UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

CONTROLE DE ROTA PARA VIGILANTES UTILIZANDO NFC PARA VALIDAÇÃO DE PRESENÇA

HELIO INACIO DA SILVA JUNIOR

BLUMENAU
2018

HELIO INACIO DA SILVA JUNIOR

CONTROLE DE ROTA PARA VIGILANTES UTILIZANDO NFC PARA VALIDAÇÃO DE PRESENÇA

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.

Prof(a). Alexander Roberto Valdameri, Titulação - Orientador

BLUMENAU
2018

CONTROLE DE ROTA PARA VIGILANTES UTILIZANDO NFC PARA VALIDAÇÃO DE PRESENÇA

Por

HELIO INACIO DA SILVA JUNIOR

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). Alexander Roberto Valdameri, Mestre – Orientador, FURB
Membro:	Prof(a). Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Prof(a). Francisco Adell Péricas, Mestre – FURB

Blumenau, dia 12 de Julho de 2018

Dedico este trabalho a todos que me apoiaram em sua realização, direta e indiretamente.

AGRADECIMENTOS

À Deus por ter me dado força para continuar na caminhada.

À minha família que sempre ficou do meu lado desde o começo.

Ao meu amigo André Raulino por me ajudar na realização deste trabalho.

Ao meu orientador que me auxiliou em todo o caminho, me ajudando em tudo que precisava.

No meio da dificuldade encontra-se a oportunidade.

Albert Einstein

RESUMO

Atualmente, o tema segurança tem se tornado casa vez mais discutido, tanto por pessoas comuns do dia a dia, quanto por empresários, classe política e diversos segmentos da sociedade. Segurança nos dias de hoje é algo que todos precisam e necessitam e muitas vezes para se garantir essa segurança, as pessoas recorrem a vários métodos. Uma das formas mais comuns de segurança que encontradas atualmente são os vigilantes de ronda, onde o trabalho basicamente é feito por um vigia que segue um caminho pré-definido, garantindo assim a segurança de diferentes espaços, entre alojamentos, terrenos entre outros exemplos. Como é um trabalho muito manual, busca-se a automatização de certas funções, para assim facilitar o trabalho do vigia. Este trabalho apresenta implementações de melhorias para um sistema de controle de rotas para vigilantes de ronda utilizando NFC desenvolvido por Raulino (2016). As melhorias incluem um controle dos dispositivos utilizados pelos vigias, fazendo com que o administrador do sistema tenha um maior controle dos acessos e também fazer com que a validação dos pontos não precise estar constantemente conectada à internet. Foi utilizado para o trabalho Java e SQLite. Como resultado, acredita-se que esteja disponível um sistema que atende as necessidades de um vigilante de ronda de forma simples e eficaz, tendo pontos semelhantes com as soluções fornecidas no mercado atualmente.

Palavras-chave: Segurança. NFC. Automatização.

ABSTRACT

Nowadays, the theme of security has become more and more discussed, both by everyday people, as well as by businessmen, political class and various segments of society. Security these days is something that everyone needs and need and often to ensure that security, people resort to various methods. One of the most common forms of security currently encountered is round-the-clock vigilantes, where the work is basically done by a watchman who follows a predefined path, thus ensuring the safety of different spaces, between dwellings, terrains and other examples. As it is a very manual work, the automation of certain functions is sought, in order to facilitate the work of the lookout. This work presents improvements implementations for a route control system for round watchers using NFC developed by Raulino (2016). The improvements include a control of the devices used by the lookouts, making the system administrator have a greater control of the accesses and also make that the validation of the points does not have to be constantly connected to the internet. It was used for working Java and SQLite. As a result, it is believed that a system is available that meets the needs of a round attendant in a simple and effective manner, having similar points with the solutions provided in the market today.

Key-words: Safety. NFC. Automation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CSS – Cascading Style Sheets

DAO – Data Access Object

HTTP – HyperText Transfer Protocol

IDE – Integrated Develeopment Environment

JSON – JavaScript Object Notation

JSP – Java Server Page

MAC – Media Access Control

MER – Modelo Entidade Relacionamento

MDL – Material Design Lite

MVC – Model View Controller

NFC – Near-field Communication

RFID – Radio Frequency Identification

XML – Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente a segurança é uma das maiores preocupações das pessoas (SARDEIRO, 2013). Via de regra, em uma sociedade as pessoas são livres para fazer o que quiserem, desde que suas ações estejam dentro das leis e que não interfiram negativamente na convivência com a população. Entretanto, conforme Bauman (2001, p. 9), "A segurança e a liberdade são dois valores igualmente preciosos e desejados que podem ser bem ou mal equilibrados, mas nunca inteiramente ajustados e sem atrito. ". Diante desse desiquilíbrio, as pessoas buscam cada vez mais por segurança para proteger suas propriedades ou seus familiares.

Nos dias de hoje é possível encontrar no mercado métodos variados de segurança oferecidos por empresas especializadas. Dentre eles podemos destacar os sistemas de circuitos fechados de televisão (CFTV), controle de acesso, sistemas de alarmes de intrusão e segurança perimetral e vários outros (JANES, 2009). Muitos desses sistemas de segurança são automatizados, necessitando muito pouco de uma intervenção humana em seu processo. Entretanto há casos em que se faz necessário a utilização de pessoas no processo de segurança, sendo o caso dos vigilantes que fazem ronda em determinados pontos da propriedade onde os outros processos automatizados não são eficazes ou como uma forma de complementar o sistema de segurança implantado.

Mas como o processo de ronda depende da intervenção humana, emerge a questão da confiabilidade no seu processo. Por vezes há uma preocupação em saber se realmente o vigilante está cumprindo as rotas estabelecidas, se está cumprindo os horários das rondas e se realmente está efetuando a ronda. Conforme Prioste (2014) "O componente mais vulnerável desse sistema é o recurso humano, portanto é aquele que precisa de mais atenção, metodologia e persistência para assimilar e praticar um comportamento seguro". Pensando em prevenir esses problemas de forma segura e definitiva, foram desenvolvidas soluções que controlam esse processo de ronda dos vigilantes. Tais soluções demandam equipamentos específicos, feitos para atender essa necessidade, tornando a aplicação dessas soluções mais custosas.

Diante do cenário apresentado, este trabalho propõe aplicar melhorias em um protótipo de sistema feito por Raulino (2016) que utiliza Near Field Communication (NFC) dos *smartphones* para validação de presença dos vigilantes, oferecendo uma ferramenta mais barata em relação às soluções oferecidas no mercado atualmente.

1.1 OBJETIVOS

O objetivo desse trabalho é aplicar melhorias ao sistema de controle de rondas de vigias que utiliza o NFC para a validação de rota desenvolvido por Raulino (2016).

Os objetivos específicos dessas melhorias são:

a) apresentar alternativas para a realização de conexão entre os módulos do protótipo, de forma que não seja necessária conexão constante à internet;
b) expandir a estrutura de dados que são gerenciados pelo sistema;
c) não impactar no performance do sistema com as melhorias implementadas no sistema.

1.2 ESTRUTURA

Esta monografia está dividida em quatro capítulos. O primeiro capítulo apresenta uma breve introdução ao assunto pesquisado durante o trabalho e os objetivos a serem alcançados durante a pesquisa. O capítulo dois apresenta sobre um pouco sobre o NFC e como pode ser utilizado e também é apresentado sobre sistemas automatizados de segurança e sua importância nos dias de hoje. No terceiro capítulo é apresentado o desenvolvimento das melhorias no sistema, mostrando as técnicas e as ferramentas utilizadas no desenvolvimento, mostrando também sua aplicabilidade. No quarto e último capítulo é apresentado uma conclusão em relação ao trabalho feito e também sugestões de melhorias para o sistema para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo irá abordar temas referentes ao trabalho apresentado, tais como o NFC e sistemas automatizados de segurança. Além disso, apresentará os trabalhos correlatos.

2.1 NFC

Near Field Communication (NFC) é um padrão que permite interações seguras entre dispositivos eletrônicos, permitindo a troca de informações sem a necessidade de cabos. O NFC foi padronizado pelo NFC Fórum, que é um consórcio global criado em 2004 composta por empresas de hardware, software, cartões de crédito, bancos e provedores de internet (NFC FORUM, 2017).

O NFC teve origem do Radio-Frequency IDentification (RFID), mas acabou por se distanciar pois é necessária uma comunicação a curta distância. Essa curta distância foi estabelecida com o objetivo de tornar a comunicação mais segura. De acordo com o trabalho de Nassar e Horn (2014, p. 4):

> As aplicações com NFC podem ser categorizadas em três tipos de operação: a) Reader/Writer: um dispositivo com NFC identifica dados em uma Tag NFC; b) Card Emulation: um dispositivo com NFC simula o comportamento de um cartão, para a realização de transações financeiras ou acesso a catracas e portas; c) Peer-to-Peer: dois dispositivos são emparelhados, estabelecendo uma comunicação entre ambos.

De maneira geral, o NFC pode funcionar de duas maneiras: pode-se trabalhar de maneira ativa e maneira passiva. Se caracteriza O NFC ativo quando os dispositivos geram e recebem o sinal de rádio. Pode-se tomar como exemplo um smartphone, que recebe a informação de um emissor e após isso, emite a confirmação para o mesmo dispositivo.

Já para o NFC passivo, o funcionamento se caracteriza quando o dispositivo apenas recebe a informação. Como exemplo temos as *tags* NFC que funcionam como um agente passivo, pois a *tag* não consegue receber um sinal, apenas emitir. Mas mesmo tendo um funcionamento passivo, as *tags* podem ser trabalhadas juntamente com um *smartphone* que funcionaria como um agente ativo. Nassar e Horn (2014, p. 6) afirmam sobre as *tags* NFC

> Dentre as possibilidades, pode-se integrar o chip em diversos objetos, como pulseiras, chaveiros, cartazes e cartões. Além disso, o chip também pode estar presente em equipamentos eletrônicos, como aparelhos de música, televisores, computadores, entre outros, atuando de forma ativa para acionar funções predeterminadas.

Desta forma, pode-se utilizar o NFC para muitas funções. Dentre elas pode-se destacar a utilização de NFC para pagamentos da mesma forma que se é utilizado os cartões de crédito, para interagir com revistas e produtos e podendo pesquisar informações sobre o assunto em tempo real (ORITZ, 2008).

2.2 SISTEMAS AUTOMATIZADOS DE SEGURANÇA

A segurança nos dias de hoje é vital para assegurar a integridade física tanto de propriedades de pessoas comuns como para propriedades de empresas e organizações. De acordo com Martins (2009, p. 16) “a segurança não se contrapõe à liberdade e é condição para o seu exercício, fazendo parte de uma das inúmeras e complexas vias por onde trafega a qualidade de vida dos cidadãos.”. Com isso, se conclui que a segurança é fundamental e deve estar apta para prevenir riscos a pessoas e propriedades.

Sabendo da importância da segurança nos dias de hoje, as empresas seguem investindo cada vez mais em segurança visando manter de forma segura seus negócios, sem querer correr riscos (CRUZ, 2009). Com a procura por sistema de segurança confiáveis, mais se investe em tecnologias para otimização de processos de segurança.

Um fator determinante que justifica o grande investimento em tecnologias voltadas a segurança é a violência que se torna cada vez mais comum em nossa sociedade. Essa exposição a violência causa transtornos físicos e psicológicos nas pessoas, devido ao medo constante (CARDIA, 2003). Através disso, adicionando uma grande propriedade nesse contexto, podemos observar a grande relevância nos investimentos na tecnologia da segurança pois uma eventual perda ou furto dentro do perímetro da propriedade pode trazer danos tanto quanto financeiros ao proprietário como também danos morais (RAULINO, 2016).

Segundo Lopes (2006, p. 2) “A sensação de insegurança [...] é alimentada por vários fatores: visibilidade dos crimes convencionais, participação intensiva da mídia, o comportamento elitista da sociedade e o distanciamento entre as pessoas.”. A partir disso, empreendedores procuram por sistemas de segurança mais sofisticados para garantir a integridade de seus negócios.

Entre esses investimentos pode destacar a automatização desses processos. Segundo Luz e Kuiawinski (2006, p. 03 apud HOUAISS, 2004) “A automação pode ser definida como um desenvolvimento [...] que os processos operacionais em fábricas são controlados e executados por meio de dispositivos mecânicos ou eletrônicos, substituindo o trabalho humano”. A partir dessa definição tem-se exemplos de sistemas de segurança automatizados oferecidos no mercado atualmente como os controladores de acesso, os circuitos fechados de TV e, como mostrado neste trabalho na seção de trabalhos correlatos, sistemas automatizados para vigilantes de ronda.

2.3 SOFTWARE ATUAL

Desenvolvido por Raulino (2016), o software tem como objetivo gerenciar a ronda de vigilantes utilizando o NFC dos *smartphones* para fazer a validação de presença dos vigilantes. O software foi feito para oferecer uma ferramenta mais prática e mais barata em relação as ferramentas que são comercializadas atualmente.

O sistema é dividido em dois módulos: um *web* e um móvel. No módulo web pode-se cadastrar rotas, vigias e administradores do sistema e visualizar pontos de checagem e execução das rondas. Além disso possui um gerador de QR *code* dos usuários cadastrados no módulo web, onde ao ler o QR *code* com o aplicativo Android a conexão e feita automaticamente, sem a necessidade de se realizar login com os dados pessoais e senha e sem a necessidade de informar o Internet Protocol (IP) e a porta de conexão. No módulo móvel é possível inserir em uma *tag* NFC os pontos de checagem, visualizar rotas sob responsabilidade de cada vigia e realizar uma ronda, gravando o dia e hora do início da ronda, mostrando um horário máximo para a execução da ronda e também mostra quais os próximos pontos para se realizar a validação da rota atual. O *login* ao aplicativo é feito lendo o QR *code* gerado no módulo web ou informado login, senha, IP e a porta de conexão.

Para o desenvolvimento e teste do sistema, Raulino (2016) utilizou um computador com o sistema operacional Windows 10, utilizando para o desenvolvimento do módulo do servidor a Integrated Development Environment (IDE) Eclipse Mars release 4.5.2, para acesso ao banco de dados foi utilizado o pgAdmin 1.20. Para se realizar os testes foi utilizado o servidor Apache Tomcat v7.0. O desenvolvimento do *back-end* do módulo web foi feito em Java 8 e utilizado o padrão Model View Controller (MVC) com auxílio do *framework* brasileiro Mentawai Framework. Esse *framework* foi escolhido pois traz a facilidade de acessar funções do sistema a partir da página web sem a necessidade da utilização de configurações via XML. Já para o *front-end* do módulo web, as páginas foram feitas com Java Server Page (JSP) e Cascading Style Sheets (CSS) para a estruturação da página e também foi utilizado javascript para rodar scripts necessários.

O aplicativo móvel de Raulino (2016) foi desenvolvido em Java utilizando *activities* comuns do android, programando a interface em XML. Para a comunicação entre os módulos do sistema foi utilizado mensagens JavaScript Object Notation (JSON) enviadas através do protocolo de comunicação Hypertext Transfer Protocol (HTTP).

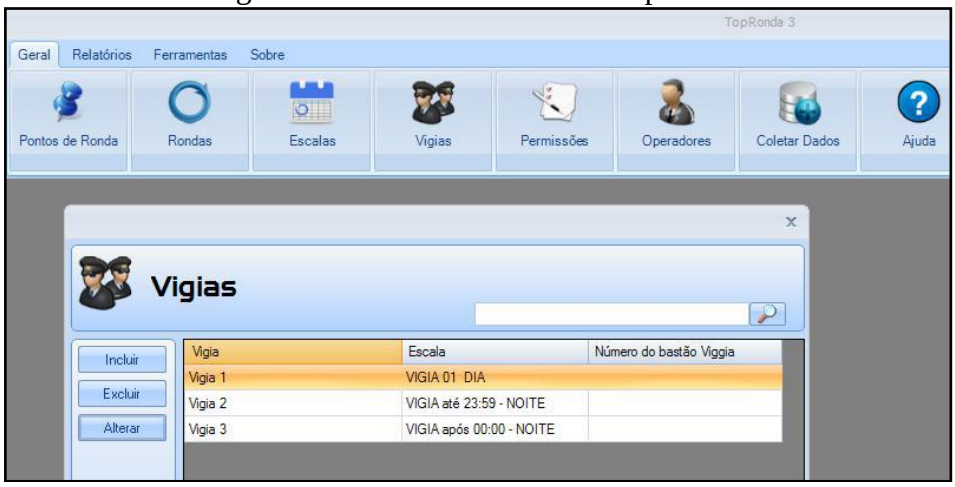
2.4 TRABALHOS CORRELATOS

A seguir serão mostrados três trabalhos correlatos que foram escolhidos pois possuem objetivos semelhantes ao trabalho proposto. O item 2.1 irá abordar o software TopRonda que é um sistema de controle de vigilantes desenvolvido pela TopData (2017). No item 2.3 será apresentado o trabalho de conclusão de curso de Krüguer (2002) que consiste em um protótipo de um sistema de segurança predial e o item 2.3 detalha o trabalho de conclusão de curso de Baumann (2008) que consiste em um protótipo de um sistema de segurança residencial.

O sistema TopRonda possui o objetivo de controlar rondas de vigilantes e por isso é o software que mais se assemelha ao trabalho proposto. Foi desenvolvido pela TopData e permite a configuração de rotas e horários, coletas de dados, emissão de relatórios e utiliza um hardware específico para validação de presença nas rondas (TOPDATA, 2017). O software é instalável e é acessado pelo próprio computador. A Figura 1 mostra como é a interface do sistema, mostrando algumas opções de cadastro e consulta de dados.

Ao contrário deste trabalho que utiliza o NFC dos dispositivos móveis em conjunto com tags NFC para fazer a validação da presença, o sistema TopRonda utiliza um bastão, chamado de Viggia, que funciona em conjunto com botões inteligentes, chamados de *i-buttons*, conforme mostrado na Figura 2. Os *i-buttons* são instalados nos pontos de ronda e com o bastão é feito uma validação por aproximação. Os dados são salvos em uma memória interna do bastão e depois são carregadas no computador que possui o software através da porta USB encontrada no bastão.

Figura 1 - Interface do sistema TopRonda



Fonte: TopData (2017).

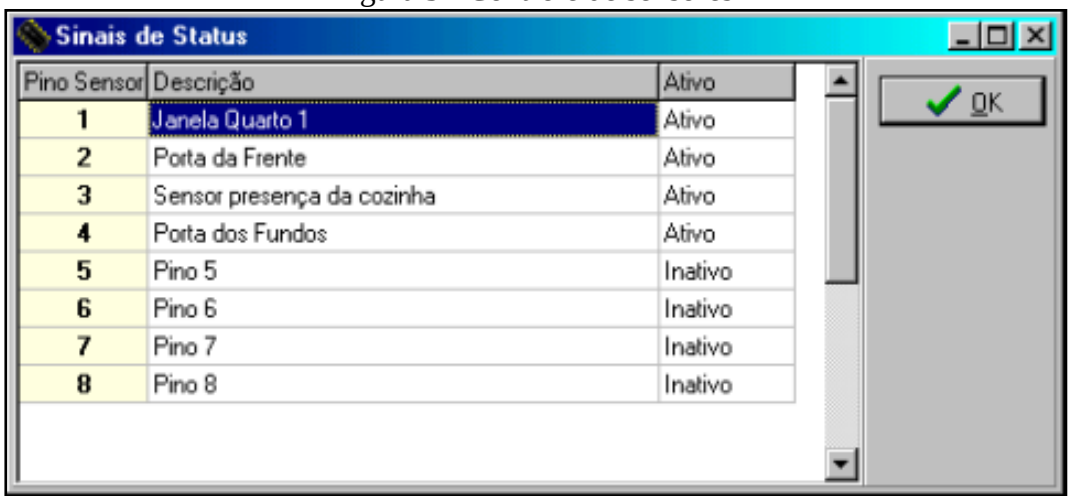
Figura 2 - Bastão Viggia e i-Button

Fonte: TopData (2017).

O trabalho de conclusão de curso feito por Krüguer (2002) tinha como objetivo o desenvolvimento de um protótipo de sistema de segurança, através de monitoramento, voltada para prédios, com elementos simples e de baixo custo utilizando os recursos da internet. O funcionamento da aplicação consiste em realizar o monitoramento da residência através de sensores espalhados na moradia onde o sistema esteja conectado e oferece ao usuário a opção de programar o sistema por horário ou evento. Através desses sensores o sistema recebe as informações através de leituras dos sensores e com essas informações o sistema gera um arquivo de *log*, que pode ser salvo ou enviado por e-mail cadastrado no sistema para o usuário.

Na Figura 3 pode-se observar a tela de gerenciamento de sensores do sistema, onde se pode ver os sensores ativos e inativos e uma descrição de onde estão fisicamente espalhados pela residência. Já na Figura 4 é possível observar um exemplo de arquivo de *log* gerado pelo sistema.

Figura 3 - Controle de sensores



Fonte: Krüguer (2002).

Figura 4 - Arquivo log gerado pelo sistema

```
SISM-NET - Arquivo de LOG
Aplicativo desenvolvido por Erasmo Kruger para TCC - 2002/2
Orientador: Prof. Antonio Carlos Tavares
Data de Criação do arquivo: 07/11/02 23:19:48

07/11/02 23:19:48: Monitoramento Iniciado.
08/11/02 00:00:05: Programação por horário foi acionada: Alimentar Sensor de
Presença
08/11/02 06:58:33: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 07:11:47: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 07:18:41: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 07:32:51: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 07:43:28: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 12:09:25: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 12:58:58: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 13:45:51: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 14:07:45: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 14:27:48: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 14:57:08: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 15:10:14: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 15:16:26: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 15:27:54: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 15:56:12: Evento detectado no sensor: Monitoramento Quarto (Presença)
08/11/02 15:56:39: Monitoramento Finalizado.
```

Fonte: Krüguer (2002).

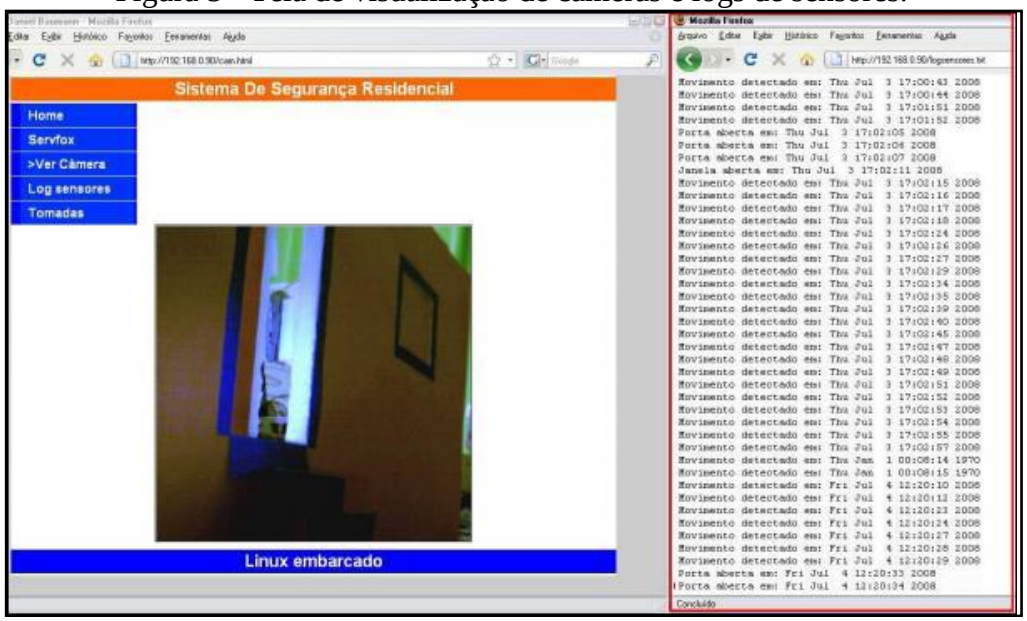
O sistema ainda possui funções especiais devido a sua flexibilidade nas suas configurações. Podendo ser destacado a possibilidade de se ligar e desligar luzes dentro da residência de maneira diferenciada durante o decorrer da semana através da programação por horários, ludibriando possíveis assaltantes. Outro ponto a se destacar é a possibilidade de se gerenciar as rotas e os horários de ronda dos vigilantes, sendo essa a funcionalidade que se assemelha ao que está sendo proposto nesse trabalho (KRÜGER, 2002).

O trabalho desenvolvido por Baumann (2008) tinha como objetivo desenvolver um protótipo de um sistema de segurança residencial, integrando sensores, câmeras e acionamento de tomadas em um único dispositivo, utilizando para isso um hardware denominado Fox Board que possui como núcleo o sistema operacional Linux. Além disso, desenvolveu um protótipo de sistema web onde é possível o usuário do sistema visualizar a câmera que está conectada na Fox Board, verificar qual a data e a hora em que os sensores foram ativados através de logs e

também possibilita acionar ou desacionar tomadas, luzes ou alarmes. Pode-se destacar como vantagem no protótipo desenvolvido por Baumann (2008) o custo benefício e além disso, ser independente de plataforma por ser uma aplicação web.

Como desvantagem pode-se destacar que, para se visualizar os *logs* ou acionar e desacionar tomadas, o protótipo redireciona o usuário a outra página, não concentrando a aplicação toda em apenas uma. Na Figura 5 é possível observar o protótipo web acessando a câmera em tempo real e o *log* dos sensores e na Figura 6 tem-se a placa Fox Board nacional conectada à rede, câmera e circuito de tomadas e sensores.

Figura 5 - Tela de visualização de câmeras e logs de sensores.



Fonte: Baumann (2008).

Figura 6 - Conexão de dispositivos a Fox Board

Fonte: Baumann (2008).

3 DESENVOLVIMENTO

Neste capítulo serão apresentadas as etapas de desenvolvimento propostas no trabalho no módulo web e no módulo mobile. A seção 3.1 mostra os Requisitos Funcionais e os Requisitos Não Funcionais atendidos no planejamento do sistema.

3.1 REQUISITOS

Nesta seção serão apresentados os Requisitos Funcionais e os Requisitos Não Funcionais atendidos no planejamento da melhoria do sistema desenvolvido por Raulino (2016). A aplicação descrita neste trabalho deverá, juntamente com os requisitos funcionais e requisitos não funcionais do software desenvolvido por Raulino (2016):

a) permitir o vigia iniciar uma execução de ronda em uma de suas rotas no aplicativo mobile estando conectado à rede de internet (RF);
b) permitir o vigia verificar o atual ponto de checagem que precise ser validado no aplicativo mobile estando desconectado da rede de internet (RF);
c) permitir o vigia validar um ponto de checagem ao aproximar o aparelho de uma tag NFC no aplicativo mobile estando desconectado da rede de internet (RF);
d) permitir o vigia a encerrar a execução de ronda estando conectado à rede de internet (RF).
e) utilizar Java para o back-end do servidor (RNF);
f) utilizar o Banco de dados PostgreSql para armazenamento de dados no servidor (RNF);
g) executar o cliente no sistema operacional Android (RNF).
h) utilizar a biblioteca SQLite para armazenamento das informações de ponto de checagem e ronda no dispositivo Android (RNF).

3.2 ESPECIFICAÇÃO

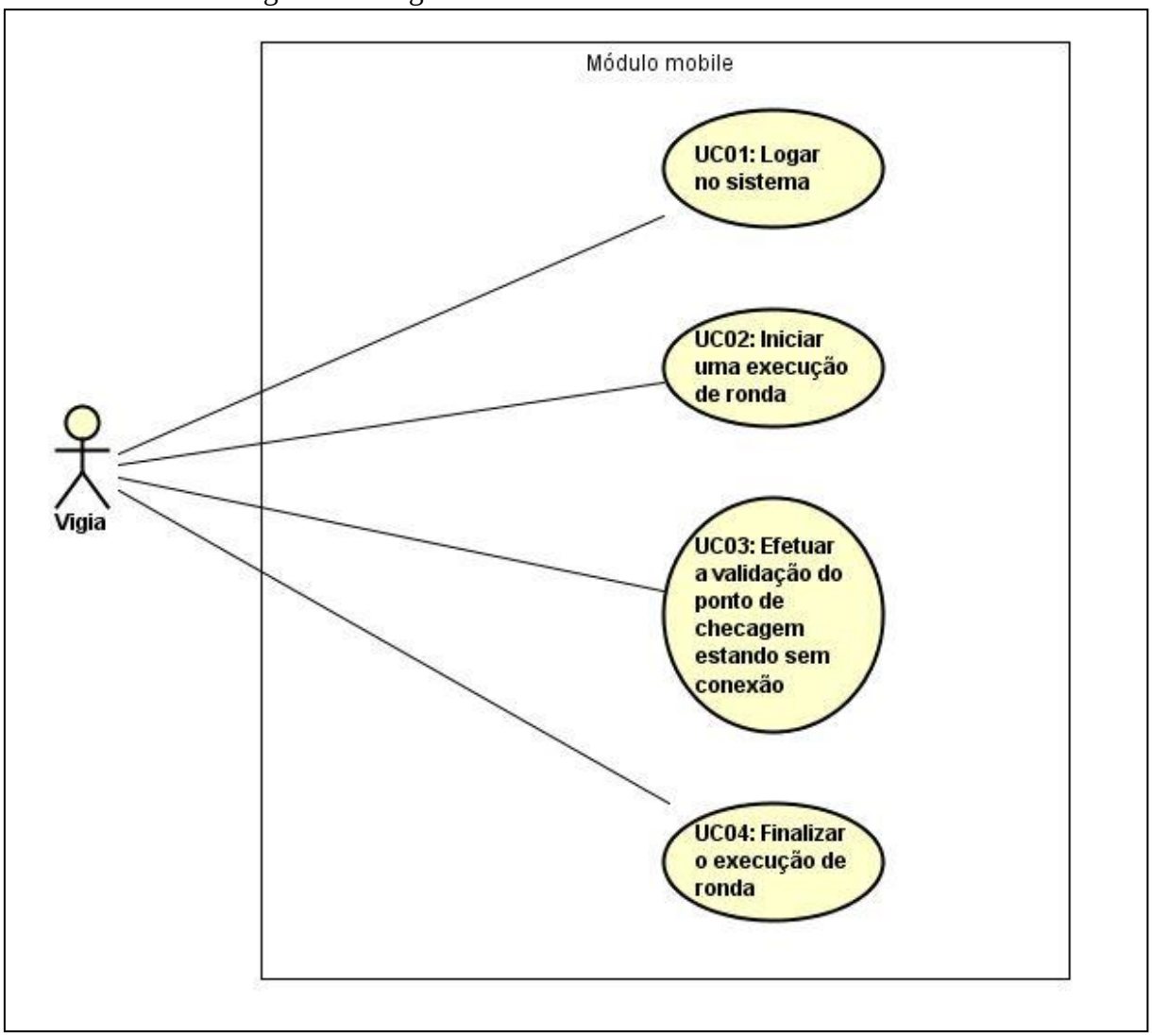
Esta seção apresentará os diagramas de caso de uso e a matriz de rastreabilidade. As especificações dos casos de uso estão apresentadas no Apêndice A.

3.2.1 Diagramas de casos de uso

Esta seção apresenta os casos de uso envolvendo o aplicativo móvel. As descrições dos casos de uso se encontram no Apêndice A.

Como definido por Raulino (2016), apenas administradores do sistema podem ter acesso ao servidor web para realizar os devidos cadastros. Agora, além dos demais cadastros que o administrador realiza, este pode realizar o cadastro do endereço *Media Access Control* (MAC) da rede *wifi* dos *smartphones* que os vigilantes utilizam para o aplicativo móvel para realizar a ronda. O administrador pode decidir no momento do cadastro do vigilante se o sistema irá validar o endereço MAC no momento do *login* do aplicativo. Figura 7 apresenta as novas interações do vigia no aplicativo móvel do sistema.

Figura 7 - Diagrama de casos de uso do módulo mobile



Fonte: do autor.

Na interface móvel do sistema a interação do vigia com o módulo consiste em iniciar uma execução de ronda e no momento em que for realizar a validação do ponto de checagem, não seja necessário que o aparelho de *smartphone* esteja conectado à rede de internet, podendo executar a ronda sem se preocupar com uma possível queda de sinal. O vigia poderá realizar a execução da ronda quase em sua totalidade de maneira off-line, tendo apenas que estar *online*

na leitura do último ponto de checagem da ronda, pois nesse momento as informações da ronda são enviadas ao servidor e gravadas no banco de dados principal.

3.2.2 Matriz de rastreabilidade RF x UC

O Quadro 1 apresenta a matriz de rastreabilidade entre casos de uso da Figura 7 e os requisitos funcionais da seção 3.1.

Quadro 1 - Matriz de rastreabilidade

Requisitos funcionais	**Casos de uso**
RF a: permitir o vigia iniciar uma execução de ronda em uma de suas rotas no aplicativo mobile estando conectado da rede de internet	UC02
RF b: permitir o vigia verificar o atual ponto de checagem que precise ser validado no aplicativo mobile estando desconectado da rede de internet	UC02
RF c: permitir o vigia validar um ponto de checagem ao aproximar o aparelho de uma tag NFC no aplicativo mobile estando desconectado da rede de internet	UC03
RF d: permitir o vigia a encerrar a execução de ronda estando conectado à rede de internet	UC04

Fonte: elabora pelo autor.

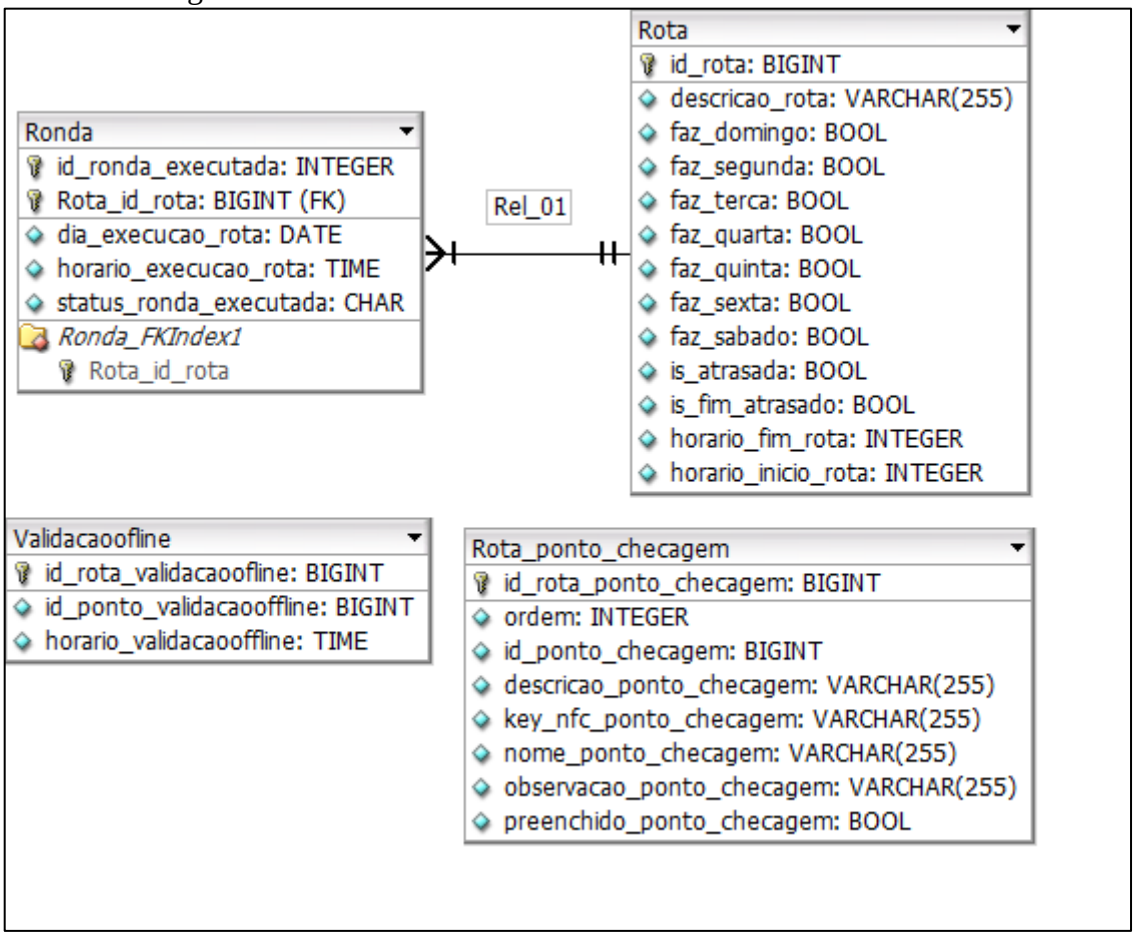
3.2.3 Modelo Entidade e relacionamento módulo móvel e servidor

Nesta seção é apresentado o Modelo Entidade Relacionamento (MER) das tabelas da base de dados embutida no dispositivo móvel que irá gravar as informações das execuções de ronda.

Raulino (2016) já utilizava a base de dados para gravar as informações com objetivo de que o tráfego de informações entre os módulos não fosse sobrecarregado, fazendo com que a troca de dados entre os módulos e também entre as próprias *activities* do Android fossem mais rápido. Entretanto, não eram mantidas essas informações na base de forma que a conexão à rede sempre era exigida.

Para tratar isso, foi adicionado uma nova tabela, que em conjunto com as que já estavam criadas, salvam as informações dos registros de ponto para serem enviadas ao servidor em sua finalização. A Figura 8 mostra o MER utilizado na construção do aplicativo móvel.

Figura 8 - Modelo Entidade Relacionamento do módulo mobile



Fonte: elaborado pelo autor.

O único relacionamento utilizado por Raulino (2016) na base de dados do módulo móvel do sistema foi entre as tabelas rota e ronda, onde a ronda está associada a uma rota. Esse tipo de implementação da base sem quase nenhum relacionamento entre as tabelas ocorre porque a ideia da base de dados no módulo mobile era apenas guardar momentaneamente os registros feitos durante as execuções das rondas, todo o tratamento de consistência dos dados estava sendo tratado pelo servidor. Por esse motivo, está sendo gravado somente as informações de ponto e horário na tabela implementada pela melhoria. Assim que os registros são enviados da tabela de validação *off-line*, é enviado a informações do vigia que executou a ronda. Segue breve descrição das tabelas listadas na figura 8:

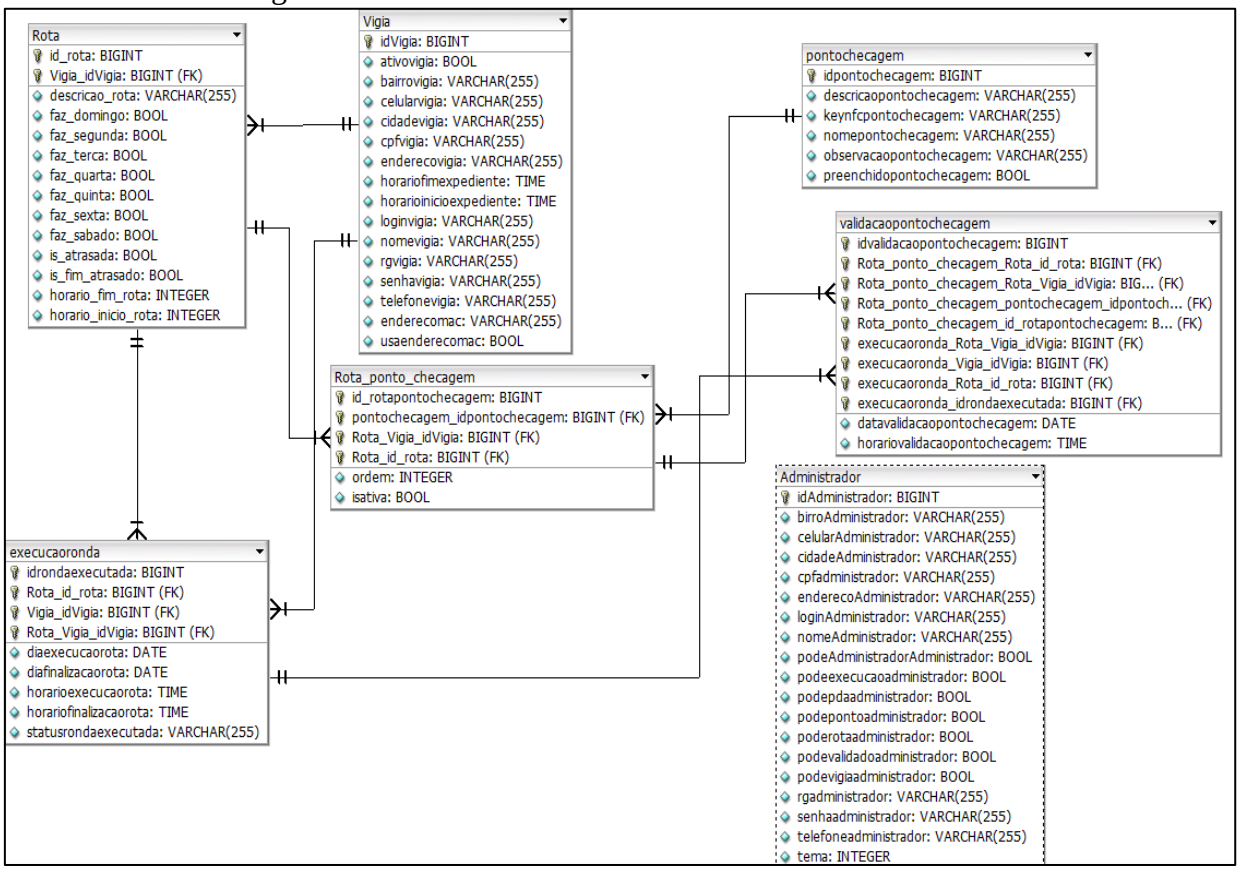
a) Ronda: nesta tabela são gravadas informações sobre a ronda executada pelo vigia, sendo salvos dados como dia, horário e o status da ronda sendo executada;
b) Rota: nesta tabela são guardadas as informações sobre a rota que está sendo executada pela execução da ronda. São gravados os registros de qual dia da semana é realizado essa rota, horário de início e finalização da rota e sua descrição;

c) Rota_ponto_checagem: nesta tabela são gravadas as informações referentes aos pontos de checagem que devem ser validados durante a execução de ronda. Também é guardado a ordem em que o ponto de checagem deve ser validado;

d) Validacaoofline: está é a tabela implementada para que salve as informações dos pontos de checagem quando há falha de conexão com o servidor. Nesta tabela grava-se somente o ponto verificado na validação e o horário em que o ponto foi validado.

Após gravadas as informações na base do módulo móvel, as informações são enviadas ao servidor na finalização da ronda, enviando os registros nas tabelas principais da base de dados do sistema. Na Figura 9 mostra o MER utilizado na base dados do servidor principal.

Figura 9 - Modelo Entidade Relacionamento do servidor



Fonte: elaborado pelo autor.

No momento do envio dos registros da base de dados móvel para o servidor algumas tabelas são preenchidas com as informações. Segue breve descrição das tabelas mostradas na figura acima em que são gravados os dados:

a) execucaoronda: nesta tabela são gravadas as informações gerais da execução da ronda, gravando o dia em que se iniciou e se finalizou, horário de início e horário de término, qual vigia executou a ronda e qual rota foi realizada.

b) validacaopontochecagem: nesta tabela são gravados os detalhes referentes aos pontos validados na execução da ronda. É gravado nessa tabela as informações do horário em que o ponto de checagem foi validado, a data da validação, em qual ronda foi validado e qual o ponto de checagem foi verificado.
c) rotapontochecagem: nesta tabela são gravadas as associações entre os pontos de checagem e as rotas que esses pontos estão associados. Grava-se qual o ponto de checagem, a rota e se o ponto está ativo associado a rota.
d) pontochecagem: nesta tabela são gravadas as informações de cadastro de pontos de checagem no módulo web. Esta tabela guarda a informação se o ponto está ou não já gravado e também a chave NFC que foi gravado no ponto.
e) vigia: nesta tabela são gravadas as informações de cadastro dos vigias que utilizam o módulo móvel do sistema. Foi adicionado nesta tabela os campos de validação de endereço MAC que podem ou não serem cadastradas para os vigias.
f) rota: nesta tabela é gravado as informações do cadastro de rota que serão executadas no momento da ronda pelo vigia a qual é associada. Além de gravar a qual vigia a rota pertence, também é gravado qual dia da semana é executada e qual a hora de sua execução.
g) administrador: nesta tabela ficam gravadas as informações do cadastro de administradores do módulo web do sistema. Foi adicionado a essa tabela o campo TEMA, onde é gravado qual tema do *layout* do módulo web para o administrador.

3.3 IMPLEMENTAÇÃO

Nesta seção serão mostradas as técnicas de implementação das melhorias do sistema e as ferramentas utilizadas.

3.3.1 Técnicas e ferramentas utilizadas

As melhorias foram desenvolvidas e testadas em um computador com o sistema operacional Windows 10. Para desenvolver as melhorias no módulo servidor foi utilizado o Eclipse Jee Oxygen como Integrated Development Environment (IDE). A ferramenta utilizada para acessar o banco de dados do servidor foi o pgAdmin 3. Para executar o sistema para realização de testes foi utilizado o servidor Apache Tomcat v7.0, utilizando o navegador Chrome para verificação do módulo web do sistema.

Já para implementar as melhorias do aplicativo móvel do sistema, a IDE utilizada foi o Android Studio 2.3.1. Para realizar os testes do aplicativo foi utilizado o aparelho celular Motorola Moto G5 Plus com a versão do Android 7.0. Utilizado também duas *tags* NFC Smarttrac para realização das simulações de rondas.

Para a aplicação das melhorias no *back-end* do módulo web foi mantido o padrão de desenvolvimento de projetos MVC utilizando o *framework* Mentawai Framework. Para o *front-end* foi mantido também na parte das interfaces do sistema o padrão Material Design criado pelo Google, utilizando o Material Design Lite (MDL) que já foi utilizado na primeira versão do sistema por Raulino (2016) mas implementado novos *templates* também na interface web. Quando um novo administrador for ser cadastrado, esta nova versão permitirá que seja escolhido um novo tema para a interface web. O Quadro 2 mostra parte do trecho de código onde as opções de temas são implementadas e ficam à disposição do operador para que possam ser escolhidas no cadastro ou alteração dos administradores do módulo web.

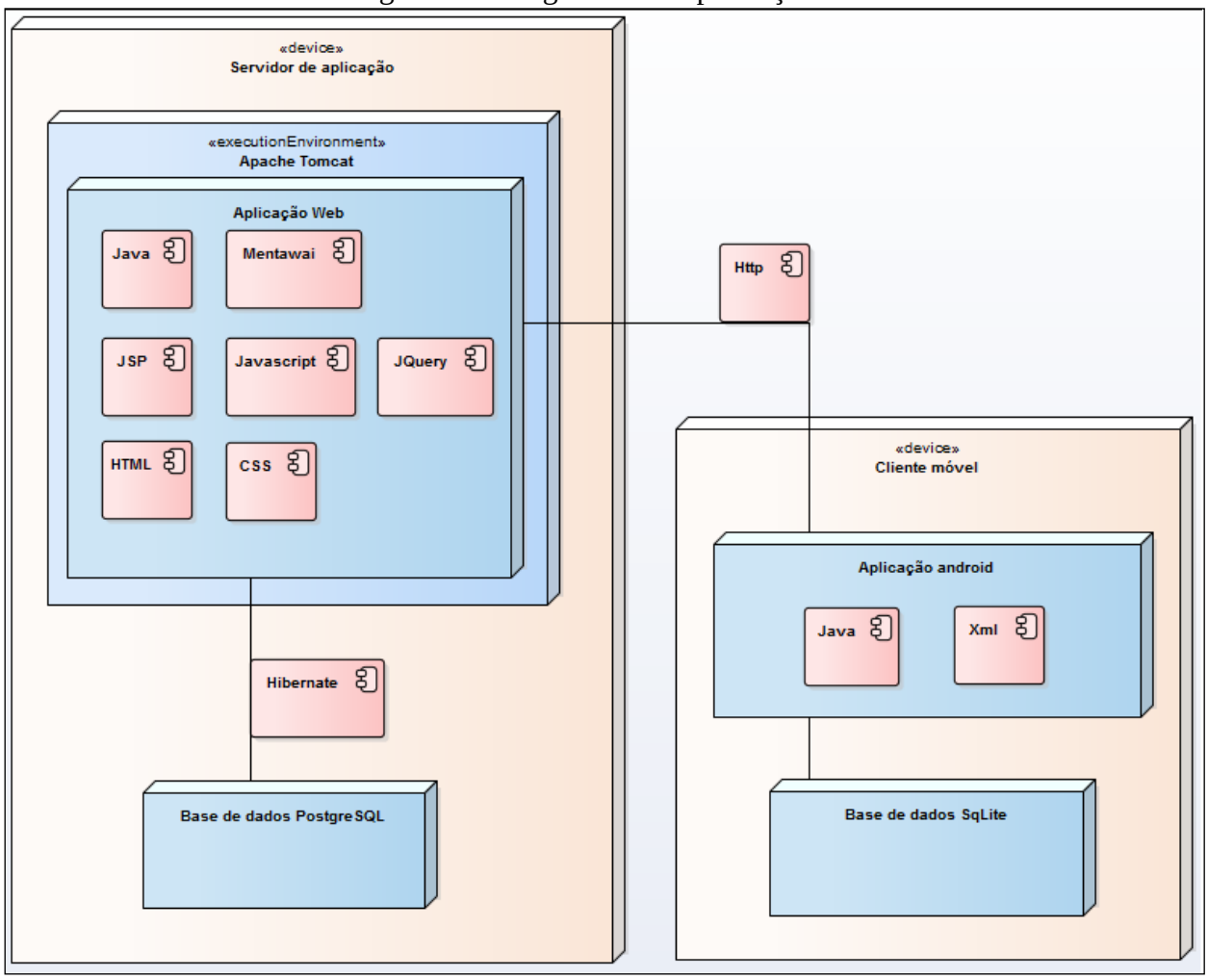
Quadro 2 - Temas do módulo web

```
private SimpleListData temasList(){
    SimpleListData sld = new SimpleListData();
    sld.add(1, "Azul metalico");
    sld.add(2, "Indigo");
    sld.add(3, "Copa do mundo");
    sld.add(4, "Rosa");
    return sld;
}
```

Fonte: do autor.

As melhorias aplicadas no módulo web foram desenvolvidas em Java, utilizando como base de dados o SqLite para gravar as informações registradas no aparelho *smartphone*. Para a conexão entre o aplicativo móvel e o módulo web foram utilizadas mensagens *JavaScript Object Notation* (JSON) enviadas pelo protocolo de comunicação HyperText Transfer Protocol (HTTP). A Figura 10 apresenta a interação entre os módulos do sistema e as tecnologias empregadas.

Figura 10 - Diagrama de implantação



Fonte: Adaptada de Raulino (2016)

Para realizar a gravação dos dados no banco de dados PostgreSQL, foi utilizado o *framework* Hibernate, que é uma ferramenta para mapeamento objeto-relacional, facilitando o mapeamento dos atributos entre a base de dados e os objetos. O Hibernate utiliza para realizar a comunicação um arquivo XML onde se configura as informações gerais do banco de dados, como nome da base, endereço de conexão, *login* e senha para realizar a conexão. O Quadro 3 mostra a estrutura do arquivo XML utilizado pelo Hibernate para conexão.

Quadro 3 - XML de configuração do Hibernate

```xml
<?xml version="1.0" encoding="UTF-8" ?>
  <persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
      version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">
      <persistence-unit name="crudHibernatePU" transaction-type="RESOURCE_LOCAL">
          <provider>org.hibernate.ejb.HibernatePersistence</provider>
          <properties>
              <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect" />
              <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
              <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/TCControleVigias" />
              <property name="javax.persistence.jdbc.user" value="postgres" />
              <property name="javax.persistence.jdbc.password" value="Targus20" />

              <property name="hibernate.show_sql" value="true" />
              <property name="hibernate.format_sql" value="false" />
              <property name="hibernate.use_sql_comments" value="false" />
              <property name="hibernate.jdbc.wrap_result_sets" value="false" />
              <property name="hibernate.hibernate.cache.use_query_cache" value="true" />
              <property name="hibernate.hbm2ddl.auto" value="update" />
          </properties>
      </persistence-unit>
  </persistence>
```

Fonte: do autor.

Com o objetivo de tornar o acesso dos vigias a partir do aplicativo móvel mais seguro e fazer com que o administrador tenha um maior controle dos dispositivos que acessam o aplicativo, foi pensado em controlar esse acesso ao servidor através da informação de endereço MAC da rede *wi-fi* do dispositivo que o vigia está utilizando. Para isso, foram adicionados dois novos atributos a classe Vigia. Para que as alterações feitas na classe Vigia fossem facilmente feitas na base de dados, foi utilizado a biblioteca Hibernate Annotations.

Essa biblioteca permite que, através de anotações nas classes do sistema, seja feita a configuração do banco de dados. Como estava sendo adicionado mais dois campos para a tabela de vigias, foi utilizado a marcação @column que identifica o tipo de valor que deve ser inserido na base. Foram adicionados os campos usaEnderecoMac do tipo boolean e o campo enderecoMac do tipo String. O Quadro 4 mostra a configuração dos dois novos campos adicionados à classe Vigia utilizando o Hibernate Annotations.

Quadro 4 - Estrutura da classe Vigia

```java
@Entity
@Table(name = "vigia")
public class Vigia {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long idVigia;
    @Column
    private String nomeVigia;
    @Column
    private String rgVigia;
    @Column
    private String cpfVigia;
    @Column
    private String enderecoVigia;
    @Column
    private String bairroVigia;
    @Column
    private String cidadeVigia;
    @Column
    private String telefoneVigia;
    @Column
    private String celularVigia;
    @Column
    private String loginVigia;
    @Column
    private String senhaVigia;
    @Column
    private boolean ativoVigia;
    @Column
    private Time horarioInicioExpediente;
    @Column
    private Time horarioFimExpediente;
    @Column
    private boolean usaEnderecoMac;
    @Column
    private String enderecoMac;
```

Fonte: do autor.

A partir disso, o administrador irá definir, no momento em que efetuar o cadastro do vigia, se o mesmo irá ser controlado pelo endereço MAC da rede *wi-fi* no momento em que efetuar o *login* no aplicativo móvel. O administrador poderá alterar esse cadastro a qualquer momento, para qualquer vigia, uma vez que possui as permissões necessárias.

Caso o cadastro do vigia estiver selecionado para utilizar o endereço MAC, no momento em que efetuar o login no aplicativo, e enviar para o servidor as informações de usuário e senha, também será enviado o endereço MAC de seu dispositivo e no servidor será verificado se o endereço cadastrado para o vigia é o mesmo que foi enviado pelo aplicativo no momento do *login*. O Quadro 5 mostra a validação do MAC no servidor no momento do *login*.

Quadro 5 - Validação do endereço MAC

```
    if(vigia.isUsaEnderecoMac() && !vigia.getEnderecoMac().equals(mac)) {
        output.setValue("resultLogin", "macErrado");
    }else {
        output.setValue("resultLogin", vigia.getDTO());
        output.setValue("roleLogin", "vigia");
    }
} catch (Exception e) {
    AdministradorJpaDAO administradorDAO = AdministradorJpaDAO.getInstance();
    try {
        Administrador administrador = administradorDAO.getByLogin(username, password);

        output.setValue("resultLogin", administrador);
        output.setValue("roleLogin", "administrador");
    } catch (Exception e2) {
        output.setValue("resultLogin", "false");
    }
```

Fonte: do autor.

O Quadro 6 mostra o método utilizado para obter a informação do endereço MAC do *smartphone* do vigia no Android Studio.

Quadro 6 - Método getMac()

```
private static String getMac(){
    try {
        List<NetworkInterface> all = Collections.list(NetworkInterface.getNetworkInterfaces());
        for(NetworkInterface nif : all){
            if(!nif.getName().equalsIgnoreCase("wlan0")) continue;

            byte[] macBytes = nif.getHardwareAddress();
            if(macBytes == null){
                return "";
            }

            StringBuilder res1 = new StringBuilder();
            for(byte b : macBytes){
                res1.append(Integer.toHexString(b & 0xFF)+ ":");
            }

            if(res1.length() > 0){
                res1.deleteCharAt(res1.length() - 1);
            }
            return res1.toString();
        }
     }catch (Exception ex){
```

Fonte: do autor.

Caso o endereço MAC não seja igual ao que foi cadastrado para o vigia e o do aparelho que ele está realizado, o servidor retorna para o aplicativo, que faz a tratativa conforme a resposta. Se for o MAC igual ao cadastrado, permite realizar o *login* e caso não seja igual uma mensagem é enviada ao vigia informado que ele não pode acessar o aplicativo utilizando o atual

dispositivo. O Quadro 7 mostra a validação do MAC no aplicativo móvel, permitindo ou não o *login* do vigia.

Quadro 7 - Validação MAC no módulo móvel

```
    if(!result.equals("false") && !result.equals("macErrado")){
        role = returnObject.getString("roleLogin");
    }
} catch (JSONException e) {
    result = "exception";
}

pdLoading.dismiss();
if (result.equalsIgnoreCase("false")) {
    new AlertDialog.Builder(MainActivity.this)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setTitle("Erro")
            .setMessage("Login ou senha são invalidos")
            .setPositiveButton("Ok",null)
            .show();

}else if (result.equalsIgnoreCase("macErrado")) {
    new AlertDialog.Builder(MainActivity.this)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setTitle("Erro")
            .setMessage("Este dispositivo não pode ser usado por este vigia")
            .setPositiveButton("Ok",null)
            .show();
```

Fonte: do autor.

Para que fosse possível realizar as execuções de ronda, o vigia antes não poderia perder a conexão com a rede de internet. Para corrigir esse processo, foi utilizado uma base de dados através do SQLite que auxilia na implantação de uma base de dados embutida, lendo e escrevendo diretamente no banco de dados, sendo geralmente usado para pequenas aplicações. Assim os dados dos pontos de checagem que são lidos durante a execução da ronda são salvos, sem ocorrer a perda dos dados. Foi utilizado o SQLiteOpenHelper onde é definido as configurações da base de dados como sua versão, nome da base e onde é feito a definição das tabelas que serão criadas no banco. Além disso, também é possível manipular os dados gravados nas tabelas do SQLite, podendo inserir, alterar e apagar os dados, conforme for necessário. O Quadro 8 mostra a configuração da tabela *table_validacaoofline* utilizada na execução de ronda sem a conexão à internet.

Quadro 8 - Estrutura da tabela de validação off-line

```
private static final String CREATE_TABLE_VALIDACAOOFFLINE = "CREATE TABLE " + TABLE_VALIDACAOOFLINE
        +"("+
        ID_ROTA_VALIDACAOOFFLINE +" bigint, " +
        ID_PONTO_VALIDACAOOFFLINE +" bigint, " +
        HORARIO_VALIDACAOOFFLINE +" time without time zone)";

public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {

    // creating required tables
    db.execSQL(CREATE_TABLE_VALIDACAOOFFLINE);
```

Fonte: do autor.

Nesta tabela é salvo o *id* da rota que o vigia está executando no momento da execução da ronda, assim como é salvo também o *id* do ponto de checagem do ponto que foi lido e não foi enviado para o servidor e o horário em que foi lido o ponto de checagem. Para realizar as operações de gravação e leitura nessa nova tabela foi criado um *Data Access Object* (DAO). A classe DAO da validação *off-line* é utilizada no momento da validação de conexão com o servidor. O Quadro 9 mostra a estrutura da classe ValidacaoOfflineDAO.

Quadro 9 - Estrutura da classe de validação off-line

```
public class ValidacaoOfflineDAO {

    private static final String TABLE_VALIDACAOOFLINE = "validacaoOffline";

    private static final String ID_ROTA_VALIDACAOOFFLINE = "idrotavalidacaooffline";
    private static final String ID_PONTO_VALIDACAOOFFLINE  = "idpontovalidacaoofline";
    private static final String HORARIO_VALIDACAOOFFLINE = "horariovalidacaoofline";
```

Fonte: do autor.

Inicialmente, o sistema tenta lançar os dados coletados para o servidor através de um protocolo HTTP, caso a conexão retorne como sem sucesso com o servidor, é criado um objeto da classe ValidacaoOfflineDAO. O Quadro 10 mostra a validação de conexão sem sucesso com o servidor e a criação do objeto da classe ValidacaoOfflineDAO.

Quadro 10 - Validação da conexão

```
if (result.equalsIgnoreCase("exception") || result.equalsIgnoreCase("unsuccessful")) {

    Toast.makeText(RondaActivity.this, "Problema na conexão, salvando validação para ser enviada na finalização.

    ValidacaoOfflineDAO dao = new ValidacaoOfflineDAO();

    dao.createValidacaoOffline(getApplicationContext(), idPontoSuporte);
} else {
```

Fonte: do autor.

Após criar o objeto, é chamado o método createValidacaoOffline. Esse método é o responsável em inserir os registros coletados na execução de ronda que não foram enviadas para o servidor. O Quadro 11 apresenta os detalhes da inserção de dados no banco de dados do SQLite.

Quadro 11 - Inserção de registros na base de dados

```
public long createValidacaoOffline(Context context,  String idPontoValidacaoOffline) {
    SQLiteDatabase db = new DatabaseHelper(context).getWritableDatabase();

    DateFormat df = new SimpleDateFormat("yyMMddHHmmss");

    ContentValues values = new ContentValues();
    values.put(ID_PONTO_VALIDACAOOFFLINE, idPontoValidacaoOffline);
    values.put(HORARIO_VALIDACAOOFFLINE, df.format(new Date()));

    long todo_id = db.insert(TABLE_VALIDACAOOFLINE, null, values);

    return todo_id;
}
```

Fonte: do autor.

Após a validação do ponto de checagem e da inserção dos dados na base de dados do dispositivo, é incrementado o contador de ponto atual. Essa variável foi criada por Raulino (2016) para realizar o controle das ordens de leitura dos pontos de checagem. Essa variável é iniciada em zero e, conforme o vigia valida os pontos de checagem, vai sendo incrementada gradualmente e é zerada novamente quando a execução da rota é finalizada. Além de ter sido criada para controlar a ordem de leitura dos pontos, essa variável, em conjunto com a base de dados que é criada no dispositivo, evita que a cada validação de ponto, os dados sejam enviados ao servidor para que ele controle a ordem das checagens. Assim, as informações da ordem a ser executada a rota são gravadas no próprio dispositivo. Essa função é de extrema importância para a melhoria implantada ao módulo móvel do sistema pois através desse contador o vigia obtém a informação do último ponto de checagem a ser verificado na execução da ronda. Como

o vigia deve estar obrigatoriamente conectado à rede na validação do último ponto de checagem. O vigia ao perceber que é o último ponto a ser validado ainda não está conectado, tem a opção de se conectar e validar o ponto. Quando o vigia ler o último ponto, o sistema irá validar se é o último ponto da ronda. Caso for o último, o sistema cria uma lista do tipo ValidacaoOfflineDAO, onde é jogado todos os pontos checados que estão na base de dados do dispositivo. O Quadro 12 mostra a validação do contador para saber sé o último ponto e a criação da lista.

Quadro 12 - Validação do contador

```
}
contadorValidacoes++;
if (contadorValidacoes == contadorValidacoesTotal) {
    ValidacaoOfflineDAO dao = new ValidacaoOfflineDAO();

    List<ValidacaoOffline> listValidacao = dao.getPontos(getApplicationContext());
```

Fonte: do autor.

A lista é preenchida com as informações dos pontos através do método getPontos. Esse método utiliza um comando na base de dados, pegando todas as informações gravadas na base do dispositivo e jogando em um *ArrayList*. Após gravar todos os registros, o método retorna o *ArrayList* com os dados. O Quadro 13 mostra a estrutura do método getPontos.

Quadro 13 - Estrutura do método getPontos

```
public List<ValidacaoOffline> getPontos(Context context) {
    List<ValidacaoOffline> validacoes = new ArrayList<~>();
    String selectQuery = "SELECT  * FROM " + TABLE_VALIDACAOOFLINE;

    Log.e("DatabaseHelper", selectQuery);

    SQLiteDatabase db = new DatabaseHelper(context).getReadableDatabase();
    Cursor c = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (c.moveToFirst()) {
        do {
            ValidacaoOffline vc = new ValidacaoOffline();
            vc.setIdPontoValidacaoOffline(c.getString(c.getColumnIndex(ID_PONTO_VALIDACAOOFFLINE)));
            vc.setHorarioValidacaoOffline(c.getString(c.getColumnIndex(HORARIO_VALIDACAOOFFLINE)));

            validacoes.add(vc);
        } while (c.moveToNext());
    }

    return validacoes;
}
```

Fonte: do autor.

Com o retorno da lista com as informações, a lista é submetida a um laço de repetição onde passa por todos os registros da lista e concatena esses registros em uma variável do tipo

String que é utilizada para fazer esse controle. Em seguida é chamado o método deleteAll que apaga as informações da execução de ronda que foram gravadas no dispositivo e também é chamado o método de finalização da ronda. O Quadro 14 mostra o código correspondente e este controle.

Quadro 14 - Laço de repetição para varrer os registros

```
    for (ValidacaoOffline val : listValidacao){
        offvalString += val.getIdPontoValidacaoOffline()+"Horario"+val.getHorarioValidacaoOffline()+"proximo";
    }

    offvalString += "off";

    dao.deleteAll(getApplicationContext());

    new AsyncFinalizaRonda().execute(rondaAtual.getIdRondaExecutada() + "", offvalString);
}
```

Fonte: do autor.

Na chamada do método AsyncFinalizaRonda, os dados são enviados para o servidor via HTTP. O Quadro 15 mostra os detalhes da tentativa de conexão com o servidor via HTTP.

Quadro 15 - Tentativa de conexão via HTTP

```
try {
    // Setup HttpURLConnection class to send and receive data from php and mysql
    conn = (HttpURLConnection) url.openConnection();
    conn.setReadTimeout(READ_TIMEOUT);
    conn.setConnectTimeout(CONNECTION_TIMEOUT);
    conn.setRequestMethod("POST");

    // setDoInput and setDoOutput method depict handling of both send and receive
    conn.setDoInput(true);
    conn.setDoOutput(true);

    // Append parameters to URL
    Uri.Builder builder = new Uri.Builder()
            .appendQueryParameter("codigoRondaFinalizar", params[0])
            .appendQueryParameter("posProcessa", params[1]);

    String query = builder.build().getEncodedQuery();

    // Open connection for sending data
    OutputStream os = conn.getOutputStream();
    BufferedWriter writer = new BufferedWriter(
            new OutputStreamWriter(os, "UTF-8"));
    writer.write(query);
    writer.flush();
    writer.close();
    os.close();
    conn.connect();
```

Fonte: do autor.

Após a tentativa de conexão com o servidor, o dispositivo recebe a mensagem do servidor informando se a conexão foi feita com sucesso ou se não foi possível realizar a

conexão. O Quadro 16 mostra a validação da mensagem recebida pelo servidor se a conexão foi feita com sucesso.

Quadro 16 - Validação da mensagem de retorno do servidor

```
try {

    int response_code = conn.getResponseCode();

    // Check if successful connection made
    if (response_code == HttpURLConnection.HTTP_OK) {

        // Read data sent from server
        InputStream input = conn.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        StringBuilder result = new StringBuilder();
        String line;

        while ((line = reader.readLine()) != null)
            result.append(line);

        try {
            JSONObject returnObject = new JSONObject(result.toString());
            return (returnObject.getString("result"));//vem do json do servidor
        } catch (JSONException e) {
            e.printStackTrace();
            return "exception";
        }

    } else {

        return ("unsuccessful");
    }
```

Fonte: do autor.

Caso o retorno do servidor for de sucesso, uma mensagem é apresentada ao vigia informando que a ronda foi finalizada. O Quadro 17 mostra o tratamento da mensagem vinda do servidor para ser mostrada ao usuário.

Quadro 17 - Mensagem apresentada ao vigia

```
protected void onPostExecute(String result) {

    pdLoading.dismiss();

    if (result.equalsIgnoreCase("exception") || result.equalsIgnoreCase("unsuccessful")) {

        Toast.makeText(RondaActivity.this, "Problema na conexão.", Toast.LENGTH_LONG).show();

    } else {

        AlertDialog.Builder builder = new AlertDialog.Builder(RondaActivity.this);
        builder.setTitle("Ronda finalizada com sucesso.");
        // Set up the buttons
        builder.setPositiveButton("OK", (dialog, which) -> {
                Intent intent = new Intent(RondaActivity.this, MainMenuVigiaActivity.class);
                startActivity(intent);
                RondaActivity.this.finish();
        });
        builder.show();
    }
}
```

Fonte: do autor.

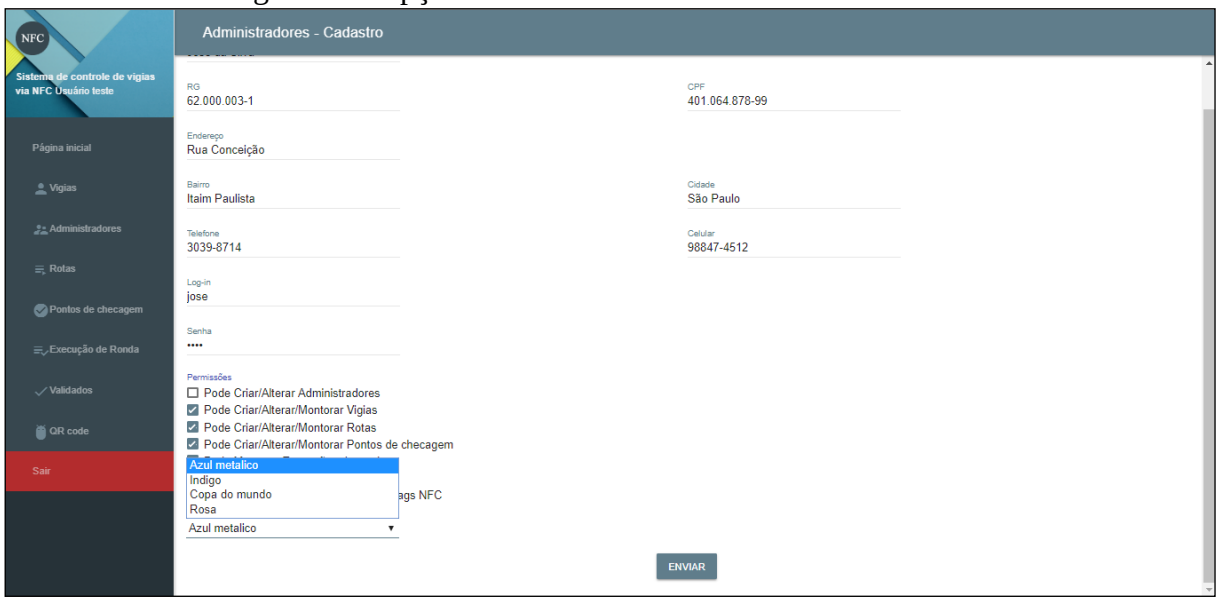
3.3.2 Operacionalidade da Implantação

Nesta seção será apresentada as melhorias implementadas no módulo web e no módulo móvel do sistema. A seção será dividida em duas partes. A seção 3.3.2.1 irá mostrar as implementações realizadas na parte web e a seção 3.3.2.2 irá mostrar as implementações feitas no aplicativo móvel do sistema.

3.3.2.1 Novas funcionalidades do servidor WEB

Uma nova funcionalidade incorporada ao módulo web do sistema foi a opção de escolha de tema de cores da interface web no cadastro de admiradores do sistema. São apresentadas 4 opções de temas, que são: Azul metálico, Indigo, Copa do Mundo e Rosa. O tema pode ser alterado a qualquer momento, mas apenas pode ser alterado por administradores que possuam permissão de criar e alterar novos administradores. A Figura 11 mostra a opção de escolha de tema para o novo administrador durante o cadastro.

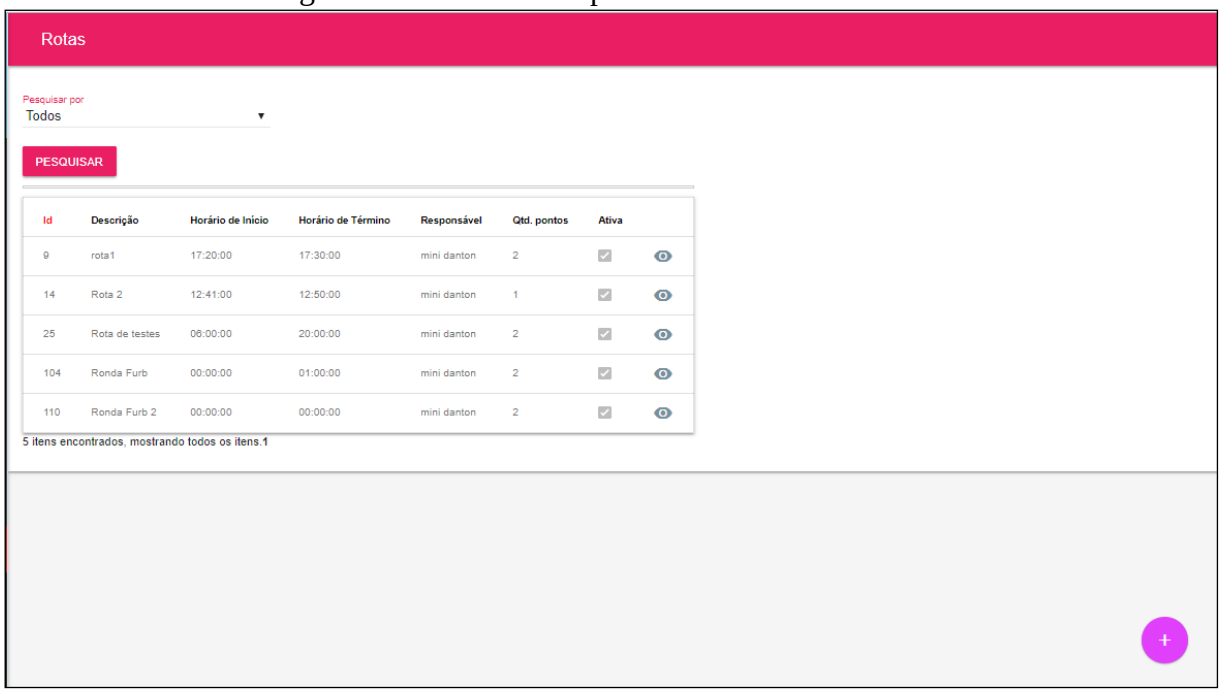
Figura 11 - Opção de temas no cadastro de administradores



Fonte: do autor.

Assim que um novo administrador realizar o *login* no sistema, será apresentado o tema escolhido durante o cadastro. A Figura 12 mostra o tema escolhido aplicado a um administrador cadastrado.

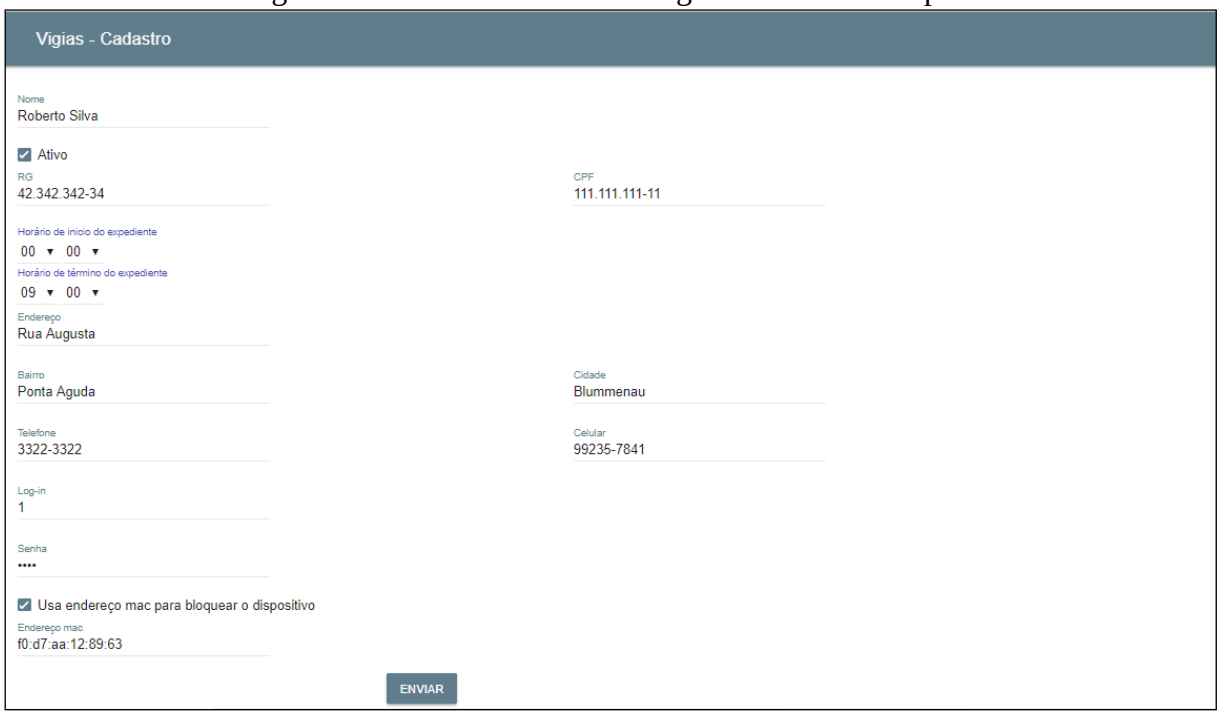
Figura 12 - Tema novo aplicado a um administrador



Fonte: do autor.

Também foi adicionado um campo de cadastro na tela de cadastro de vigias. Durante o cadastro do vigia, o administrador pode escolher se no momento do *login* no aplicativo será validade o endereço MAC clicando em um *checkbox*. Se estiver selecionado o sistema irá validar o MAC do dispositivo do vigia, se não estiver, não haverá a validação. Além do *checkbox* que faz o controle da validação, há o campo onde se informa o endereço MAC do dispositivo. A Figura 13 mostra a tela de cadastros de vigias com a adição nos novos campos.

Figura 13 - Tela de cadastro de vigia com novos campos

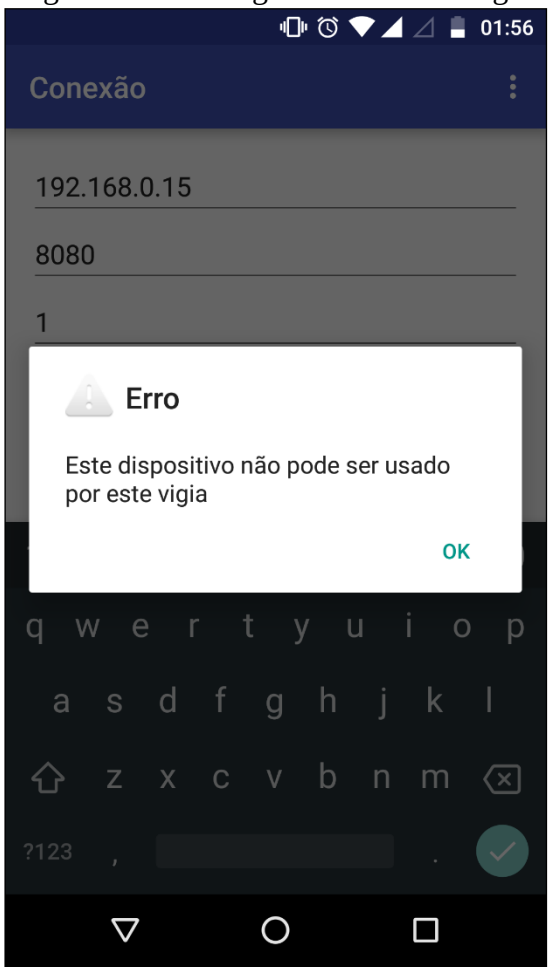


Fonte: do autor.

3.3.2.2 Novas funcionalidades do dispositivo móvel

Entre as melhorias criadas para o módulo móvel do sistema foi o controle de acesso do aplicativo pelo endereço MAC do dispositivo. O vigia não precisará informar este dado no momento do *login*, apenas precisa informar seu login de acesso e senha que o próprio sistema trata de enviar a informação do endereço MAC para o servidor para ser validado. Caso o vigia tenha em seu cadastro a opção de controlar o endereço MAC no momento do acesso ao aplicativo, e tentar efetuar *login* em um dispositivo que não está cadastrado para ele no servidor, uma mensagem é apresentada ao vigia informando que ele não pode efetuar o *login* pois ele não pode utilizar o dispositivo. A Figura 14 mostra a mensagem de aviso ao vigia caso acesse de um dispositivo diferente do que foi cadastrado no servidor.

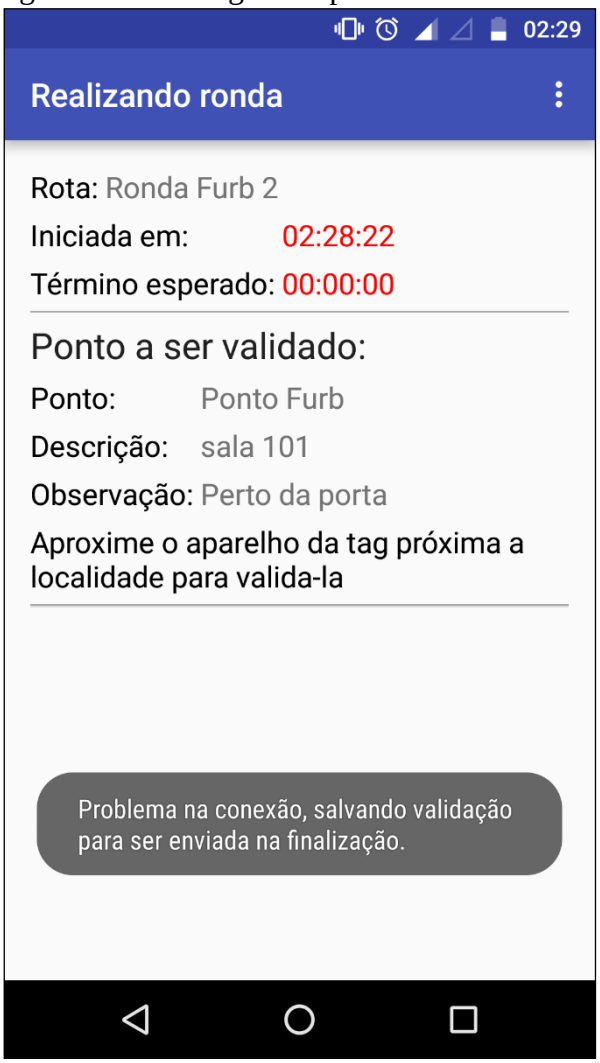
Figura 14 - Mensagem de aviso ao vigia



Fonte: do autor.

Já quando o vigia conseguir efetuar o *login*, ele poderá efetuar o a execução de uma ronda atribuída para ele. Ao estar executando a ronda e o aplicativo não conseguir enviar as informações para o servidor, uma mensagem é mostrada ao usuário informando que houve um problema na conexão com o servidor e que está sendo salvo as informações da validação para ser enviado na finalização da ronda. A Figura 15 mostra a mensagem do aplicativo para o vigia informando o problema na conexão e o salvamento da validação.

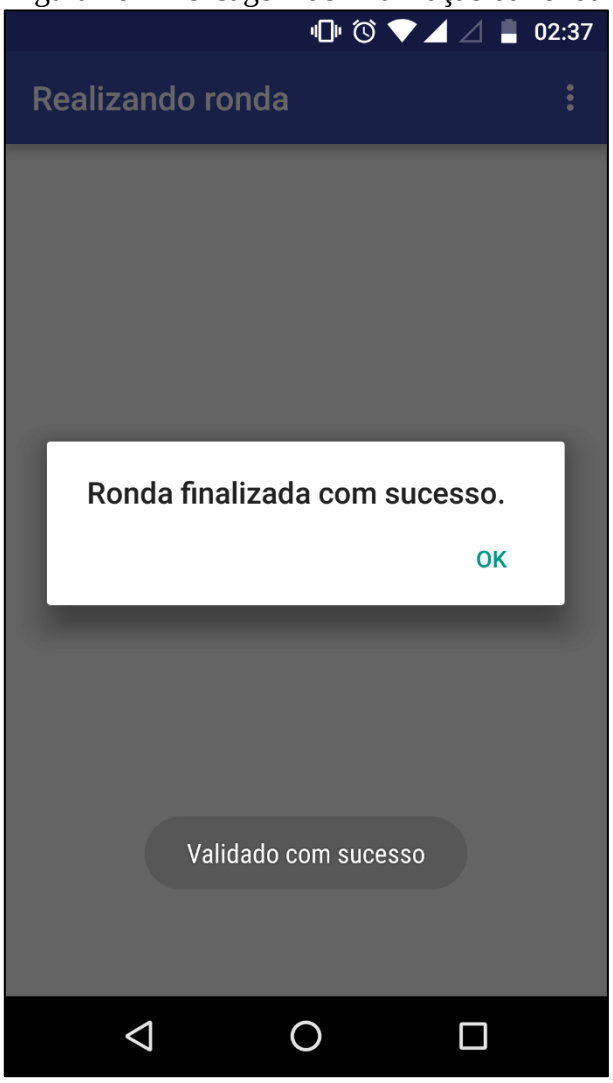
Figura 15 - Mensagem de problema na conexão



Fonte: do autor.

Quando o vigia efetuar então a validação do último ponto de checagem da ronda, ele deve estar conectado à rede. Assim, as informações são enviadas ao servidor e é mostrado a mensagem ao vigia informando que a ronda foi finalizada e que o ponto foi validado com sucesso. A Figura 16 mostra a mensagem apresentada ao vigia no momento da finalização da ronda.

Figura 16 - Mensagem de finalização da ronda



Fonte: do autor.

3.4 RESULTADOS E DISCUSSÕES

As melhorias desenvolvidas no sistema contemplam todos os requisitos funcionais e todos os requisitos não funcionais definidos no trabalho. Os testes foram realizados pelo autor e por Raulino (2016), tendo sido positivos os resultados apurados. As ferramentas utilizadas funcionaram corretamente. Entretanto, como foi testado em um ambiente controlado, não se pode afirmar como o módulo móvel se comportaria em um ambiente mais conturbado onde poderia haver maiores quedas de sinal e problemas na conexão.

Observando a performance do aplicativo móvel, não foi encontrado nenhuma queda significante que impactasse seu uso, apresentado uma performance aceitável durante os testes realizados. Porém, deve-se ressaltar que a conexão entre o aplicativo móvel e o servidor deve

ocorrer no momento da última validação de ponto de checagem da ronda, sendo que a conexão tenha que ser na mesma rede do servidor e que a coxão seja consistente para que as informações possam trafegar entra as aplicações sem perda de dados.

Em comparação com os trabalhos correlatos, pode-se observar que o sistema de segurança predial desenvolvido por Krüguer (2002) e o sistema de segurança com Linux embarcado desenvolvido por Baumann (2008) utilizam a internet para fazer a comunicação de sensores e câmeras com os sistemas desenvolvidos o que torna o processo de comunicação mais simples e fácil. É importante atentar ao fato de que sem a internet a comunicação não ocorre. Esse caso não afeta ao sistema TopRonda, que não necessita de internet para validar os pontos de checagem e com as melhorias implantadas neste trabalho, o protótipo também não necessita mais de uma conexão à rede para que funcione corretamente durante grande parte de sua execução, sendo necessário apenas no momento da finalização da ronda. Outra característica comparada é a possibilidade de controle da utilização do hardware entre os sistemas. Em todos os trabalhos correlatos é necessário a utilização de algum hardware, porém eles não controlam o acesso dos hardwares por usuários específicos, já o trabalho desenvolvido controla esse acesso. O Quadro 18 mostra um comparativo entre os trabalhos correlatos apresentados e as melhorias implementadas ao sistema.

Quadro 18 - Comparativo entre os trabalhos correlatos

Características/Trabalhos	Sistema Desenvolvido	TopRonda	Sistema de segurança predial	Sistema de segurança com Linux embarcado
Necessidade de hardware	Sim	Sim	Sim	Sim
Controle de Hardware	Sim	Não	Não	Não
Uso da internet	Não	Não	Sim	Sim

Fonte: elaborado pelo autor.

4 CONCLUSÕES

O principal objetivo do trabalho que era apresentar alternativas para a realização de conexão entre os módulos do protótipo, de forma que não seja necessária conexão constante à internet, foi alcançado, proporcionando um grande desafio para o seu cumprimento. Como a tecnologia do SQLite não era conhecida pelo autor, foi necessário um estudo para levantar as funcionalidades dela para observar se atenderia as necessidades apresentadas.

As ferramentas utilizadas na implementação das melhorias se mostraram satisfatórias em todas as etapas do desenvolvimento e se adequaram bem as necessidades que foram encontradas durante a implementação. Ao longo da implementação, os módulos foram sendo testados e ajustados, onde o resultado obtido atendeu as expectativas iniciais.

Os testes realizados com o aplicativo móvel não apontaram de que houvera perda de performance após as novas funcionalidades. Para estes testes contou-se com a colaboração de Raulino (2016) que paralelamente utilizou uma versão do módulo desenvolvido por ele.

Após a finalização do trabalho, concluiu-se que com as melhorias implementadas tornaram o sistema de controle de rota para vigilantes um software com potencial para a área de segurança pois apresenta características que diferenciam de outros sistemas de controle de ronda oferecidos no mercado atualmente, uma vez que se mostra uma solução com melhor custo benefício e uma melhor interface para o controlador. Além disso, acredita-se que sejam possíveis novas melhorias de forma a torná-lo atraente ao segmento de controle de segurança por empresas do ramo.

4.1 EXTENSÕES

As sugestões de melhorias para esse trabalho são enumeradas abaixo:

a) apresentar uma alternativa para que não seja necessário estar conectado à rede no momento da validação do último ponto de checagem da ronda;
b) criar controle de filiais para os vigias cadastrados, aumentando assim o nível de controle do sistema;
c) aplicar os conceitos de rede mesh para o envio de informações entre o módulo móvel o servidor.

REFERÊNCIAS

BAUMAN, Zygmunt. **Comunidade:** A busca por segurança no mundo atual. Rio de Janeiro: Zahar, 2003.

BAUMANN, Daniel. **Protótipo de um Sistema de Segurança Residencial com Linux Embarcado**. 2008. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CARDIA, Nancy; ADORNO, Sérgio Adorno; Poleto, Frederico. **Homicídio e violação de direitos humanos em São Paulo**, 2003. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-40142003000100004>. Acesso em: 16 jul. 2018.

CRUZ, Aparecido. **Tecnologia de atendimento em recepção e portaria**, 2009. Disponível em: < http://www.dominiopublico.gov.br/download/texto/ea000750.pdf>. Acesso em 13 set. 2017.

JANES, Ricardo. **Estudo sobre sistemas de segurança em instalações elétricas automatizadas**. 2009. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3143/tde-29062009-181507/publico/Dissertacao_Ricardo_Janes.pdf>. Acesso em: 07 set. 2017.

KRÜGER, Erasmo. **Protótipo de sistema de segurança predial através de monitoramento utilizando recursos da internet**. 2002. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

LOPES, Liliane Roquete. **Segurança Pública:** Uma questão social, legal e de polícia. 2006. Disponível em: <http://www.atenas.edu.br/Faculdade/arquivos/NucleoIniciacaoCiencia/REVISTAJURI2006/9.pdf>. Acesso em: 25 out. 2017.

LUZ, Gilberto Barbosa da; KUIAWINSKI, Darci Luíz. **Mecanização, Autonomação e Automação – Uma Revisão Conceitual e Crítica,** 2006. Disponível em: <http://www.simpep.feb.unesp.br/anais/anais_13/artigos/1210.pdf>. Acesso em 12 set. 2017.

MARTINS, Walterlando Guimarães. **Segurança Pública e Direitos Humanos**, 2009. Disponível em: <http://br.monografias.com/trabalhos-pdf/seguranca-publica-direitos-humanos/seguranca-publica-direitos-humanos.pdf>. Acesso em 12 set. 2017.

NASSAR, Victor; HORN, Vieira Milton Luiz. **A Internet das coisas com as tecnologias RFID e NFC** São Paulo: Blucjer, 2014.

NFC Forum. **About the Technology**. 2017. Disponível em: <https://nfc-forum.org/what-is-nfc/about-the-technology/>. Acesso em: 11 set. 2017.

ORITZ, C. Enrique. **An Introduction to Near-Field Communication and the Contactless Communication API**, 2008. Disponível em: <http://www.oracle.com/technetwork/articles/javame/nfc-140183.html>. Acesso em: 13 set. 2017.

PRIOSTE, Erasmo. **A prevenção é a melhor precaução**, 2014. Disponível em: <http://www.de-seguranca.com.br/a-prevencao-e-a-melhor-precaucao/>. Acesso em: 07 set. 2017.

RAULINO, André Felipe. **Controle de Rota para Vigilantes Utilizando NFC para Validação de Presença**. 2016. Trabalho de conclusão de curso (Bacharelado em Sistemas de Informação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SARDEIRO, Paulo. **Uma Visão Cósmica da Vida**. São Paulo: Editora Baraúna, 2013.

TOPDATA. **Sistema para controle de rondas.** Disponível em: <https://www.topdata.com.br/>. Acesso em: 21 ago. 2017.

APÊNDICE A – Descrição dos Casos de Uso

Nos quadros abaixo são descritos os principais casos de uso apresentados no diagrama da seção 3.2.1.

No Quadro 19 é apresentado o caso de uso Logar no sistema do módulo mobile.

Quadro 19 - Descrição do caso de uso UC01

Caso de uso	UC1 – Logar no sistema
Descrição	Utilizar as credenciais de acesso cadastradas no servidor para iniciar uma sessão no aplicativo.
Ator	Vigia.
Pré-condição	Vigia deve estar devidamente cadastrado na base de dados.
Fluxo principal	a) O vigia informa seu usuário e senha; b) O sistema valida as informações passadas c) O sistema verifica se o usuário deve ser controlado por endereço MAC; d) O sistema verifica que o usuário é controlado por endereço MAC e que foi enviado o MAC correto no momento do login; e) Sistema redireciona o vigia para a tela de inicial do aplicativo.
Fluxo alternativo	a) O vigia é controlado por endereço MAC mas o endereço enviado para a validação é diferente do informado em seu cadastro; b) Alerta com a mensagem de que o usuário não pode utilizar o aparelho atual é apresentado ao usuário.
Pós condição	O vigia inicia uma sessão no aplicativo móvel sem problemas.

Fonte: elaborado pelo autor.

No Quadro 20 é apresentado o caso de uso Efetuar a validação do ponto de checagem estando sem conexão do módulo mobile.

Quadro 20 - Descrição do caso de uso UC03

Caso de uso	UC3 – Efetuar a validação do ponto de checagem estando sem conexão.
Descrição	Efetuar a validação de um ponto de checagem sem conexão à rede.
Ator	Vigia.
Pré-condição	Vigia deve possuir alguma ronda para ser executada.
Fluxo principal	a) Vigia efetua a validação de um ponto de checagem sem estar conectado à rede; b) O sistema verifica que não há conexão e grava as informações na base de dados do aparelho.
Fluxo alternativo	a) Vigia efetua a validação de um ponto de checagem estando conectado à rede; b) O sistema verifica que há conexão disponível e envia as informações do ponto de checagem para o servidor.
Pós condição	O ponto e validado normalmente pelo aplicativo.

Fonte: elaborado pelo autor