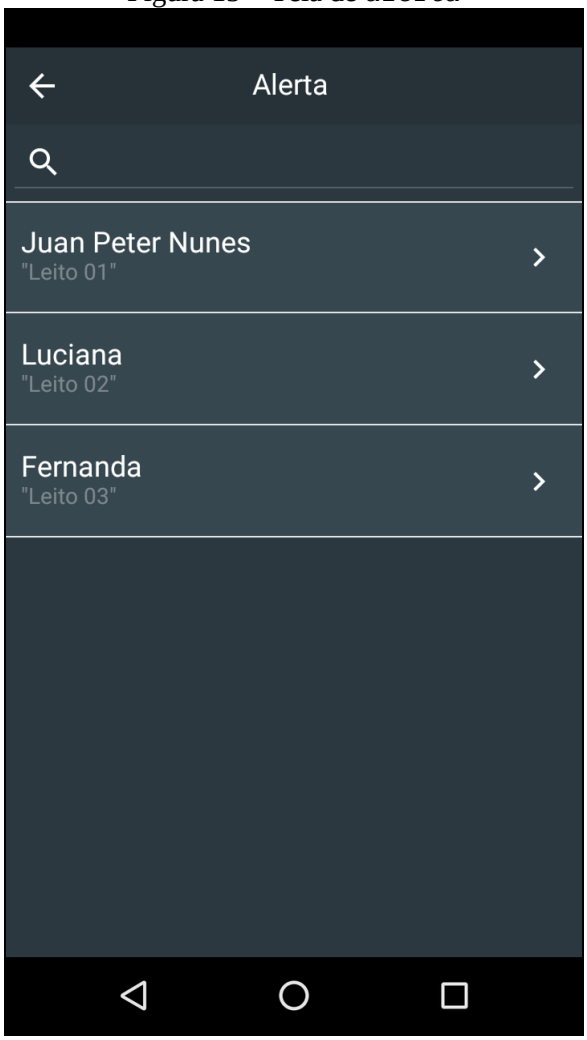
Conforme foi visto na Figura 12, após efetuar o *login* caso o usuário seja administrador a aplicação posiciona no menu principal do sistema. A Figura 14 apresenta o `menu principal` da aplicação.

Figura 14 – Tela do `menu principal`

Fonte: Elaborado pelo autor.

Como o intuito da aplicação é servir os profissionais de saúde com agilidade o cadastro de informações deve ser responsabilidade somente ao administrador da aplicação. A tela de `menu` dá acesso a todos os cadastros da aplicação e a tela de `alerta`. No caso, ao efetuar o *login* com um usuário não administrador a aplicação posicionará diretamente na tela de `alerta`. A tela de `alerta` pode ser visualizada na Figura 15.

Figura 15 – Tela de `alerta`



Fonte: Elaborado pelo autor.

A tela de `alerta` aciona automaticamente a *thread* que faz o escaneamento dos *beacon*s através do Bluetooth. Caso não possua o Bluetooth ativo o sistema emitirá uma mensagem informando que o Bluetooth deve ser ativado. Somente serão apresentados registros nessa lista se:

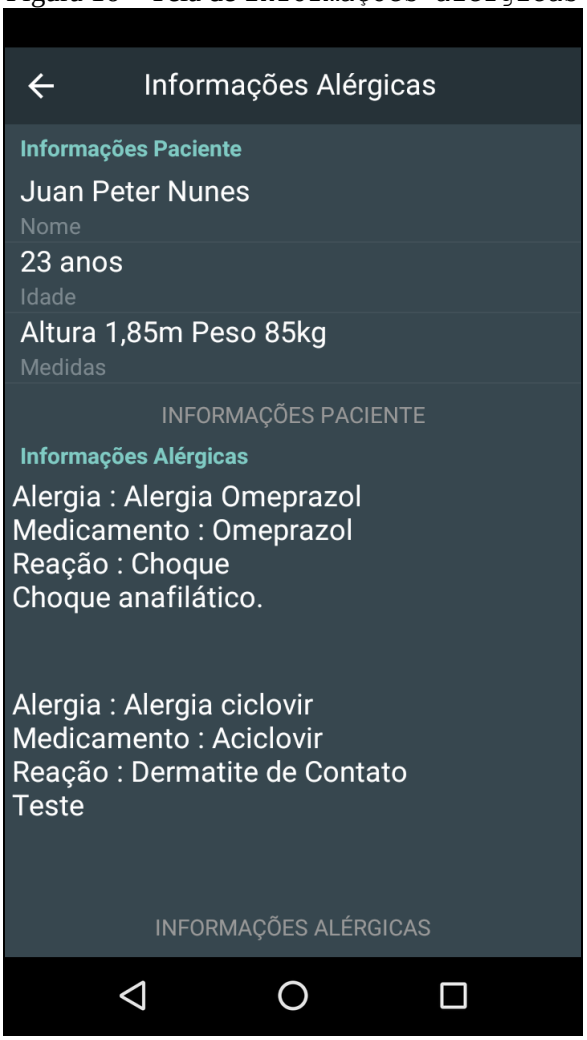
a) leito estiver vinculado com um *beacon*;
b) leito estiver ocupado com um paciente;
c) *beacon* estiver anexado ao leito;
d) usuário estiver a menos de um metro de distância do leito.

Atendendo essas regras, o sistema alimenta a lista de alertas. Caso o usuário se afaste do leito o sistema entende que não é mais necessário manter o alerta e apaga a informação da lista. A grande utilidade da tela de `alerta` é informar de uma maneira facilitada que existe um paciente neste leito. Caso o usuário deseje visualizar as informações alérgica do paciente, ele deve clicar sobre um alerta. O sistema consiste se o paciente do leito possui informações

alérgicas cadastradas no sistema, essa consistência que é feita no lado do servidor da aplicação.

Passando por este fluxo a aplicação apresenta a tela de `informações alérgicas` do paciente que pode ser visto na Figura 16.

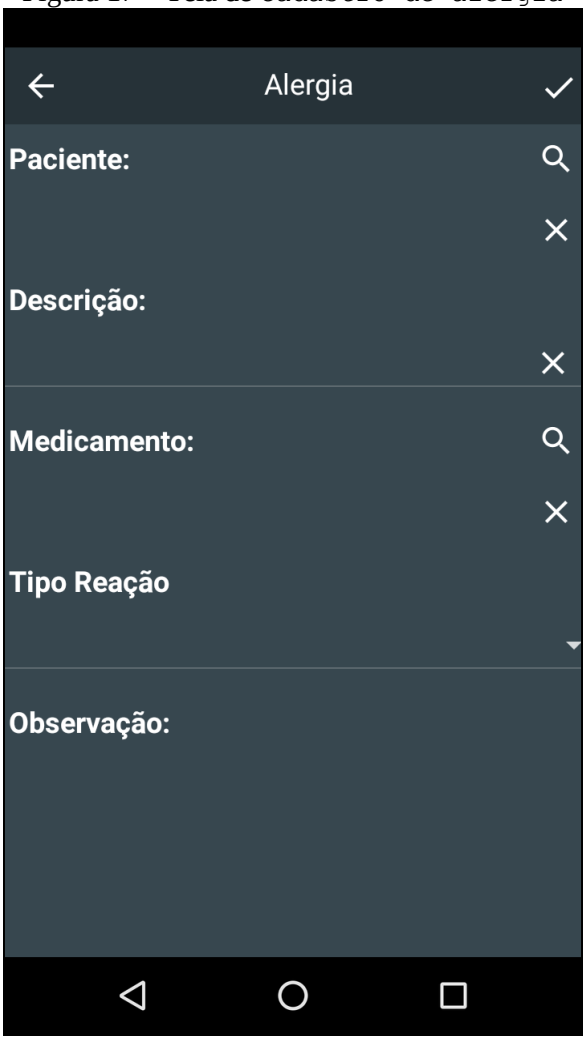
Figura 16 – Tela de `informações alérgicas`



Fonte: Elaborado pelo autor.

As informações de `medidas` e `idade` servem apenas de apoio ao profissional de saúde. Nenhum componente desta tela é editável para o usuário ou administrador. As informações alérgicas correspondem ao `medicamento`, `reação alérgica` e uma breve `observação` previamente cadastradas no `cadastro de alergia`. Na Figura 17 pode-se visualizar o `cadastro de alergia`.

Figura 17 – Tela de `cadastro de alergia`



Fonte: Elaborado pelo autor.

O `cadastro de alergia` pode ser acessado pelo menu no botão `Alergia` que foi visualizado na Figura 14. Neste cadastro as informações de `paciente`, `medicamento` e `tipo de reação` são obrigatórias. Para essas informações não existem campos de texto, apenas podem ser selecionados em uma pesquisa acionada pela `lupa` a sua direita. Conforme o registro é selecionado o componente `TListBoxItem` referente é preenchido com sua descrição.

A tela de `cadastro de medicamento` é onde a aplicação armazena os medicamentos que podem ser potencialmente alérgicos. A Figura 18 demonstra a tela de `consulta` e `cadastro de medicamentos` que pode ser acessada pelo menu principal `Medicamento`.

Figura 18 – Tela de `consulta` e `cadastro de medicamentos`

Fonte: Elaborado pelo autor.

Na tela de consulta pode-se pesquisar e excluir medicamentos já existentes. A pesquisa é feita através da `lupa` superior. Já a exclusão é acionada no botão direito de cada item com a descrição `Del`. A aplicação barra a exclusão caso o medicamento esteja vinculado a alguma alergia medicamentosa. O cadastro de um novo medicamento pode ser acionado pelo botão com símbolo de soma no canto superior direito. Este botão posiciona na tela de `cadastro de medicamento` com o campo de `descrição` em branco. Já a alteração de um medicamento pode ser acionada selecionando algum item na tela de consulta. A aplicação posiciona na tela de `cadastro de medicamento` com a `descrição` do medicamento selecionado.

A tela de `cadastro de tipo de reação alérgica` é onde ficam armazenadas as possíveis reações alérgicas a medicamentos. A Figura 19 demonstra a tela de consulta e cadastro de reações alérgicas que pode ser acessada pelo menu principal `Reação`.

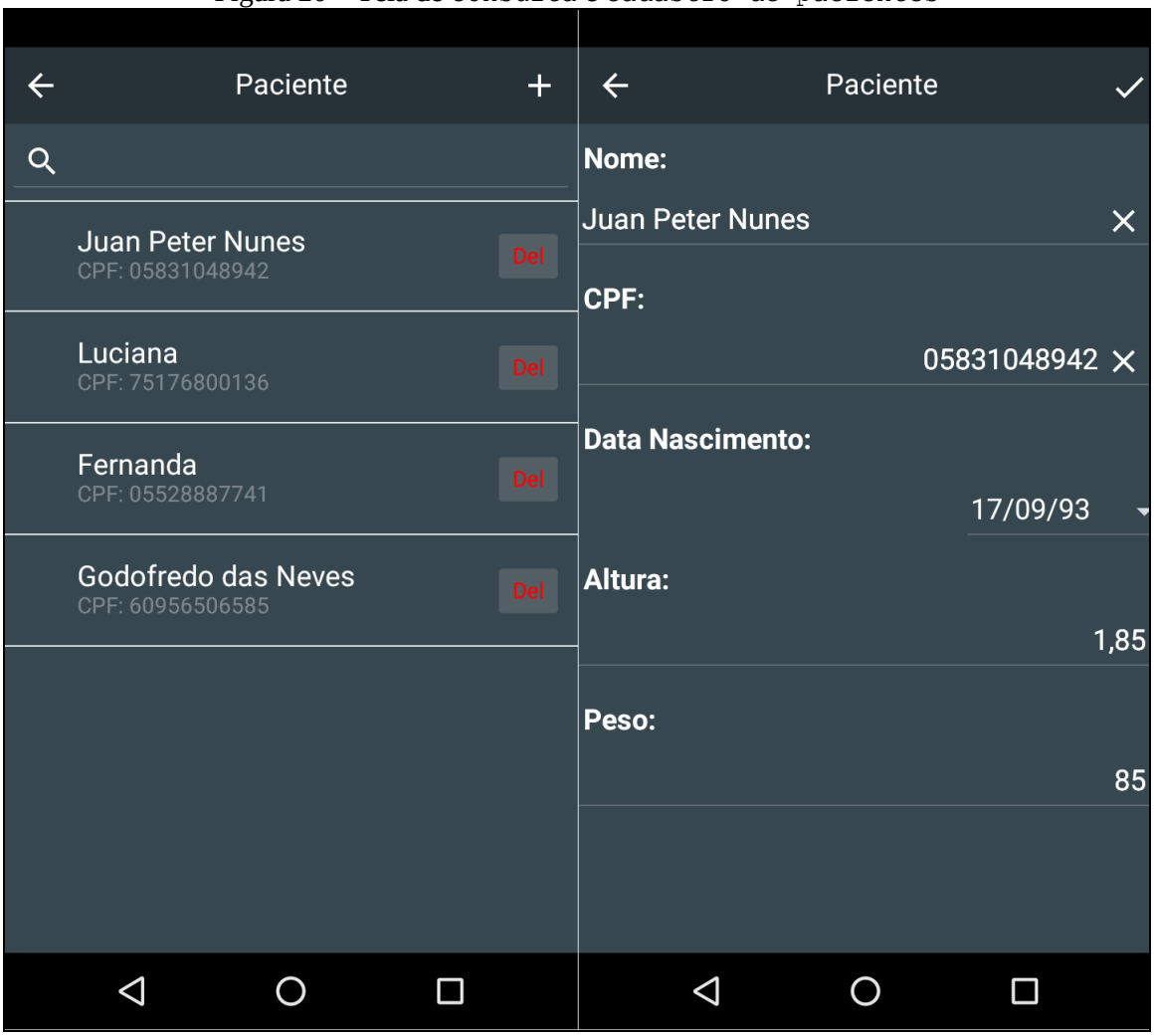
Figura 19 – Tela de `consulta` e `cadastro de tipo de reação alérgica`

Fonte: Elaborado pelo autor.

Na tela de consulta pode-se pesquisar e excluir tipo de reação já existentes. A pesquisa é feita através da `lupa` superior. Já a exclusão é acionada no botão direito de cada item com a descrição `Del`. A aplicação barra a exclusão caso a reação esteja vinculada a alguma alergia medicamentosa. A inserção de um novo registro pode ser acionada pelo botão com símbolo de soma no canto superior direito da tela de consulta. Este botão posiciona na tela de `cadastro de tipo de reação alérgica` com o campo de `descrição` em branco. Já a alteração de um tipo de reação alérgica pode ser acionada selecionando algum item na consulta. A aplicação posiciona na tela de `cadastro de reações alérgicas` com a descrição da reação carregada no campo `descrição`.

A tela de `cadastro de pacientes` é onde são cadastrados os pacientes da aplicação. A Figura 20 demonstra a tela de `consulta` e `cadastro de pacientes` que pode ser acessada pelo menu principal `Paciente`.

Figura 20 – Tela de `consulta` e `cadastro de pacientes`

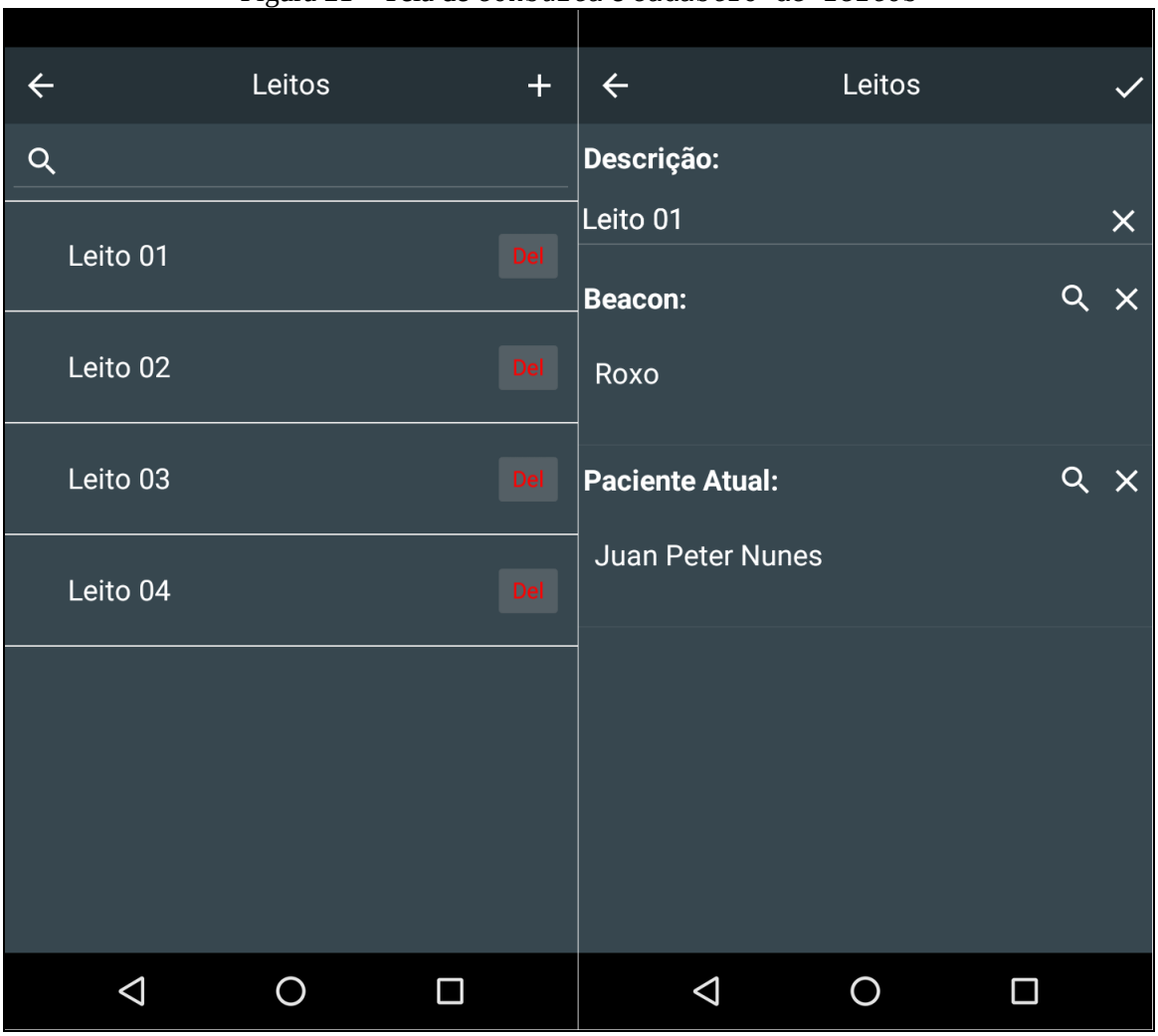


Fonte: Elaborado pelo autor.

Na tela de consulta é possível pesquisar os pacientes cadastrados e fazer a exclusão dos mesmos. A exclusão é acionada no botão direito de cada item com a descrição `Del`. A aplicação barra a exclusão caso o paciente tenha alergias vinculadas ou se o paciente está vinculado a algum leito. O cadastro de um novo paciente pode ser acionado pelo botão com símbolo de soma no canto superior direito da tela de consulta. Este botão posiciona na tela de cadastro com todos as informações do paciente sem preenchimento. As informações de `CPF`, `nome` e `data nascimento` são obrigatórias. Os campos `altura` e `peso` são informações não obrigatórias, sendo utilizadas apenas para uso do profissional de saúde. Já a alteração de um paciente pode ser acionada selecionando algum item na consulta. A aplicação posiciona na tela de `cadastro de pacientes` com cada informação do paciente carregada em seu respectivo campo.

A tela de `cadastro de leitos` é onde são cadastrados os leitos hospitalares. A Figura 21 demonstra a tela de consulta e cadastro de leitos que pode ser acessada pelo menu principal `Leito`.

Figura 21 – Tela de `consulta` e `cadastro de leitos`

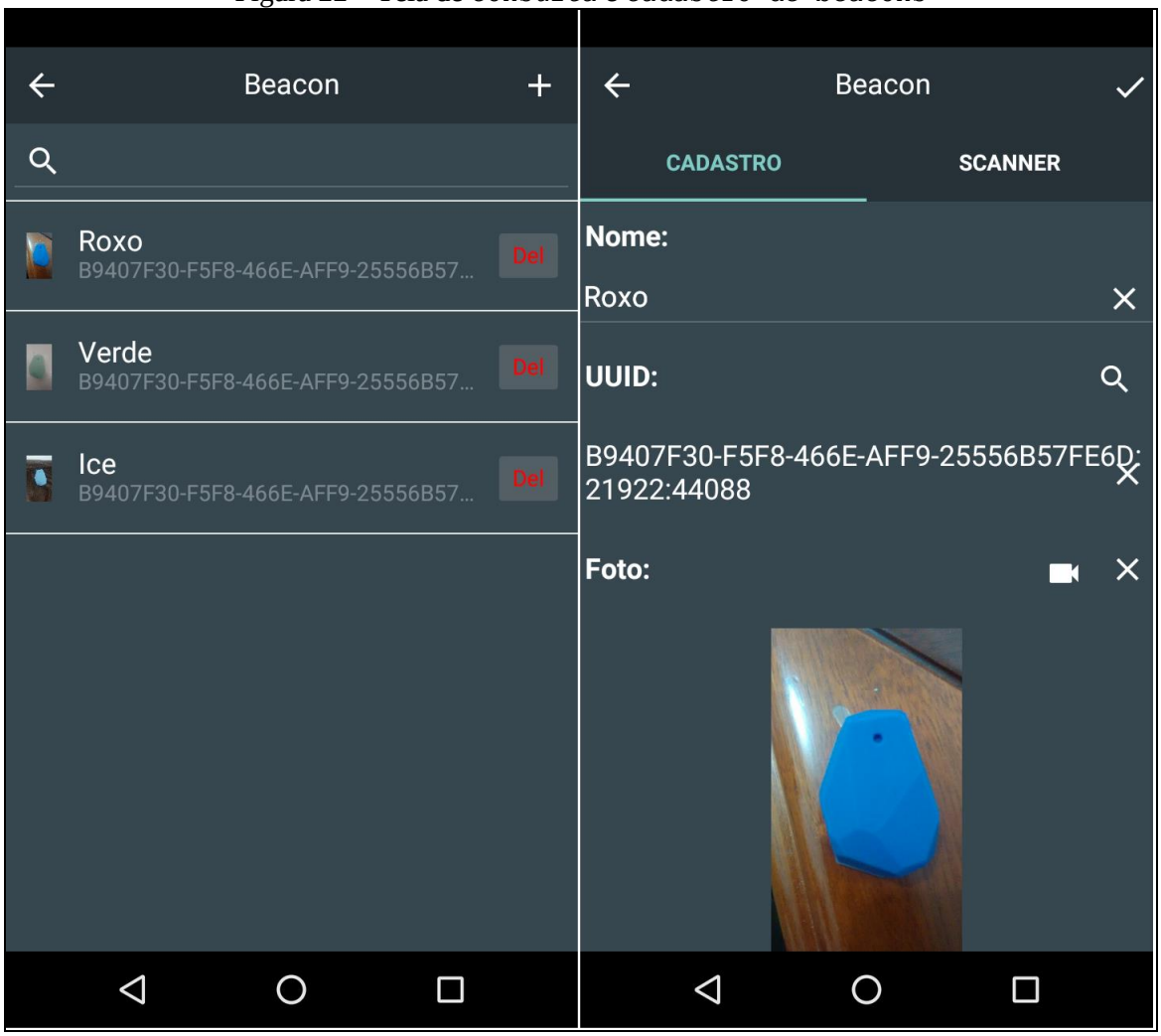


Fonte: Elaborado pelo autor.

Na tela de consulta é possível pesquisar os leitos cadastrados e fazer a exclusão dos mesmos. A exclusão é acionada no botão direito de cada item com a descrição `Del`. A inserção de um novo leito pode ser acionada pelo botão com símbolo de `soma` no canto superior direito da tela de consulta. Este botão posiciona na tela de cadastro com todos as informações do leito sem preenchimento. No cadastro além de informar sua `descrição`, é possível vincular um `beacon` e o `paciente atual`. O `paciente atual` indica que o leito está ocupado por aquele paciente, essa informação não é obrigatória, mas é utilizada para realizar o alerta da Figura 15.

A tela cadastro de *beacon* é onde é armazenado as informações dos dispositivos *beacon*. A Figura 22 demonstra a tela de consulta e cadastro de *beacons* que pode ser acessada pelo menu principal `Beacon`.

Figura 22 – Tela de `consulta` e `cadastro de beacons`



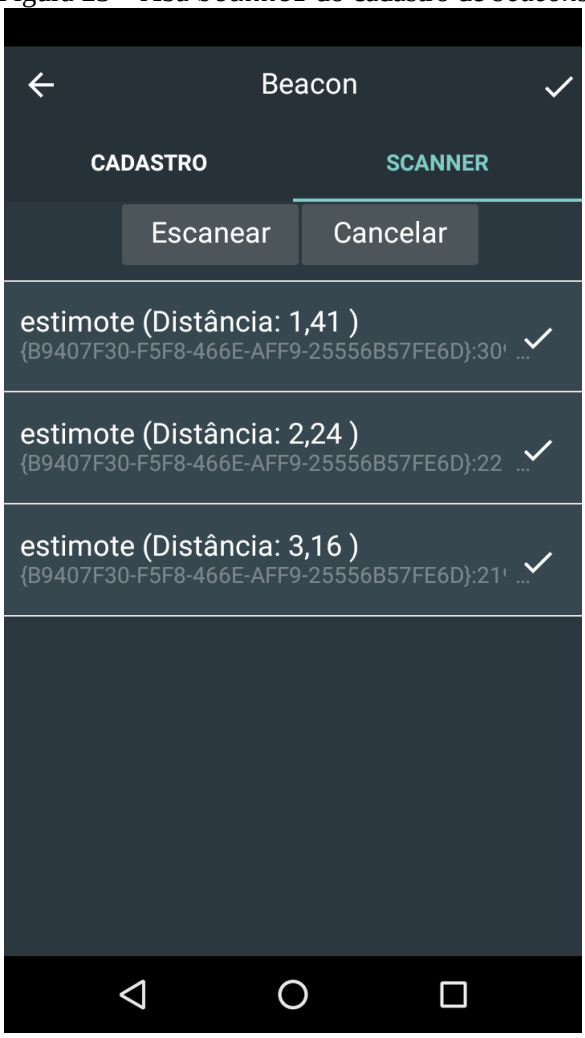
Fonte: Elaborado pelo autor.

Na tela de consulta é possível pesquisar os *beacons* cadastrados e fazer a exclusão dos mesmos. A exclusão é acionada no botão direito de cada item com a descrição `Del`. A aplicação barra a exclusão caso algum *beacon* esteja vinculado a algum leito.

A inserção de um novo registro pode ser acionada pelo botão com símbolo de `soma` no canto superior direito da tela de consulta. Este botão posiciona na tela de cadastro com todos as informações do *beacon* sem preenchimento. Neste cadastro é possível armazenar um `nome`, uma `foto` do *beacon* e seu `UUID`. Sua principal informação é o campo `UUID` que diferencia o dispositivo dos demais. O `UUID` nessa aplicação foi armazenado concatenando as informações de UUID do dispositivo mais o *major* e *minor* descritos na Figura 2 deste trabalho. Para

cadastrar essa informação foi desenvolvido a aba `Scanner`. A aba `Scanner` é acionada caso o usuário clique na `lupa` do item `UUID`, ou se selecionar por conta própria. A Figura 23 representa a aba `Scanner` do cadastro de *beacon*s.

Figura 23 – Aba `Scanner` do cadastro de *beacon*s



Fonte: Elaborado pelo autor.

Nesta aba existem dois botões, sendo o `Escanear` e o `Cancelar`. O botão `Escanear` ativa uma busca por dispositivos *beacon*s nas proximidades do aparelho. É consistido se o aparelho está com a opção Bluetooth ativa após clicar no botão. O botão `Cancelar` interrompe o processo. Pela lógica, o administrador deve aproximar o dispositivo móvel ao *beacon* e selecionar o que estiver com a distância mais adequada. Após selecionar um registro da lista a aplicação move a informação do detalhe para o campo `UUID` e posiciona na aba `cadastro` novamente.

3.4 RESULTADOS E DISCUSSÕES

O Quadro 13 faz uma comparação entre os trabalhos correlatos apresentados e o trabalho desenvolvido, destacando os pontos comuns e diferentes entre eles.

Quadro 13 – Comparação trabalhos correlatos

	CASTRO (2015)	TASY (2010)	MATTOS et al. (2015)	HAMANN (2016) e BIJORA (2014)	AllergyBeacon (2016)
Cadastro da alergia pelo profissional	-	X	X	-	X
Cadastro da alergia pelo paciente	X	-	-	X	-
Alertas no momento da prescrição	-	X	X	-	X
Alergias apresentadas no prontuário do paciente	-	X	X	-	-
Apresentação da alergia sem precisar consultar o paciente	-	-	-	-	X
Usa dispositivo externo *beacon*.	-	-	-	-	X
Plataforma	*Mobile*	*Desktop*	*Web*	*Mobile*	*Mobile*

Fonte: Elaborado pelo autor.

Pode-se perceber que o trabalho desenvolvido não apresenta as alergias no prontuário do paciente, pois a aplicação busca tratar somente da informação alérgica a ser consultada no momento da prescrição ou gestão medicamentosa. A aplicação tem como diferencial a apresentação das alergias do paciente sem que o profissional precise consultar o paciente, pois o paciente será identificado através dos *beacons* em seu leito quando o profissional fazer a aproximação e seu dispositivo móvel.

Para alcançar o objetivo de testar a viabilidade dos *beacons* foram estipulados seis casos de testes. O objetivo dos casos de testes foi testar o comportamento da aplicação na prática. Todos os testes foram realizados com dados já cadastrados e vinculados corretamente do *beacon* e leito utilizando assim a tela de alerta demonstrada na Figura 15. Os casos validam como os *beacons* se comportariam em distâncias e quantidades diferentes em um ambiente de laboratório. A ideia do ambiente de laboratório foi simular um ambiente hospitalar real. No Quadro 14 pode–se identificar os resultados esperados e obtidos dos casos de teste.

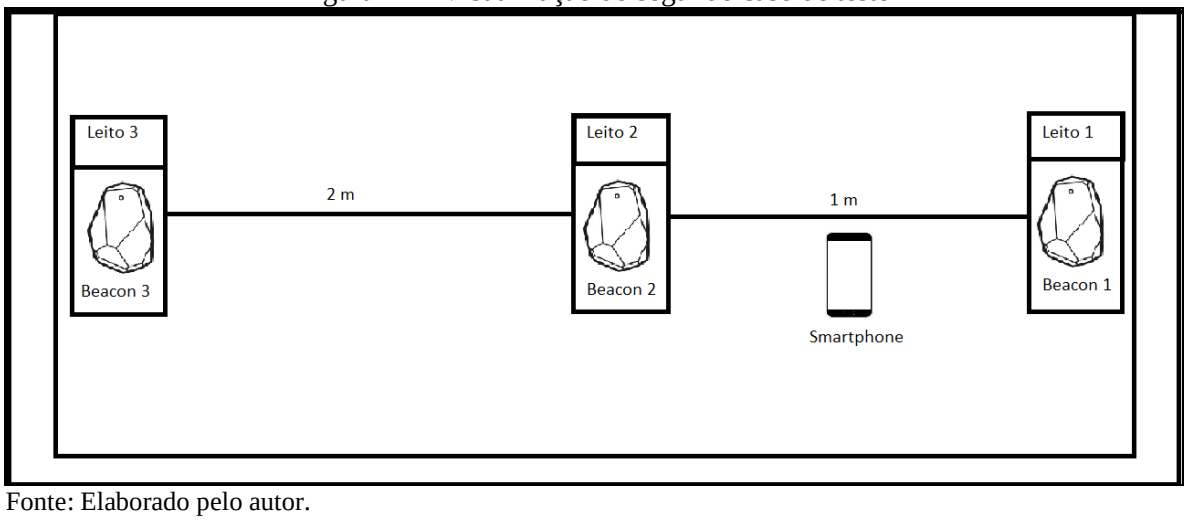
Quadro 14 – Quadro com resultados esperado e obtido dos casos de testes.

Caso	Resultado Esperado	Resultado Obtido
Caso 1 – 2 *beacons* com distância de 1m e celular posicionado no meio.	Identificar 2 *beacons*	Verdadeiro.
Caso 2 – 3 *beacons* com distância de 1m dos dois primeiros e celular posicionado no meio. Terceiro *beacon* posicionado a 2m.	Identificar 2 *beacons*.	Verdadeiro.
Caso 3 – 1 *beacon* com distância de 10 cm do celular.	Identificar 1 *beacon.*	Verdadeiro.
Caso 4 – 1 *beacon* com distância de 2m.	Não identificar o *beacon*.	Verdadeiro.
Caso 5 – 3 *beacons* com distância de 1m celular posicionado no meio.	Identificar 3 *beacons*.	Verdadeiro.
Caso 5 – 3 *beacons* com distância de 10 cm celular posicionado no meio.	Indentificar 3 *beacons*	Verdadeiro

Fonte: Elaborado pelo autor.

No primeiro caso o objetivo foi posicionar dois *beacons* com distância entre eles de um metro. O posicionamento do dispositivo móvel entre os dois *beacons* deveria resultar a identificação de dois dispositivos e foi obtido. Possíveis falhas poderiam ocorrer devido ao fato de um *beacon* sobrepor o sinal do outro, mas a aplicação conseguiu captura-los corretamente. Na Figura 24 pode-se ter a visão do posicionamento dos *beacons* com o smarthphone em relação ao segundo caso de teste.

Figura 24 – Visualização do segundo caso de teste



Fonte: Elaborado pelo autor.

O segundo caso visa capturar dois *beacons* próximos e ignorar o *beacon* que está a mais de um metro de distância. O objetivo foi alcançado, porém foi possível notar uma demora na identificação dos dispositivos. O terceiro caso visa testar um único *beacon* em uma distância menor do que um metro. A identificação do dispositivo ocorreu de forma normal e o resultado esperado foi obtido. O quarto caso que visa ignorar o *beacon* com distância superior a um metro. Este caso de teste também foi alcançado, para isso foi aguardado cerca de trezentos segundos para que a validação fosse concluída e a aplicação não identificou nenhum *beacon* neste intervalo de tempo.

O quinto caso testou o posicionamento de três *beacons* com distâncias de um metro entre os mesmos. O dispositivo móvel foi posicionado no meio e o objetivo foi alcançado, porém a demora também foi notada. Por último o sexto caso que testa três *beacons* com proximidade a dez centímetros com o dispositivo móvel posicionado ao meio.

Devido a lentidão foram elaborados caso de testes para medir a performance na identificação dos *beacons* pela aplicação. Na Tabela 1 pode–se visualizar o tempo estimado da identificação dos *beacons* de cada caso de teste.

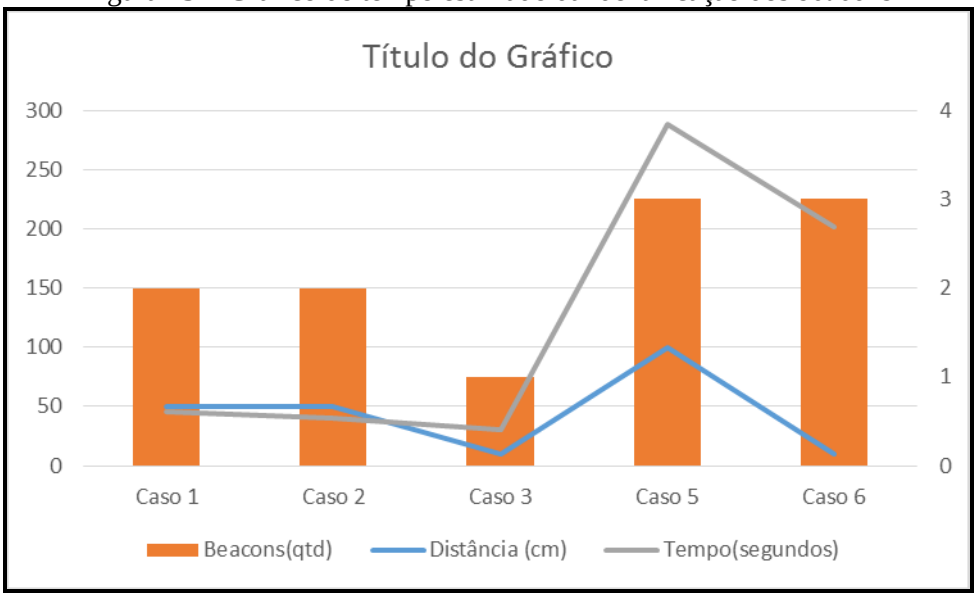
Tabela 1 - Tempo estimado para identificação dos *beacons*

Caso	**Beacons**	**Distância**	**Tempo de identificação**
Caso 1	2	50 cm	46 segundos
Caso 2	2	10 cm	40 segundos
Caso 3	1	10 cm	30 segundos
Caso 5	3	100 cm	288 segundos
Caso 6	3	10 cm	202 segundos

Fonte: Elaborado pelo autor.

O quarto caso foi removido desta avaliação pois o mesmo não identificou nenhum *beacon* e seus dados seriam irrelevantes. Desta forma pode-se visualizar na Figura 25 a representação da Tabela 1 em modo de gráfico.

Figura 25 - Gráfico do tempo estimado da identificação dos *beacons*



Fonte: Elaborado pelo autor.

É possível visualizar um aumento no tempo considerado alto de acordo com a distância e a quantidade de *beacons* de cada caso. O primeiro e o segundo caso tiveram resultados próximos, a distância não interferiu e a quantidade de *beacons* foi a mesma. Já no caso de número três como foi utilizado um único dispositivo seu tempo de resposta foi menor em consideração aos demais. É possível visualizar nos casos cinco e seis que a quantidade de *beacons* influenciou de forma negativa a demora da identificação do *beacon*. Portanto, da maneira com que foi desenvolvida, a aplicação demonstrou uma característica demorada quanto a identificação se comparado com NFC.

Uma limitação técnica encontrada com os dispositivos *beacons* se deu na alteração de suas propriedades. A fabricante disponibiliza uma aplicação móvel para alterar as configurações de hardware do dispositivo *beacon*, como o *tx Power*. Porém esta aplicação se demonstrou falha e não foi possível alterar as configurações dos *beacons*. Desta forma foi tido como uma limitação técnica do fabricante.

4 CONCLUSÕES

O objetivo de facilitar o acesso as informações alérgicas com uso da tecnologia *IoT beacons* foi alcançado. A geração da informação facilitada ao profissional de saúde também visa melhorar a forma como o mesmo trata questões sérias como alergias do paciente.

A utilização da ferramenta de desenvolvimento Delphi 10 Seattle se mostrou robusto e facilitou o desenvolvimento da aplicação. A grande gama de componentes da própria ferramenta para banco de dados, componentes mobile, *framework* cliente–servidor e componentes *IoT* também foi de grande relevância para o desenvolvimento.

Deve–se preocupar com o consumo de bateria do dispositivo móvel devido escaneamento ser contínuo da *thread* na busca pelos *beacons*. Algumas limitações técnicas foram encontradas durante o desenvolvimento com os dispositivos *beacons* quanto ao fabricante. O uso dos dispositivos já usados também dificultou quanto a duração da sua bateria. Foi levado em conta como um fator de risco aos testes finais.

Em relação aos objetivos pode-se considerar que aos testes finais da aplicação rodando em um ambiente de laboratório foi atendido. O dispositivo *beacon*, apesar de apresentar uma demora no reconhecimento, atendeu às expectativas parcialmente se mostrando preciso e confiável, porém, com um tempo identificação demorado.

4.1 EXTENSÕES

Como extensão desse trabalho sugere-se:

a) desenvolver a aplicação para plataforma iOS;
b) implementar sincronização com banco de dados local, removendo a dependência do uso de internet para o funcionamento da aplicação;
c) implementar serviço de segundo plano para checar a proximidade do *beacon* sem a necessidade da aplicação aberta. Assim o serviço pode acionar o profissional com uma notificação do tipo *push* e abrir a aplicação para visualizar as informações mais detalhadas. Atentar ao consumo de bateria do dispositivo móvel;
d) integrar a aplicação com algum sistema hospitalar legado já existente mesclando as informações dos pacientes, leitos, medicamentos e alergias;
e) Explorar o sensor de temperatura e acelerômetro dos *beacons*.

REFERÊNCIAS

AFONSO, Bruno S.; PEREIRA, Roberto B. de O.; PEREIRA, Mauricio F. L. **Utilização da Internet das Coisas para o desenvolvimento de miniestação de baixo custo para monitoramento de condições do tempo em áreas agrícolas**. Cuiabá, p. 183–189, 2015. Disponível em: <http://anaiserimt.ic.ufmt.br/index.php/erimt/article/view/50>. Acesso em: 02 abril 2016.

ARAÚJO, Luciana P. de; BERKENBROCK, Carla D. M.; MATTOS, Mauro M. Usando pesquisa-ação no desenvolvimento de um sistema colaborativo para tratamento multidisciplinar na rede do SUS. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS (SBSC), XI, 2014, Curitiba. **Anais....** Curitiba: Sociedade Brasileira de Computação (SBC), p. 111-118.

ASBAI. **Associação Brasileira de Alergia e Imunopatologia**. 2012. Disponível em: <http://www.sbai.org.br/secao.asp?s=81&id=563>. Acessado em 04 abril 16.

BIJORA, Helito. **Para emergências: veja como configurar sua ficha médica no iOS 8**. 2014. Disponível em: < http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/09/para-emergencias-veja-como-configurar-sua-ficha-medica-no-ios-8.html>. Acesso em 20 nov. 16.

CARREIRO, Rodrigo. **Conheça o Rad Studio 10 Seattle**. 2015. Disponível em <http://www.devmedia.com.br/conheca-o-rad-studio-10-seattle/33646#>. Acesso em 03 abril 16.

CASTRO, F. F. M. **Alergia a Medicamentos**. São Paulo 2015. Disponível em: <https://play.google.com/store/apps/details?id=br.com.imabrasil.alergiaamedicamentos&hl=pt_BR>. Acesso em: 02 abril 2016.

COPETTI, Cinara; GHISLENI, Taís, S. **Mobile marketing:** a tecnologia qr code utilizada em ação da Heineken. 2013. 11 f. Trabalho Final de Graduação (TFG) – Centro Universitário Franciscano, Santa Maria, RS, 2013.

CINELLI, Lucas P.; DUARTE, Otto C. M. B. Unified Authentication with Smart Cards via Near Field Communication. In: NETWORK VIRTUALIZATION AND INTELLIGENCE FOR THE FUTURE INTERNET, 2. 2013, Angra dos Reis. Rio de Janeiro: Wnetvirt'2013, 2013. 1 p. **Anais....** Disponível em: <http://www.gta.ufrj.br/publicacoes/#2013>. Acesso em: 20 nov. 2016.

CRUCIOL, Joice Mara et al. **Avaliação de prescrições medicamentosas de um hospital universitário brasileiro**. Londrina, p. 192-193, 2008. Disponível em: <https://www.researchgate.net/publication/250991598_Avaliacao_de_prescricoes_medicamentosas_de_um_hospital_universitario_brasileiro>. Acesso em: 27 mar. 16.

EMBARCADERO. **DataSnap Overview and Architecture**. 2014. Disponível em: < http://docwiki.embarcadero.com/RADStudio/Seattle/en/DataSnap_Overview_and_Architecture>Acesso em 18 nov. 16.

EMBARCADERO. **FireDAC Multi-Device Data Access Library**. 2016. Disponível em: <https://www.embarcadero.com/br/products/rad-studio/firedac>. Acesso em 18 nov. 16.

EMBARCADERO. **Internet of Things Solutions**. 2015. Disponível em: <https://www.embarcadero.com/br/solutions/internet-of-things>. Acesso em 03 abril 16.

EMBARCADERO. **System.Bluetooth.TBluetoothLEDevice**. 2014. Disponível em: < http://docwiki.embarcadero.com/Libraries/XE8/en/System.Bluetooth.TBluetoothLEDevice>. Acesso em 18 nov. 16.

EMBARCADERO. **BeaconFence**. 2016. Disponível em: <https://www.embarcadero.com/br/products/beaconfence>. Acesso 03 abril 16.

HAMANN, Renan. **Como adicionar informações de emergência na tela de bloqueio do Android**. 2016. Disponível em: < http://www.tecmundo.com.br/android-nougat/102186-adicionar-informacoes-emergencia-tela-bloqueio-android-n.htm>. Acesso em 20 nov. 16.

KARASINSKI, Lucas. **O que significa cada quadrado de um QR Code?**.2013. Disponível em: <http://www.tecmundo.com.br/qr-code/37372-o-que-significa-cada-quadrado-de-um-qr-code-.htm>. Acesso em 20 nov. 16.

KOHN, L. T; CORRIGAN, J. M.; DONALDSON, M. S. **To err is human:** building a safer health system. Washington: National Academy Press, 2000. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=Jj25GlLKXSgC&oi=fnd&pg=PT25&ots=bHljtsK09C&sig=geE4ltRd8f0CLyesSB8TCtX9jj4&redir_esc=y#v=onepage&q&f=false>. Acesso em 04 abr. 2016.

MAIO, Antônio José de Freitas. **Bluetooth Low Energy para monitorização da postura no ciclismo**. Portugal, p. 1–139, 2014. Disponível em: <http://repositorium.sdum.uminho.pt/handle/1822/34095>. Acesso em 03 abril 16.

MANGANELLI, Jean Marin. **Tecnologias de redes sem fio para smart grids**. 2014. 51 f. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

MATTOS, Mauro M. et al. PRONTO System: integration between doctors and pharmacists in the basic health care. In: INTERNATIONAL CONFERENCE SOFTWARE ENGINEERING RESEARCH AND PRACTICE (SERP), 15, 2015, Las Vegas. **Anais...** Las Vegas, EUA: Mercury, 2015.

MENEGOTTO, José Luis. Sensoriamento da edificação: um sistema de localização baseado em beacons BLE. In: ENCONTRO BRASILEIRO DE TECNOLOGIA DE INFORMAÇÃO E COMUNICAÇÃO NA CONSTRUÇÃO, 7. 2015, Recife. **Anais...** Porto Alegre: ANTAC, 2015.

MENEZES, Diogo R. F.; MUNIZ Ana P. **Evolução da linguagem Delphi**. 2014. Disponível em: <http://pt.slideshare.net/diogorochamenezes/evoluo-da-linguagem-delphi-artigo>. Acesso em 03 abril 16.

NÉRI, E. D. R. et al. Erros de prescrição de medicamentos em um hospital brasileiro. **Revista da Associação Médica Brasileira**, São Paulo, v. 57, n. 3, p. 306-314, Maio/Junho. 2011. Disponível em: <http://www.scielo.br/scielo.php?pid=S0104-42302011000300013&script=sci_arttext&tlng=pt/atom.xml>. Acesso em: 04 abr. 2016.

PALMEIRA, Eduardo J. S. de S.; FERNANDES, Luiz R. de A. **Controle de acesso para estádios de futebol utilizando tecnologia NFC (near field communication).** 2013. 84 f. Trabalho de Conclusão de Curso (Graduação) – Universidade Tecnológica Federal do Paraná, Curitiba, 2013.

ROCHA, Carlos E. et al. **Framework para Aplicações em Plataformas Móveis Usando Panoramas com Camadas**. Rio de Janeiro, p. 1–77, 2014. Disponível em: <http://www.visgraf.impa.br/cgi-bin/refQuery.cgi?submit=Browse>. Acesso em 22 maio 16.

RODRIGUES FILHO, José et al. **A tecnologia da informação na área hospitalar:** um caso de implementação de um sistema de registro de pacientes. Curitiba, 2001. Disponível em: <http://www.scielo.br/scielo.php?pid=S1415-65552001000100007&script=sci_arttext&tlng=es>. Acesso em: 27 mar. 16.

SANTOS, Adriano. **Tecnologia na Ponta da Língua**. 2015. Disponível em:<http://www.tdevrocks.com.br/2014/10/31/tutorial-criando-meu-app-step-by-step-datasnaprest-parte-vi>. Acesso em 18 nov. 2016.

SILVA, A., et al. Criatividade e Inovação: Internet das Coisas (IoT – Internet of Things). **Revista Expressão**. Sete Lagoas, n. 9, p. 1-20, mar. 2016.

STECZKIEWICZ, Agnieszka. **How to modify UUID, Major, and Minor values?** Disponível em: <https://community.estimote.com/hc/en-us/articles/200868188-How-to-modify-UUID-Major-and-Minor-values->. Acessado em: 25 de nov. 16.

TASY. **Philips Tasy**. Blumenau 2010. Disponível em: <http://www.cilatam.philips.com.br/solucoes/13/tasy-prestador/>. Acesso em: 02 abril. 2016.

TEIXEIRA, Fabrício. **Tudo o que você precisa saber para começar a brincar com iBeacons**, 2014. Disponível em: <http://arquiteturadeinformacao.com/ux-em-espacos-fisicos/tudo-o-que-voce-precisa-saber-para-comecar-a-brincar-com-ibeacons/>. Acesso em 04 abril 16.

APÊNDICE A – Dicionário de Dados

Este Apêndice apresenta da descrição das entidades do banco de dados mencionadas na seção 3.2.6. Os tipos de dados utilizados para os campos são:

a) `int`: armazena valores inteiros;
b) `decimal`: armazena valores decimais de até 4 bytes;
c) `varcar`: armazena caracteres alfanuméricos;
d) `text`: armazena caracteres alfanuméricos; mas para este trabalho é utilizado para armazenamento de imagens;
e) `bool`: armazena valores do tipo booleano;
f) `date`: armazena os valores do tipo data.

Quadro 15 – Tabela `Usuario`

Usuario – Entidade responsável por armazenar os dados do usuário.				
Campo	Descrição	Tipo	Tamanho	Chave Primária
idUsuario	Código do usuário	Int	11	Sim
Login	Login do usuário	Varchar	100	Não
Senha	Senha do usuário	Varchar	100	Não
Email	*e-mail do usuário*	Varchar	100	Não
Admin	Indica se usuário é administrador.	Bool	1	Não

Fonte: Elaborado pelo autor.

Quadro 16 – Tabela `Beacon`

Beacon – Entidade responsável por armazenar os dados do dispositivo *beacon*				
Campo	Descrição	Tipo	Tamanho	Chave Primária
Idbeacon	Código da tabela beacon	Int	11	Sim
UUID	Chave de rastreamento do *beacon*	Varchar	100	Não
Nome	Nome do *beacon*	Varchar	100	Não
Foto	Foto do *beacon*	Text	-	Não

Fonte: Elaborado pelo autor.

Quadro 17 – Tabela `TipoAlergiaReacao`

TipoAlergiaReacao – Entidade responsável por armazenar os dados dos tipos de reações alérgicas do aplicativo.				
Campo	Descrição	Tipo	Tamanho	Chave Primária
idTipoAlergiaReacao	Código da reação	Int	11	Sim
Descrição	Descrição da reação	Varchar	100	Não

Fonte: Elaborado pelo autor.

Quadro 18 – Tabela `Medicamento`

Medicamento – Entidade responsável por armazenar os dados dos medicamentos				
Campo	Descrição	Tipo	Tamanho	Chave Primária
idMedicamento	Código do medicamneto	Int	11	Sim
Descrição	Descrição do medicamento	Varchar	100	Não

Fonte: Elaborado pelo autor.

Quadro 19 – Tabela `Leito`

Leito – Entidade responsável por armazenar os dados dos leitos				
Campo	Descrição	Tipo	Tamanho	Chave Primária
idLeito	Código do leito	Int	11	Sim
Descricao	Descrição do leito	Varchar	100	Não
Paciente_idPaciente	Código do paciente atual do leito.	Int	11	Não
Beacon_idBeacon	Código do beacon do leito.	Int	11	Não

Fonte: Elaborado pelo autor.

Quadro 20 – Tabela `Paciente`

Paciente – Entidade responsável por armazenar os dados dos pacientes				
Campo	Descrição	Tipo	Tamanho	Chave Primária
idPaciente	Código do paciente	Int	11	Sim
Nome	Nome do paciente	Varchar	100	Não
CPF	CPF do paciente.	Varchar	11	Não

dtNascimento	Data de nascimento do paciente	Date	-	Não
Peso	Peso do paciente	Decimal	10.2	Não
Altura	Altura do paciente	Decimal	10.2	Não

Fonte: Elaborado pelo autor.

Quadro 21 – Tabela `Paciente_Alergia`

Paciente_Alergia – Entidade responsável por armazenar os dados das alergias dos pacientes				
Campo	Descrição	Tipo	Tamanho	Chave Primária
idPaciente_Alergia	Código da alergia	Int	11	Sim
Medicamento_idMedicamento	Código do medicamento alérgico.	Int	11	Não
TipoAlergiaReacao_idTipoAlergiaReacao	Código do tipo de reação da alergia.	Int	11	Não
Descricao	Descrição da alergia	Varchar	100	Não
Observação	Observação da alergia	Text	10.2	Não

Fonte: Elaborado pelo autor.