CENTRO UNIVERSITÁRIO UNIFACVEST
CIÊNCIA DA COMPUTAÇÃO
GUSTAVO MARIN SUPPI

FIREHUNTER E FIREWORKER: TECNOLOGIA COMO INSTRUMENTO DE AUXÍLIO NO SALVAMENTO DE VIDAS

LAGES
2016

GUSTAVO MARIN SUPPI

FIREHUNTER E FIREWORKER: TECNOLOGIA COMO INSTRUMENTO DE AUXÍLIO NO SALVAMENTO DE VIDAS

Trabalho de conclusão de curso apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. ESP. Igor Muzeka

Coorientadores:
Profa. DRA. Ingrid Solange Sepúlveda Muñoz
Prof. ME. Márcio José Sembay

LAGES

2016

GUSTAVO MARIN SUPPI

FIREHUNTER E FIREWORKER: TECNOLOGIA COMO INSTRUMENTO DE AUXÍLIO NO SALVAMENTO DE VIDAS

Trabalho de conclusão de curso apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. ESP. Igor Muzeka

Coorientadores:
Profa. DRA. Ingrid Solange Sepúlveda Muñoz
Prof. ME. Márcio José Sembay

Lages, SC ___/___/2016. Nota _____ _____

_____
Coordenador do curso de graduação

LAGES

2016

Dedico este trabalho primeiramente a Deus, por ser essencial e autor da minha vida.

Ao meu pai Edmilton, minha mãe Eunice, minha namorada Andriele, minha irmã Ilana e minhas sobrinhas Isadora e Isabella.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus, por me conceder força e condição para a conclusão do presente trabalho.

Aos meus pais, Edmilton Luís Suppi e Eunice Marin Suppi, por me ensinarem, através do exemplo, a dedicação e a fé. E a minha irmã Ilana Marin Suppi pela assistência.

Minha namorada Andriele Pires pelo apoio, confiança e por sempre estar ao meu lado.

Ao Tenente Varela e ao Sargento Airton Pires pela colaboração e disponibilidade para troca de informações referentes ao projeto.

Ao Igor Augusto Velho por transmitir seu conhecimento sobre sistemas distribuídos.

Aos professores orientadores pelo constante auxílio, revisão e respostas às dúvidas.

Por fim, aos demais professores que mesmo não orientando o projeto auxiliaram para a conclusão do mesmo.

“Atendei-me, povo meu, e nação minha, inclinai os ouvidos para mim; porque de mim sairá a lei, e o meu juízo farei repousar para a luz dos povos”

(Isaías 51:4)

RESUMO

O corpo de bombeiros possui grande influência para a segurança e bem estar da sociedade, por esse motivo é imprescindível que a comunicação efetuada pelo cidadão com a corporação seja a mais precisa, fornecendo dados de suma importância, de maneira que essa troca de informação seja a mais interativa possível. Diante disso, o objetivo do presente estudo é evolucionar o processo de requisição de serviço ao corpo de bombeiros através do desenvolvimento de um sistema distribuído (dois *softwares* para celular, uma aplicação web e um Web Service) que possibilite o envio de informações pertinentes à corporação e esta disponibilize a posição da viatura encaminhada ao atendimento. Para tal, foram efetuadas entrevistas com perguntas abertas a corporação sobre o processo desde a solicitação até a chegada à ocorrência. A aplicação do projeto, de fato, proporcionou melhora na compreensão de todos os envolvidos através da distribuição das informações referentes ao incidente e assim sendo, favoreceu um atendimento do corpo de bombeiros mais ágil à sociedade.

Palavras Chave: *Sistemas distribuídos, Corpo de bombeiros, Android.*

ABSTRACT

The fire department has great influence to safety and welfare of society, therefore it's essential that the communication between citizen and corporation be the most accurate, providing very important data, so that exchange of information be the more interactive as possible. Thus, the aim of this study is to evolve service request process to fire department by developing a distributed system (two software for mobile, a server, a web application and a Web Service) to allow sending of relevant information to corporation which makes available the vehicle's position sent to service. To this end, interviews were conducted with open questions to corporation about the process from the request to the arrival to occurrence. The implementation of the project provided better understand of all stakeholders through distribution of incident information and therefore foster a more agile service of fire department to society.

Keywords: *Distributed systems, Fire department, Android.*

RESUMEN

El departamento de bomberos tiene una gran influencia en la seguridad y el bienestar de la sociedad, por lo tanto, es esencial que la comunicación hecha por el ciudadano con la corporación sea el más preciso, proporcionando datos muy importantes, por lo que este intercambio de información es lo más interactivo posible. Por lo tanto, el objetivo de este estudio es desarrollar el proceso de solicitud de servicio a los bomberos mediante el desarrollo de un sistema distribuido (dos software para móviles, un servidor, una aplicación web y un servicio web) para permitir el envío de información correspondiente a la corporación y esto pone a disposición de la posición de la cabina enviado al servicio. Con este fin, se realizaron entrevistas con preguntas abiertas de la empresa sobre el proceso desde la solicitud hasta la llegada de la ocurrencia. La implementación del proyecto previsto una mejor comprensión de todas las partes interesadas a través de la distribución de información relacionada con el incidente y, por tanto, favoreció un servicio más ágil del cuerpo de bomberos con la sociedad.

Palabras Clave: *Sistemas distribuídos, Corpo de los Bomberos, Android.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

CBMSC - Corpo de Bombeiros Militar de Santa Catarina

CERN – Conselho Europeu para Pesquisa Nuclear

GB – Gigabyte

GHz – Gigahertz

GPS – Global Positioning System

HD – Hard Disk

HTML – Hipertext Markup Language

HTTP – Hipertext Transfer Protocol

IDE – Integrated Developed Environment

JSON – JavaScript Object Notation

RAM – Random-Access Memory

SGML – Standard Generalized Markup Language

SOAP – Single Object Access Protocol

TDD – Test Driven Development

TFS – Team Foundation Server

UML – Unified Modeling Language

XML – eXtensible Markup Language

XP – Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Desde que o ser humano deixou de ser nômade, junto com a adaptação, surgiram novas necessidades, uma delas foi o combate ao fogo. Para suprir tal necessidade, com o tempo, surgiu a corporação hoje conhecida como corpo de bombeiros (CORPO DE BOMBEIROS PARANÁ, 2016a).

Durante séculos, a forma de solicitar atendimento foi aprimorada, iniciando com sentinelas noturnas, posteriormente com os sinos das igrejas e finalmente o sistema de alarmes Gamewell, caixas de avisos que funcionaram por pouco mais de quatro décadas até a chegada do telefone, o qual é utilizado até hoje juntamente com a rádio – meio de comunicação entre a central e as viaturas – para a troca de informação entre os envolvidos na ocorrência (POLÍCIA MILITAR-SP, 2016).

Com o intuito de novamente evolucionar o processo de requisição do serviço dos bombeiros, foi buscada informação referente à metodologia utilizada pela corporação, desde a solicitação do cidadão até a chegada da viatura ao local. Com a finalidade de aprimorar a comunicação da sociedade com o corpo de bombeiros, a fim de compartilhar informações de forma recíproca, foi então proposto um sistema distribuído.

Um sistema distribuído é a união de computadores independentes que trabalham juntos, interligados em rede e se comunicam através de mensagens, dando a aparência ao usuário de ser um sistema único (COULOURIS; DOLLIMORE; KINDBERG, 2007; TANENBAUM; STEEN, 2007; MAGRI, 2013). A crescente importância dos sistemas distribuídos é incontestável. O fato de possibilitar a divisão de suas responsabilidades em subsistemas que ao se comunicarem através da rede podem estar em máquinas distintas, onde cada *hardware* fornece o melhor recurso que dispõe, concede benefícios como, por exemplo, a conservação do sistema funcional mesmo que uma das máquinas pare de funcionar.

Com o avanço da tecnologia e o aparecimento dos *smartphones*, surgiu uma nova fase da computação móvel, a qual viabilizou ainda mais a criação de sistemas distribuídos concedendo-o ênfase na área de processamento de dados (MACHADO JUNIOR, 2014). O que proporcionou destaque para integração dos dispositivos móveis com os sistemas distribuídos não foi apenas o fato da miniaturização de um computador, mas a possível comunicação através da rede sem fio (COULOURIS; DOLLIMORE; KINDBERG, 2007).

Uma grande desvantagem imposta pela criação de sistemas distribuídos é a sua complexidade, para o desenvolvimento correto desse conjunto de *softwares* é imprescindível o uso de técnicas para potencializar a organização e gestão do projeto.

As metodologias ágeis têm sido cada vez mais adotadas por indústrias de *software* (MUSHTAQ; QURESHI, 2012), devido ao seu respaldo à administração de projetos, além de proporcionar estratégias para mudanças constantes, fator imprescindível atualmente no desenvolvimento de sistemas.

Diante da quantidade de recursos fornecidos atualmente pela tecnologia, foi possível sistematizar o processo de atendimento efetuado pelo corpo de bombeiros e através disso, distribuir, entre os envolvidos (sociedade, viatura e central), as informações referentes à ocorrência de maneira mais célere.

1.1 Justificativa

A diversidade de sistemas disponíveis às empresas, que visam reduzir o tempo gasto em determinada tarefa para obtenção de lucros com menos funcionários, é muito grande. O presente trabalho é justificado não por possuir o objetivo dessa redução, visto que não impede o trabalho braçal dos bombeiros, mas sim sistematizar o processo de requisição de atendimento até a chegada da viatura no local da ocorrência, antecipando os bombeiros operacionais sobre a situação e através disso possibilitar uma assistência mais ágil no salvamento de vidas, que é mais valoroso do que o lucro de uma corporação.

1.2 Importância

1.2.1 Acadêmica

Utilizou-se o aprendizado da vida acadêmica na instituição além do aperfeiçoamento do conhecimento adquirido no desenvolvimento de *softwares* para aplicação e utilização da sociedade.

1.2.2 Social

A aplicação desenvolvida possui subsídios para um atendimento amplo, prestando assistência de forma integral à cidade Lages e região. O aplicativo possibilita o envio de

informações importantes para a melhora na eficácia do atendimento efetuado pelo corpo de bombeiros.

2 OBJETIVO

2.1 Objetivo Geral

Desenvolver um sistema distribuído (dois softwares para celular, uma aplicação web e um Web Service) para proporcionar ao corpo de bombeiros maior agilidade através da troca de informação entre os envolvidos nos incidentes e melhorar a comunicação entre a viatura, a central e a sociedade.

2.2 Objetivos específicos

Os objetivos específicos são:

A) Desenvolver um software para celular, com o intuito de possibilitar ao usuário o envio de incidente com imagem para a central do corpo de bombeiros e conhecer a posição da viatura encaminhada a atendê-lo;

B) Desenvolver um software para uso da central de bombeiros a fim de identificar o atendimento adequado para cada incidente recepcionado;

C) Desenvolver um software para celular, para uso dos bombeiros com o objetivo de informá-los sobre o incidente identificado pelo usuário e proporcionar uma rota até o local especificado.

3 REVISÃO DE LITERATURA

Essa revisão de literatura está subdividida em seções que promovem todo o escopo de entendimento sobre o tema proposto. Na seção Corpo de Bombeiros Militar é ressaltada a importância desse profissional na vida social do cidadão comum, assim como a evolução do processo de solicitação de atendimento efetuado pelo cidadão. Em sistemas distribuídos são dispostos alguns conceitos desses sistemas, bem como a arquitetura cliente-servidor escolhida para a execução do projeto. Mais adiante, no tópico Android, são destacados alguns pontos pertinentes sobre esse sistema operacional e sua predominância no mercado. Logo após, é então mencionado a linguagem de programação usada pelo Android e os conceitos de orientação a objetos. Na seção de metodologias de desenvolvimento ágil são referidos dois métodos ágeis e outro que é consequente da união dos dois primeiros. Mais adiante, são expostos os conceitos de testes automatizados e sua importância no desenvolvimento de *software*. Na sequência, será apresentada uma breve história sobre a Web. Por fim, os conceitos de XML e JSON, que serão usados como transporte de dados e as ferramentas do projeto.

3.1 Corpo de Bombeiros Militar

Desde que o ser humano deixou de ser nômade, passaram a surgir novas necessidades, uma delas foi o combate ao fogo. Com o tempo foi dado início a uma corporação, hoje conhecida como Corpo de Bombeiros, que teve inúmeras evoluções e cada uma tornou o trabalho dos bombeiros mais eficaz. Com o decorrer dos anos, deixou de atender apenas incêndios e tornou sua principal missão a de salvar a vida alheia (CORPO DE BOMBEIROS PARANÁ, 2016a).

Na Grécia, o sistema de atendimento funcionava por meio de sentinelas noturnas, as quais tocavam alarmes ao ocorrer incêndio, no Brasil, a história iniciou no dia 2 de julho de 1856 com o decreto assinado pelo imperador Dom Pedro II, que instituiu o Corpo Provisório de Bombeiros da Corte, no Rio de Janeiro. A partir de então, a cidade já se mobilizava de maneira mais ordenada, possuindo orientações de medidas de socorro juntamente com a equipe técnica, a qual fiscalizava os trabalhos de salvamento e extinção de fogo (CORPO DE BOMBEIROS PARANÁ, 2016b).

No Brasil, por volta de 1888 os avisos de incêndio eram feitos por batidas dos sinos das igrejas ou comunicações verbais até chegar ao quartel. Mas em 1896, a companhia

telefônica da época montou um sistema de alarmes para tornar mais eficiente o aviso dos incidentes, que consistia em 50 caixas de aviso dos incêndios, com cerca de 70 quilômetros de extensão. Somente em 1911, foi então inaugurado o sistema de alarmes Gamewell, com 146 caixas de avisos que funcionou por pouco mais de quatro décadas até que em 1955, para facilitar a comunicação entre as viaturas e o quartel, foi então inaugurado a rede de rádios, a qual era responsável por informar o melhor caminho e a evolução da ocorrência. Um ano depois foram eliminadas as caixas de alarmes e dado início ao sistema sucessor, o telefone, ao qual não foi dada a atenção necessária, pois havia poucos aparelhos telefônicos e o número não era de fácil memorização, porém em 1979 entrou em funcionamento o 3º sistema de alarmes, o telefone 193 (POLÍCIA MILITAR-SP, 2016).

No Estado de Santa Catarina, o início do corpo de bombeiros foi através da lei estadual nº 1.288, sancionada pelo governador Hercílio Luz em 16 de setembro de 1919, a qual criava a seção de bombeiros no estado, hoje conhecida como Corpo de Bombeiros Militar de Santa Catarina - CBMSC, constituída de integrantes da força pública, entretanto foi inaugurada apenas em 26 de setembro de 1926 (CBMSC, 2016a).

3.2 Sistemas Distribuídos

Nos últimos anos houve grandes avanços da tecnologia, dentre eles destacou-se os voltados ao desenvolvimento para dispositivos móveis, bem como o aumento na velocidade de acesso à rede, permitindo o usuário se conectar com a internet a qualquer momento, o que fez com que os sistemas distribuídos tivessem ênfase na área de processamento de dados (MACHADO JUNIOR, 2014).

Sistemas distribuídos podem ser definidos como a junção de computadores independentes que trabalham junto, interligados em rede, dando a aparência ao usuário de ser um sistema único (COULOURIS; DOLLIMORE; KINDBERG, 2007; TANENBAUM; STEEN, 2007; MAGRI, 2013) e possui a comunicação entre tais computadores efetuada por troca de mensagens, as quais também são responsáveis por coordenar as ações do sistema (COULOURIS; DOLLIMORE; KINDBERG, 2007).

Sendo assim, a parte mais importante de um sistema distribuído é a colaboração entre os computadores, não tendo relevância o seu tipo, podendo variar desde aparelhos simples até centrais de processamento de alto desempenho (TANENBAUM; STEEN, 2007), tendo sua principal motivação o compartilhamento de recursos (COULOURIS; DOLLIMORE; KINDBERG, 2007).

3.2.1 Cliente–Servidor

Cliente-servidor é uma arquitetura centralizada onde os seus processos são divididos em duas partes, conhecidas como: cliente e servidor (TANENBAUM; STEEN, 2007; COMER, 2007; BARRETT; KING, 2010), visto que cliente se refere a um programa (processo), que demanda um serviço do servidor, que é um programa semelhante, porém com a função de processar as mensagens recebidas pelos clientes respondendo adequadamente (COULOURIS; DOLLIMORE; KINDBERG, 2007).

O objetivo principal dessa arquitetura é que as funções sejam divididas entre o cliente e o servidor, de modo que potencialize os recursos do sistema permitindo seu compartilhamento (STALLINGS, 2005). Devido o programa referido como cliente estar funcionando em um computador distinto do servidor, ele é responsável apenas por mostrar os dados ao usuário e solicitar processamento, enquanto o servidor tem a função de processar dados ou recuperá-los e fornecer ao cliente (BARRETT; KING, 2010).

3.2.2 Web Service

Com o surgimento de diversas linguagens de programação e cada uma destas com sua maneira própria de processar dados, torna-se impossível a comunicação entre elas sem um mediador. E, portanto, para solução de tal problema pode ser utilizado um Web Service, o qual efetua a troca de mensagens entre aplicações através do protocolo SOAP (*Single Object Access Protocol*) (MAGRI, 2013).

Sendo assim, Web Service é uma aplicação distribuída, que não possui interface gráfica, hospedada na internet e projetada para fornecer um serviço a outro programa, podendo atender diferentes tipos de dispositivos (figura 1) (MAGRI, 2013; MACHADO JUNIOR, 2014). Desta maneira, em sua simplicidade, Web Service é uma classe hospedada na internet que pode ser acessada por qualquer tipo de aplicação, fornecendo serviços ao solicitante através de seus métodos.

Figura 1: Web Service.

Fonte: https://community.dynamics.com/crm/b/microsoftdynamicscrmatoz/archive/2015/10/22/why-wrap-bridge-tunnel-crm-web-services

3.2.3 Protocolo SOAP

Consiste em um protocolo padrão baseado em XML, utilizado pelo Web Service para remover sua dependência com a plataforma em que está funcionando, bem como a linguagem de programação utilizada para a sua criação (MAGRI, 2013). Toda mensagem SOAP é formada por um envelope, e este é constituído por um cabeçalho (*header*), onde as informações referentes ao servidor SOAP são inseridas, e um corpo (*body*), que contém os dados da mensagem a ser transportada (LACERDA, 2007).

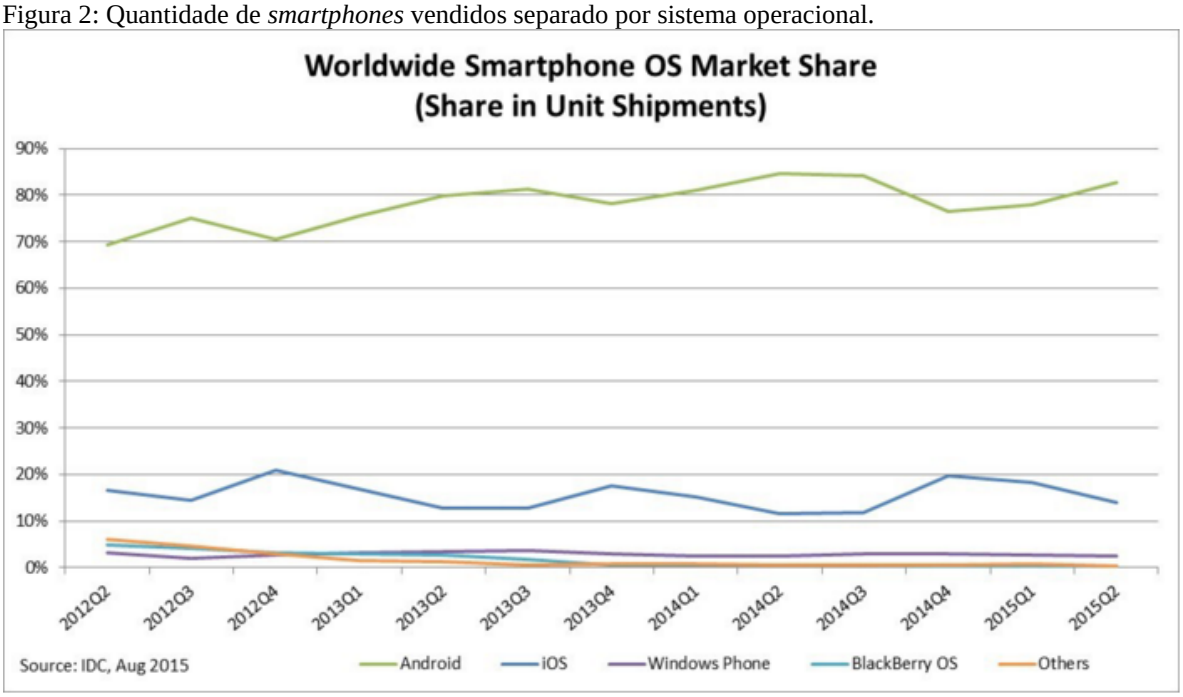
3.3 Android

Com a evolução da tecnologia, hoje cerca de três bilhões de pessoas têm um celular e pelo menos um terço está conectado à internet (OPEN HANDSET ALLIANCE, 2016), também há a previsão de que em 2016 serão lançados mais de um bilhão e meio de novos *smartphones* (CANALYS, 2016), isto ocorre devido grande parte dos usuários estar à procura de um dispositivo móvel moderno, com grandes quantidades de recursos, bem como a facilidade em sua navegação, já os desenvolvedores buscam uma plataforma de desenvolvimento eficiente para a criação de novos aplicativos (DEITEL et al., 2013; LECHETA, 2013).

Em resposta a essa busca, surgiu o sistema operacional Android, baseado em Linux, que teve sua primeira versão lançada em 2008, por possuir o seu código fonte aberto e

gratuito, que permite ver como seus recursos são desenvolvidos, o que possibilita a criação de aplicativos que extraiam o máximo que o aparelho tem para oferecer, tal sistema operacional foi desenvolvido inicialmente pela Android Inc., e posteriormente adquirida pela Google (DEITEL et al., 2013; LECHETA, 2013). Em 2015, Android foi o sistema operacional hospedado em mais de 80% dos smartphones (figura 2) (IDC, 2015).

Figura 2: Quantidade de *smartphones* vendidos separado por sistema operacional.



Fonte: IDC, 2015.

É importante ressaltar que se em determinado momento o desenvolvedor deseja utilizar de algum artifício do dispositivo como GPS e *Bluetooth*, é possível devido seus recursos serem acessíveis de qualquer parte do sistema, aprimorando o desenvolvimento de aplicativos dinâmicos (PACHECO JÚNIOR; CASTRO, 2011).

3.4 Java

Java é uma linguagem de programação orientada a objetos, versátil e muito segura, onde o software gerado não depende da estrutura física do computador nem do sistema operacional em que está funcionando, por este motivo um programa originado pelo Java é considerado independente de arquitetura (WINDER; ROBERTS, 2009; BONAN, 2008).

Além disso, Java possui um compilador que transforma o código escrito pelo usuário em *bytecodes* que são interpretados por sua máquina virtual, a qual é responsável por traduzi-los para o aparelho em que o aplicativo está sendo executado (WINDER; ROBERTS, 2009).

3.5 Programação orientada a objetos

O estilo de programação orientado a objetos se tornou muito popular nos dias de hoje por possuir uma série de ferramentas que possibilitam um código mais gerenciável e adaptável a mudanças, assim como a manutenção de um sistema legado (DIAS, 2009), além de ter inúmeras funcionalidades poderosas que auxiliam em um código mais legível tornando o sistema mais confiável (WINDER; ROBERTS, 2009).

Entretanto, para que um programa seja orientado a objetos há a necessidade de utilizar objetos e estes sejam instâncias de classes, as quais devem se relacionar uma com a outra através de herança, se ao menos um dos três pontos não for encontrado, então tal programa não é orientado a objetos (AGUILAR, 2008).

Neste modelo, os atributos e ações de um elemento são agrupados em um único objeto (FARINELLI, 2007), que faz sua comunicação com outro objeto através de troca de mensagem (TUCKER; NOONAN, 2008).

Figura 3: Os pilares da programação orientada a objetos.

Fonte: http://www.devmedia.com.br/artigo-clube-delphi-93-orientacao-a-objetos-no-delphi-for-php/10635

Conforme o quadro a seguir, serão demonstradas as funções primordiais que norteiam a orientação a objeto.

Quadro 1: Funções primordiais da orientação a objetos.

Classe	É um modelo, encontrado através da abstração, de características e comportamentos de um objeto.
Objeto	O resultado de uma construção utilizando uma classe como modelo é um objeto, ou seja, objeto é a forma concreta de uma classe.
Abstração	É encontrar o essencial, o que cada objeto tem de mais significativo, removendo detalhes irrelevantes, os quais não irão auxiliar na resolução do problema em questão.
Encapsulamento	É a divisão do código de um programa em partes menores, onde cada parte tem um contexto próprio independente, é também responsável por restringir o acesso aos dados de uma classe.
Herança	Permite que as características comuns das classes filhas sejam colocadas em uma classe mãe, onde todo o comportamento da superclasse é passada para as subclasses.
Polimorfismo	É um método hospedado na superclasse onde suas subclasses podem reescrevê-lo de maneiras diferentes, fazendo com que esse método possua várias formas.
Classe abstrata	É um modelo de objeto que não pode ser instanciado por apresentar métodos que não possuem corpo, declarados na classe "mãe" e definidos na classe "filha".
Interfaces	São utilizadas quando o resultado da abstração gera um comportamento que é comum a vários objetos que não possuem relação entre si.

Fonte: Fowler, 2004; Farinelli, 2007; Aguilar, 2008; Bonan, 2008; Tucker; Noonan, 2008; Schach, 2009; Winder; Roberts, 2009; Melo, 2010; Sbrocco, 2011; Sebesta, 2011; Guerra, 2013; Microsoft, 2016a.

3.6 Metodologias de desenvolvimento ágil

Em fevereiro, após um encontro em Utah nos Estados Unidos da América, no ano de 2001, foi criado um novo conceito em gerenciamento de projetos, o termo "ágil" que deu início a "Agile Alliance" (MARÇAL, 2009), essa que segundo Beck et al. (2001) propõe no manifesto ágil "Indivíduos e interação entre eles mais que processos e ferramentas, software em funcionamento mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos, responder a mudanças mais que seguir um plano".

Seguindo esses conceitos, metodologias ágeis se adaptam às mudanças constantes do produto com mais facilidade (MÜLLER NETO, 2009), visto que no período de desenvolvimento a documentação ainda estará sendo criada e a equipe compartilhando mais conhecimento entre seus membros (LIBARDI; BARBOSA, 2010). Seu uso está sendo adotados cada vez mais por indústrias de software por entregar produtos com mais qualidade e eficiência (MUSHTAQ; QURESHI, 2012).

Outro ponto pertinente é não tentar encontrar todos os problemas e mapear previamente tudo que possa ocorrer durante o processo de desenvolvimento do produto, com

isso, há redução do tempo da análise e planejamento antecipado que na maioria das vezes é alterado durante o desenvolvimento (LIBARDI; BARBOSA, 2010).

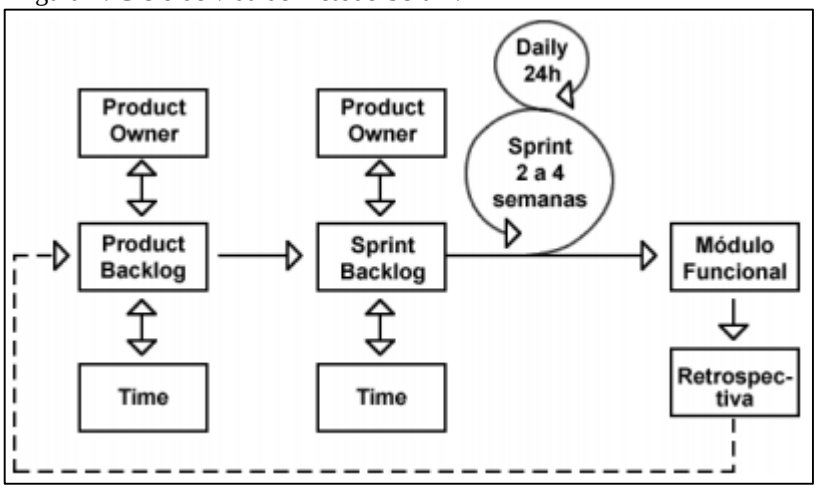
Scrum

Criado por Ken Schwaber e Jeff Sutherland, Scrum é um método ágil que possui uma série de atividades que dão o monitoramento e *feedback* tanto à equipe quanto aos *stakeholders*, essas atividades levam à tona as deficiências e impedimentos a todos os envolvidos no projeto, possibilitando tomadas de decisões estratégicas de mais valor (MARÇAL, 2009).

Além disso, Scrum procura entregar um produto de valor desenvolvendo somente o que é importante para o cliente, também procura eliminar a burocracia desnecessária como formalizações e documentações em excesso, assim como o planejamento exagerado (MÜLLER NETO, 2009).

Segundo Godoy Neto (2014), Scrum é um gerenciamento ágil de projetos iterativo e incremental (figura 4) que pode ser dividido em três fases: a primeira consiste em elaborar o planejamento da próxima *sprint*, a qual é a segunda etapa, onde é desenvolvido o incremento do sistema e então somente na terceira parte são elaborados os manuais e documentos.

Figura 4: Ciclo de vida do método Scrum.



Fonte: Godoy Neto, 2014.

Extreme Programming (XP)

Direcionada para pequenas e médias equipes, XP é uma metodologia ágil que conduz o desenvolvimento de *softwares* que possuem requisitos vagos e constante mudança (MÜLLER NETO, 2009; LODDI et al., 2010), possui o objetivo de entregar o produto com o maior valor possível ao cliente, para isso, utiliza um conjunto de práticas que trabalham em conformidade (TELES, 2014).

O uso dessa metodologia melhora o projeto de software em pelo menos quatro lugares, sendo eles, na comunicação, *feedback*, simplicidade e coragem (MÜLLER NETO, 2009), onde a comunicação é feita com o cliente durante todo o processo de desenvolvimento (TELES, 2014), pelo qual é fornecido o devido retorno aos envolvidos no projeto através do *feedback* (LODDI et al., 2010), o qual também pode ser concedido por meio de testes automatizados (MÜLLER NETO, 2009) que quando criados através do desenvolvimento guiado por testes, trazem a simplicidade e auxiliam na segurança do código.

Método ágil híbrido

Método ágil híbrido é a combinação de práticas que provêm de metodologias ágeis diferentes, ou seja, busca as melhores práticas de cada metodologia e as unem em uma nova (MUNDACA; ABARCA, 2015).

A integração dos modelos XP e Scrum é justificada devido o modelo XP fornecer grande suporte em projeção do produto contribuindo com as melhores práticas da engenharia de *software*, mas deixando a desejar em práticas de gestão de projetos, e por outro lado o Scrum possuir uma série de atividades voltada ao gerenciamento de projetos, mas não possui suporte a práticas de programação para o desenvolvimento do *software* (MUSHTAQ; QURESHI, 2012, AHMAD; SOOMRO; BROHI, 2014). O método ágil híbrido resultante da integração dessas metodologias contém as boas práticas de desenvolvimento de software do XP com a gestão de projetos do Scrum e consequentemente há melhoria na produtividade, no gerenciamento do projeto e na qualidade do produto a ser entregue ao cliente (MUSHTAQ; QURESHI, 2012).

3.7 Testes automatizados

Muitos desenvolvedores não dão a devida atenção ao código que está sendo implementado, simplesmente o escrevem para conseguir terminar uma determinada funcionalidade do sistema e entregá-la, esquecendo muitas vezes da estrutura do código e da segurança de que a funcionalidade foi desenvolvida corretamente. O fato de não tomar esse cuidado pode acarretar sérios problemas no futuro. Devido à falta de segurança, o código é mais suscetível a erros que ao se deparar com o código fonte mal estruturado, o programador levará um tempo maior para a correção.

Durante todo o processo de desenvolvimento, nota-se que uma pequena parcela do tempo é gasta em projetar e escrever o código, a maior parte do tempo é gasta em sua depuração, visto que na maioria das vezes, corrigir o erro é uma tarefa fácil, trabalhoso é encontrar onde ele está, além do fato de que ao arrumá-lo há uma grande possibilidade de fazer com que surjam novos erros (FOWLER, 2004).

Os testes sempre estarão presentes no desenvolvimento de um *software* devido à necessidade de entrega de qualidade ao usuário, sabe-se que não é possível removê-los, porém há a alternativa de automatizá-los, tendo o benefício de executar os mesmos testes que uma pessoa faria, todavia muito mais rápido.

3.7.1 Test-Driven Development

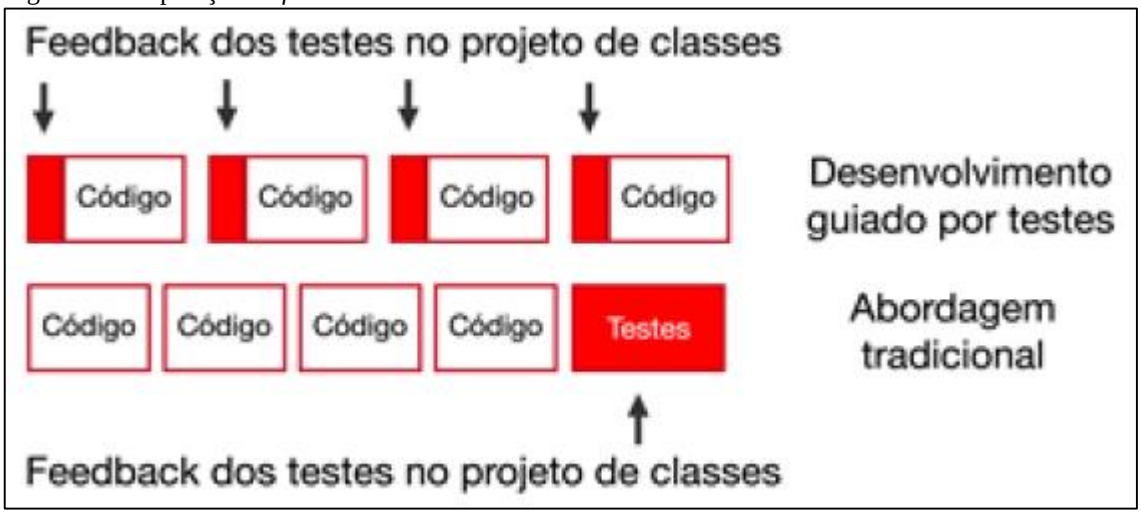
O processo tradicional, primeiro criar o código e depois testar, acostumou grande parte dos desenvolvedores, porém ao esquivar deste processo, encontramos o TDD que consiste em primeiro testar para depois desenvolver o código (ANICHE, 2012).

Parece não fazer sentido testar algo que ainda não existe, porém quando se cria o teste para certa funcionalidade antes da sua implementação, o desenvolvedor deixa de pensar em como a classe será feita e passa a pensar no que ela deve fazer, permitindo o encontro de mais cenários de teste e aprimorando a segurança do código em questão, além do benefício de ter a certeza de que foi implementado ao menos um teste para cada funcionalidade do código criado (ANICHE, 2012).

Outro benefício de desenvolver guiado por testes é de permitir alternar entre avançar rapidamente em sua implementação e progredir em passos pequenos, essa maleabilidade pode ser usada conforme a complexidade do código, utilizando passos pequenos em criação de novas funcionalidades complexas e passos longos em código simples com pouca regra de

negócio (BECK, 2010), além de que o uso do TDD faz com que o desenvolvedor busque sempre a simplicidade, escrevendo apenas o código necessário para resolver os cenários dos testes criados, auxiliando também o desenvolvedor a pensar em como a funcionalidade que está sendo implementada irá funcionar dentro do sistema como um todo (ANICHE, 2012).

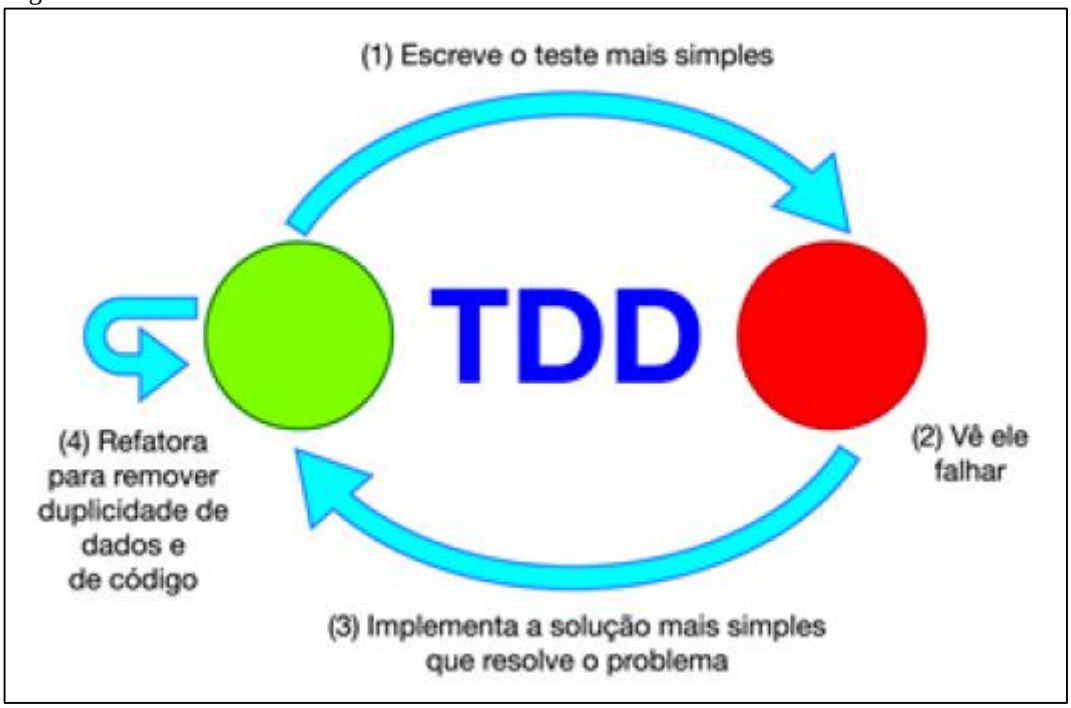
Figura 5: Comparação de *feedback* do desenvolvimento tradicional com o TDD.



Fonte: Aniche, 2012.

Segundo Aniche (2012) e Borges (2006), os desenvolvedores sabem que há maior facilidade em alterar um código que foi criado pouco tempo atrás do que esperar um longo tempo, pelo fato de receber *feedback* constante ainda em fase de desenvolvimento (figura 5), o TDD acaba mostrando ao desenvolvedor os problemas com antecedência, diminuindo o custo de manutenção e dando ao código mais segurança.

Figura 6: Ciclo do TDD



Fonte: Aniche, 2012.

O ciclo do TDD (figura 6) é: o desenvolvedor escrever um teste rapidamente, vê-lo falhar, mudar uma pequena parte do código para fazer o teste passar, rodar todos os testes e então refatorar, removendo a duplicação de código. Sendo assim, esse fluxo não permite o desenvolvedor se acomodar com um código limpo, mas sim melhorá-lo a cada ciclo (BECK, 2010). Esse fluxo conhecido como “vermelho, verde, refatora” divide o trabalho a ser executado em partes pequenas e gerenciáveis, forçando o desenvolvedor a pensar minuciosamente sobre o código ao escrevê-lo (WINDER; ROBERTS, 2009).

Testes indicam a qualidade do produto, por esse motivo quando é criado um novo teste é imprescindível que seja planejado cuidadosamente para evitar testes improvisados e irreproduzíveis que ao invés de auxiliar o desenvolvedor, acabam dificultando seu trabalho (PAULA FILHO, 2005).

3.8 World Wide Web

Desenvolvida pelo físico e engenheiro de software Tim Berners-Lee com a colaboração do físico belga Robert Cailliau, a *Web* foi criada no Conselho Europeu de Pesquisa Nuclear (CERN), de maneira que suportasse uma grande variedade de computadores e redes, com o intuito de disponibilizar o conteúdo do laboratório de pesquisa para ser

acessada por todos (CARVALHO, 2006) possibilitando aos cientistas das universidades e de institutos de todo o mundo a trocarem informações (CERN, 2016).

Para a criação da *Web*, Berners-Lee desenvolveu também a linguagem de marcação HTML, o protocolo HTTP e o *Browser* WorldWideWeb, necessários para sua utilização (CARVALHO, 2006).

HTML

HTML é uma sigla em inglês para *HyperText Markup Language*, sua tradução significa linguagem para marcação de hipertexto, teve aprovação no CERN e no mundo por ser simples de entender e escrever, além de ser baseada em SGML que era preferência da comunidade internacional direcionada a pesquisas sobre hipertextos (CARVALHO, 2006), sua função primordial é a estruturação do documento (SILVA, 2011).

HTTP

O HTTP *(Hipertext Transfer Protocol)* é um protocolo orientado a conexões usado para a comunicação entre o servidor Web e o *browser* (COMER, 2006; TANENBAUM; STEEN, 2007), onde o navegador envia requisições ao servidor, através desse protocolo, mediante solicitação do usuário (KUROSE; ROSS, 2006).

WorldWideWeb Browser

Em 1990, quando a *Web* foi criada, a única maneira de ter acesso a ela era pelo navegador WorldWideWeb que inicialmente não possuía interface gráfica e funcionava apenas no computador NeXTcube, posteriormente foi renomeado para Nexus com o intuito de não gerar confusão com a *World Wide Web* (BERNERS-LEE, 2016; CARVALHO, 2006).

3.9 XML

O XML (*Extended Markup Language*) é uma linguagem de marcação que permite a especificação dos dados no documento criando suas próprias marcações (figura 7), representando dados de uma forma textual e dando sentido a cada pedaço da informação (ALMEIDA, 2002; NARDON, 2002), tendo como alguns de seus objetivos os de ser legível a

qualquer indivíduo, suportar grande diversidade de aplicativos e ser de fácil criação (W3C, 2008).

Figura 7: Exemplo de XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<IncidentMarker>
    <Id>2</Id>
    <Latitude>-27.8217587</Latitude>
    <Longitude>-50.2641505</Longitude>
    <Title>Incêndio</Title>
    <Description>Fogo em árvore</Description>
    <GoingUser>0</GoingUser>
    <Active>true</Active>
    <Icon>0</Icon>
</IncidentMarker>
```

Fonte: Arquivo do Autor.

3.10 JSON

Assim como o XML, o JSON (*JavaScript Object Notation*) é legível e independente de plataforma, porém é muito mais simples (figura 8), principalmente aos desenvolvedores já familiarizados com JavaScript, também possui uma sintaxe sucinta além de produzir um texto formatado, o qual remove a maioria dos espaços do documento, tornando-o mais leve (MICROSOFT, 2016b). Por usar convenções entendíveis por inúmeras linguagens de programação, JSON se tornou um ótimo meio de troca de dados (JSON, 2016).

Figura 8: Exemplo de JSON.

Fonte: Arquivo do Autor.

3.11 Ferramentas do projeto

Na sequência do trabalho, serão descritas as principais ferramentas a serem utilizadas no desenvolvimento do projeto.

Android Studio

Android Studio refere-se a IDE oficial voltada para desenvolvimento de aplicativos Android, essa ferramenta disponibiliza emulador de *smartphones* e *tablets* que possuam o sistema operacional Android, assim como para Android *Wear* (ANDROID DEVELOPER, 2016). Além disso, disponibiliza um ambiente agradável e ágil para o desenvolvimento de novos aplicativos.

Visual Studio

Visual Studio é definido como um conjunto de ferramentas unidas que formam um ambiente de desenvolvimento voltado para criação de código, testes, análise de qualidade, desempenho e depuração (MICROSOFT, 2016c).

Visual Studio tem suporte para integração com o Team Foundation Server que será explicado no próximo tópico.

Team Foundation Server

A comunicação entre todos os membros da equipe envolvida é essencial para que todo projeto seja bem sucedido, para isso, o TFS possibilita o controle de versão do código em desenvolvimento e permite que sua alteração seja vinculada a tarefas o que proporciona, juntamente com o *kanban* e o *burndown*, a gestão visual da evolução do projeto de forma mais interativa (BLANKENSHIP et al., 2012). Além disso, também disponibiliza o agendamento de execução automática dos testes automatizados.

Sublime Text 3

O editor *Sublime Text*, atualmente em sua terceira versão, possui uma série de funcionalidades que auxiliam o desenvolvimento do software, entre elas, a paleta de comandos, indentação automática, *intellisense*, gerenciador de pacotes e suporte para inúmeras linguagens (HAUGHEE, 2013).

Trello

Trello é uma ferramenta que possibilita a organização de projetos de maneira simples e gratuita. Disponibiliza um *kanban* personalizável, que permite adicionar membros, selecionar a data de entrega e anexar arquivos conforme a necessidade do usuário, além de disponibilizá-lo para todos os interessados pelo andamento do projeto.

O andamento do desenvolvimento desse projeto já está disponível em: https://trello.com/b/cuZdHQxr/tcc

4 METODOLOGIA

4.1 Documentação

Pesquisa documental tem por finalidade posicionar o pesquisador de maneira que entre em contato com todo o conteúdo que exista sobre determinado assunto, englobando por completo a bibliografia e documentos públicos (MARCONI; LAKATOS, 2010). A utilização desses documentos é voltada para explicações e fonte de informações, tendo em vista o esclarecimento, bem como a afirmação de determinado assunto (SÁ-SILVA; ALMEIDA; GUIDANI, 2009).

4.2 Natureza da pesquisa

A natureza da pesquisa utilizada foi a qualitativa, que segundo Gibbs (2009) visa analisar as experiências dos indivíduos, tanto abordagens cotidianas ou profissionais, examinar interações e comunicações em desenvolvimento através da observação, além de investigação de documentos.

A pesquisa qualitativa procura gerar novas teorias e conceitos durante toda a coleta de dados e assim ampliar o conhecimento e a visão do pesquisador (GIBBS, 2009).

4.3 Método da pesquisa

O método usado nessa pesquisa foi o estudo de campo, o qual de acordo com Marconi e Lakatos (2010) trata-se de uma metodologia que tem por objetivo alcançar respostas através das informações sobre determinado problema, utilizando a observação de fatos e fenômenos assim como a coleta de dados referentes ao tema.

4.4 Técnicas da pesquisa

Foi utilizada a técnica de observação como condutora da pesquisa, a qual conforme Marconi e Lakatos (2010) é um elemento básico da investigação científica, voltada a coleta de dados, com o propósito de auxiliar o pesquisador a identificar determinados objetivos que inconscientemente orientam seu comportamento.

4.5 Coleta de dados

Entrevista é utilizada na investigação social para a aquisição de dados e tratamento de problemas sociais, é definida como o encontro de duas pessoas com o propósito de obtenção de informações sobre determinado assunto através de diálogo de natureza profissional (MARCONI; LAKATOS, 2010).

A coleta de dados iniciou-se por meio de entrevistas com perguntas abertas através de um contato direto, pessoalmente, sem o uso de formulários impressos, com o intuito de coletar informações referentes à comunicação entre os integrantes do corpo de bombeiros ao surgir um novo incidente.

Em seguida, para a coleta de informações referente à viabilidade, foram efetuadas reuniões com o Tenente Varela e com o Sargento Airton Pires. Após a aceitação do projeto pelo corpo de bombeiros de Lages - SC foi dado início ao projeto.

5 CRONOGRAMA

O seguinte cronograma foi utilizado para o desenvolvimento deste trabalho.

Quadro 2: Cronograma

Atividade	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Revisão de literatura											
Estudo de técnicas											
Especificação do protótipo											
Ajustes metodologia											
Entrega do TCC 1 à coordenação											
Defesa TCC 1											
Desenvolvimento do sistema											
Reunião com bombeiros											
Hospedagem no servidor de testes											
Hospedagem no servidor de produção											
Entrega do TCC 2 à coordenação											
Defesa TCC 2											

Fonte: Arquivo do Autor.

6 PROJETO

Em um incidente, o cidadão possui a informação referente ao local da ocorrência, o estado das vítimas, em fim, a visão completa do ambiente. O corpo de bombeiros por sua vez conhece as viaturas e para quais incidentes estão encaminhadas.

O intuito deste projeto é distribuir a informação, que ambas as partes possuem isoladamente, com finalidade de fornecer ao cidadão informações que apenas os bombeiros possuem e vice-versa.

Com os dados obtidos através das entrevistas, foi conhecido o fluxo de atendimento do incidente, o qual consiste em um cidadão entrar em contato com a central pelo telefone 193, sendo que esta, por sua vez, se comunica – por intermédio do rádio – com o bombeiro operacional, o qual se direciona ao incidente. Esse fluxo ocorre na maioria dos estados, com exceção de alguns municípios de Santa Catarina, os quais possuem o Firecast cuja metodologia será detalhada posteriormente.

Com o intuito de possibilitar a troca da informação com mais agilidade e precisão, foi desenvolvido um sistema distribuído, o qual possui dois softwares para celular, uma aplicação Web e um Web Service que serão descritos após os diagramas UML.

6.1 Diagramas UML

UML é uma linguagem de modelagem que visa auxiliar na projeção do sistema como um todo, assim como entender o seu comportamento através de modelagem visual (SBROCCO, 2011), além de possibilitar uma forma padrão de conceituar tanto regras de negócio quanto as funções do sistema (BOOCH; RUMBAUGH; JACOBSON, 2006).

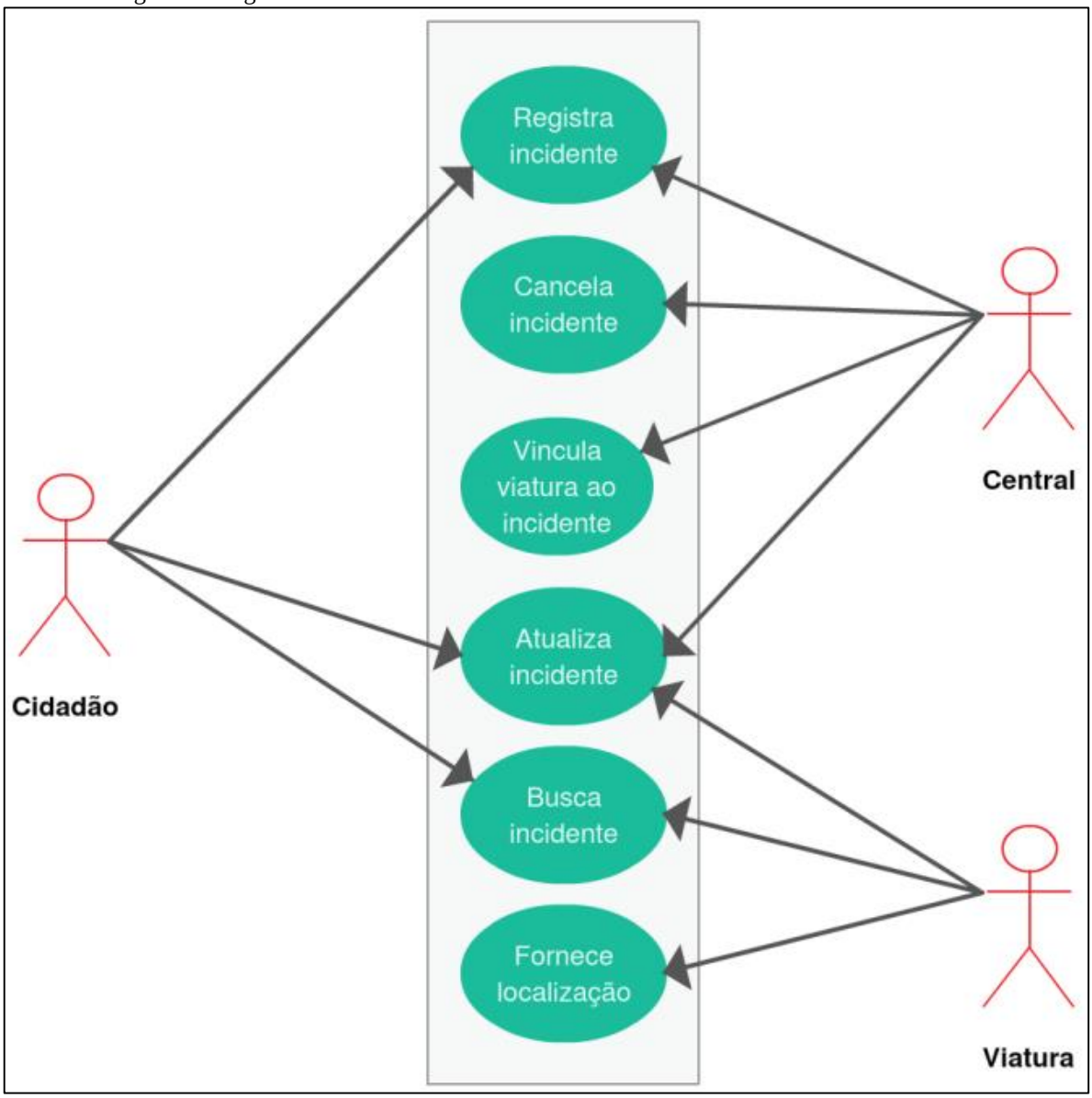
A propósito, é uma linguagem expressiva que possibilita a visão necessária para o desenvolvimento de sistemas (BOOCH; RUMBAUGH; JACOBSON, 2006). “Esta linguagem se tornou, nos últimos anos, a linguagem-padrão de modelagem de software adotada internacionalmente pela indústria de Engenharia de Software” (GUEDES, 2007, p.13).

Caso de uso

Utilizado como auxiliador para a análise dos requisitos, assim como na compreensão do sistema como um todo (GUEDES, 2007), o diagrama de caso de uso é responsável por descrever as ações que o sistema cederá aos atores e expressar seu comportamento ou parte

dele (MELO, 2010), como demonstra a figura 9 a qual aponta os procedimentos exercidos pelo usuário ao interagir com o sistema.

Figura 9: Diagrama de caso de uso.



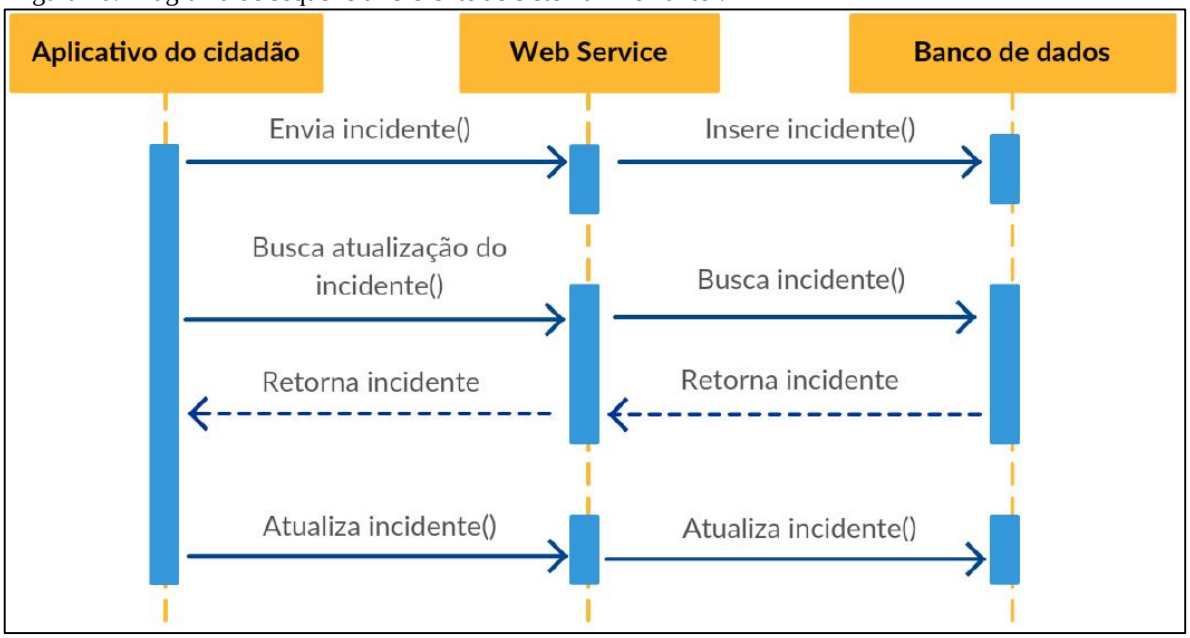
Fonte: Arquivo do Autor.

Diagrama de sequência

Com base nos casos de uso, o diagrama de sequência é responsável por enfatizar a sequência temporal das ações através da representação do comportamento dos objetos, que se relacionam uns com os outros através de trocas de mensagens (SBROCCO, 2011). Procura definir os objetos envolvidos em determinado processo, mostrando seu desfecho (GUEDES, 2007).

A figura 10 demonstra o fluxo das atividades que o aplicativo do cidadão poderá exercer, sendo elas o envio e atualização do incidente, como também a busca das atualizações feitas pela central.

Figura 10: Diagrama de sequência referente ao sistema FireHunter.



Fonte: Arquivo do Autor.

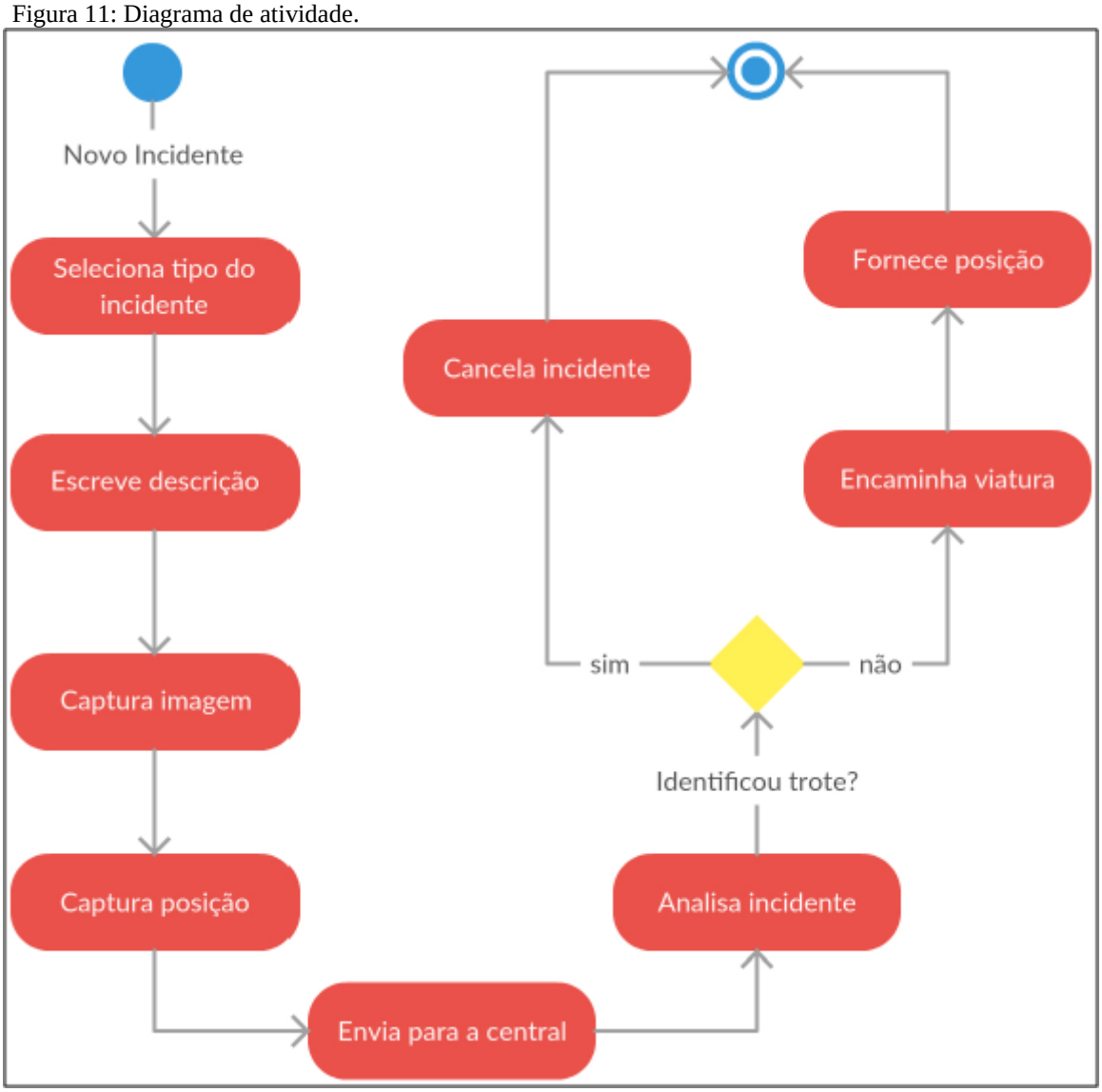
Diagrama de atividade

A partir da versão 2.0 da UML, o diagrama de atividade deixou de ser um caso especial do diagrama de estados passando a se basear em redes de Petri (GUEDES, 2007).

O propósito desse diagrama é descrever os passos a serem executados para concluir uma tarefa específica (GUEDES, 2007; SBROCCO, 2011).

A figura 11 é referente ao fluxo feito a partir do envio de um incidente pelo cidadão, passando pela central até a chegada do incidente registrado à viatura encaminhada ao local.

Figura 11: Diagrama de atividade.



Fonte: Arquivo do Autor.

6.2 FireHunter

FireHunter é o aplicativo disponível ao cidadão para o envio de incidentes.

Esse programa tem como o objetivo o envio da posição atual do usuário, juntamente com uma imagem da ocorrência, requisitando atendimento. Após o aplicativo referente à central encaminhar a viatura até o incidente, o solicitante receberá a posição da viatura que está se dirigindo até o local.

A figura 12 refere-se à tela inicial do aplicativo. Por objetivar agilidade, sua interface é simples e com pouca entrada do usuário, visto que será enviada a posição automaticamente e o tipo do incidente, que estará em um *combobox*, além de permitir o envio de uma breve descrição caso necessário. Depois de pressionado o botão de envio será solicitado que o

usuário fotografe o acidente. Além disso, possui um botão para buscar incidente próximo, o qual fornece aos usuários próximos do acidente uma maneira de visualizar o andamento do mesmo, bem como atualizá-lo, não limitando tais ações ao usuário que o registrou.

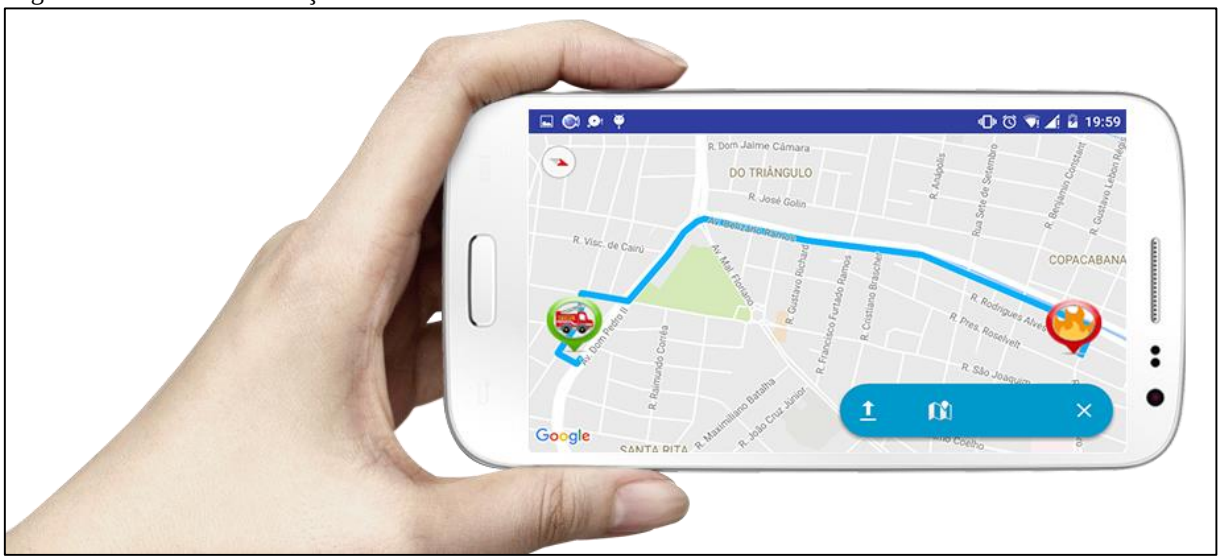
Figura 12: Tela inicial do FireHunter.

Fonte: Arquivo do Autor.

Na figura 13, é mostrada a tela de visualização do andamento do incidente, a qual é acessada após a central encaminhar uma viatura à ocorrência, nessa tela são disponibilizadas a localização e a possível rota da viatura encaminhada. Também é fornecido ao usuário um

menu com dois botões, referentes a atualização e centralização do incidente na tela, respectivamente.

Figura 13: Tela de visualização do andamento do incidente.



Fonte: Arquivo do Autor.

A figura 14, por sua vez, apresenta a tela para atualização do incidente, a qual permite atualizar a descrição, imagem e localização da ocorrência. Apenas os itens fornecidos pelo usuário serão atualizados. A permissão da atualização do local do incidente foi concedida devido à falta de precisão do localizador dos dispositivos móveis em alguns momentos, sendo assim, nesta tela o cidadão poderá visualizar a posição em que o marcador será atualizado no mapa.

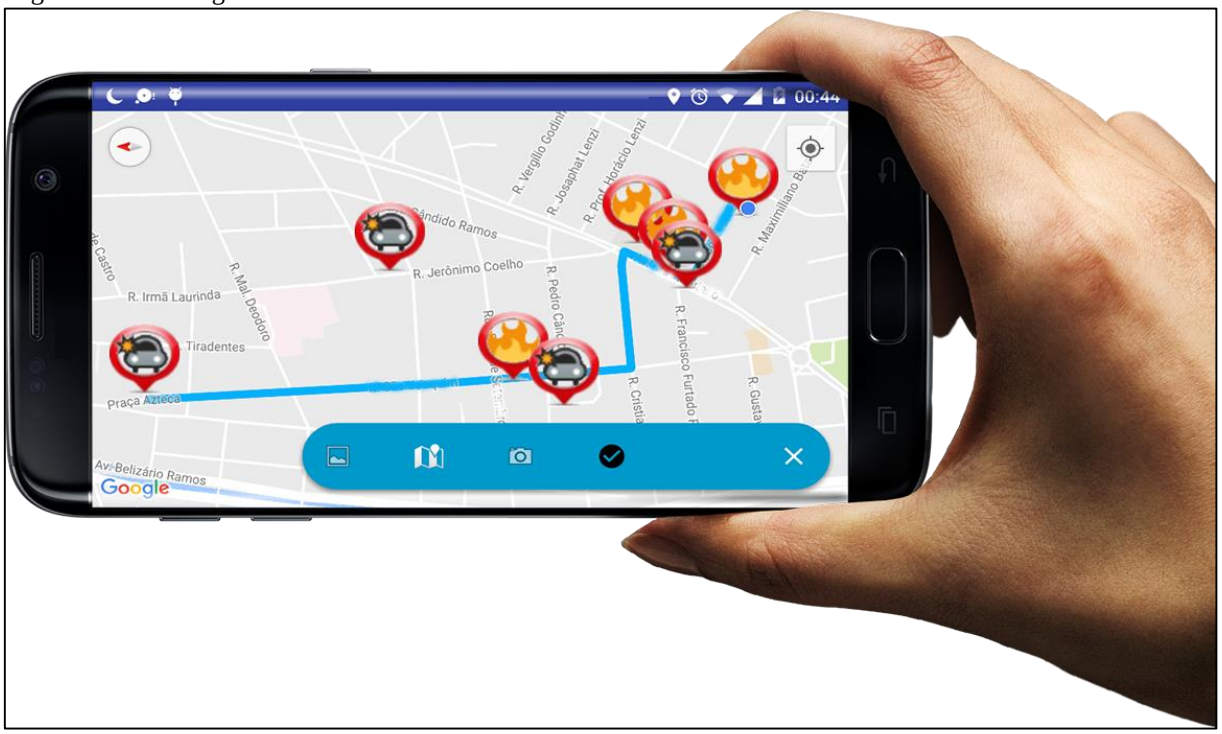
Figura 14. Tela de atualização do incidente.

Fonte: Arquivo do Autor.

6.3 FireWorker

FireWorker é o aplicativo disponível apenas ao corpo de bombeiros, localizado nas viaturas, com o intuito de mostrar no mapa os incidentes e veículos da corporação registrados, além disso, permite, através de um menu, o bombeiro operacional a visualizar a imagem, centralizar o marcador do incidente, fotografar e encerrar a ocorrência.

Figura 15: Tela de gerenciamento do incidente do FireWorker.



Fonte: Arquivo do Autor.

6.4 Sistema localizado na central do corpo de bombeiros

O sistema localizado da central possui a finalidade de proporcionar a análise do incidente recepcionado e permitir a seleção de quais viaturas irão atendê-lo. Mostra uma visão mais ampla de todos os incidentes.

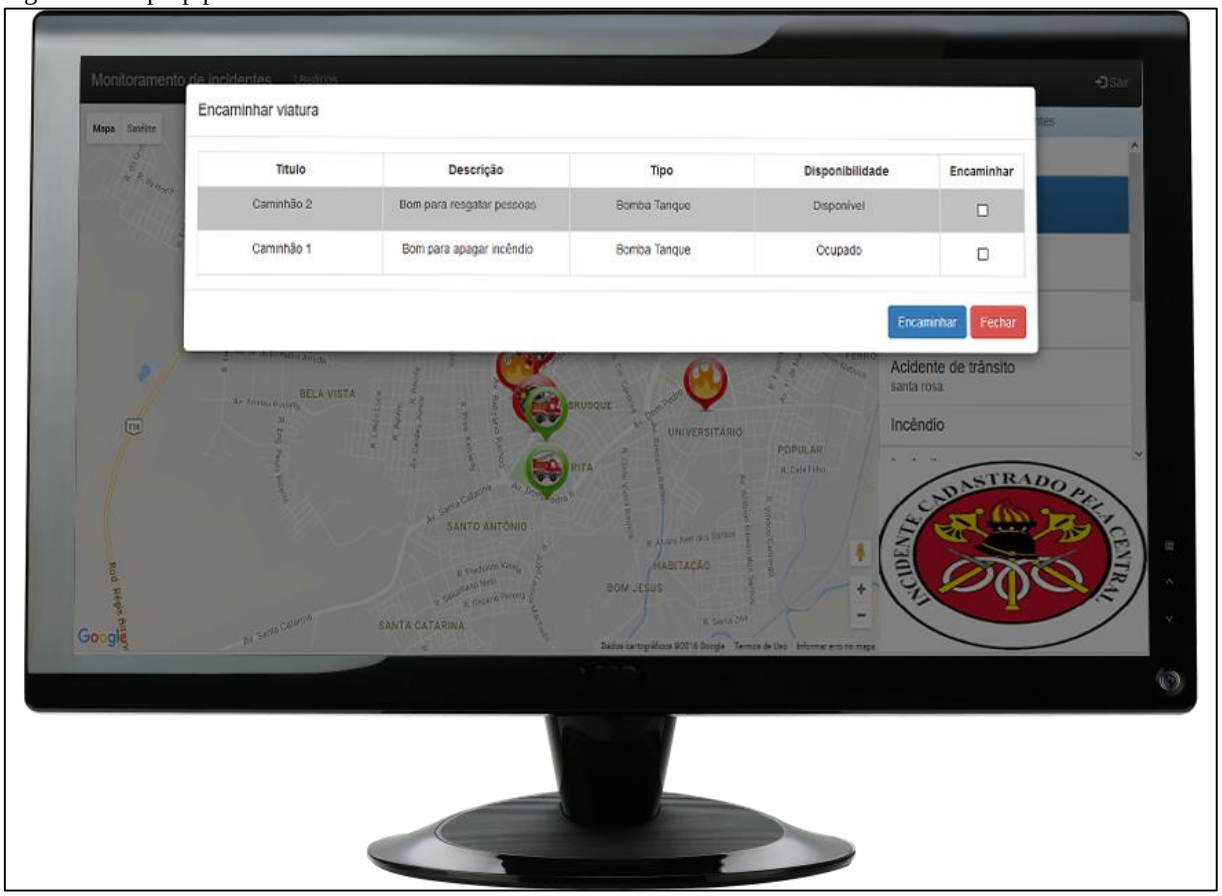
Conforme a figura 16, o responsável por gerenciar o aplicativo da central pode navegar entre os incidentes através da lista, além de visualizar todas as viaturas. É encarregado de vincular os veículos operacionais aos incidentes enviados pelos cidadãos.

Figura 16: Tela de monitoramento dos incidentes pela central.

Fonte: Arquivo do Autor.

Ao clicar no botão “Encaminhar viatura”, é aberto um *pop-up* para a escolha dos veículos a serem enviadas para a ocorrência selecionada, como demonstra a figura 17.

Figura 17: Pop-up para encaminhamento de viaturas ao incidente.



Fonte: Arquivo do Autor.

6.5 Web Service

O Web Service do projeto é responsável por fazer a comunicação entre as diferentes linguagens de programação utilizadas, além de processar as mensagens e retornar os dados solicitados pelos sistemas clientes.

6.6 Banco de dados

O banco de dados atua como responsável por armazenar os dados dos incidentes, das viaturas e dos usuários. Como sistema gerenciador de banco de dados foi utilizado o SQL Server para coordenar as inserções, alterações, exclusões e busca de dados.

6.7 Hardware

No início do desenvolvimento do sistema, os *softwares* foram instalados em máquinas pessoais.

Os *softwares* hospedados nos dispositivos móveis foram testados nos seguintes aparelhos:

- Celular Moto E com 1GB de memória RAM, capacidade de armazenamento de 8GB, processador *Quad-Core* de 1.2 GHz e resolução de 540 x 960 *pixels*;

- Celular LG L70 com 1GB de memória RAM, capacidade de armazenamento de 6GB, processador Dual Core 1.2 GHz e resolução de 400 x 800 *pixels*;

- *Tablet* Positivo YPY com 1GB de memória RAM, capacidade de armazenamento de 16GB, processador Cortex A9 1GHz e resolução de 600 x 1024 *pixels*.

Além dessas configurações, todos os dispositivos móveis possuem WI-Fi, GPS e câmera, componentes necessários para que os *softwares* funcionem corretamente.

O Web Service, o banco de dados e a aplicação Web ficaram hospedados em um *notebook* Asus K45VM-VX105H, com processador Core i7, memória RAM de 8GB, HD 750GB, com placa de vídeo GeForce 630M.

A partir do momento em que os *softwares* estavam com sua primeira versão finalizada, os programas instalados no *notebook* passaram a estar em um servidor na nuvem adquirido da empresa BH Servers, a qual disponibilizou um servidor com 2GB de memória RAM, processador Intel Xeon v2 e HD de 40GB.

7 TRABALHOS CORRELATOS

7.1 Firecast

Firecast é um aplicativo desenvolvido com o intuito de possibilitar a comunicação entre a central de operações e a guarnição, disponibiliza detalhes sobre determinado incidente e possíveis rotas para a viatura encaminhada pela central. Atualmente, o aplicativo está sendo utilizado nos municípios de Barra Velha, Canoinhas, Balneário Camboriú, São José e Florianópolis (CBMSC, 2016b).

O diferencial do presente projeto é possibilitar a sociedade a interagir com a central não apenas pelo telefone 193, mas por um aplicativo disponível ao cidadão, podendo acompanhar o andamento do incidente automaticamente através do *software*, ou seja, saber quando uma viatura for encaminhada para a ocorrência e a partir de então receber atualizações sobre sua posição.

8 LIMITAÇÕES DA PESQUISA

O projeto limita-se na troca de informação referente ao incidente desde a solicitação do cidadão até a chegada da viatura ao local, informações posteriores não serão abordadas.

Além disso, há inúmeros dispositivos que usam o sistema operacional Android, consequentemente os aparelhos possuem diversas resoluções, onde algumas serão testadas no simulador fornecido pelo Android Studio, porém em dispositivos físicos serão testados apenas nos aparelhos mencionados no tópico *hardware*.

Também não será desenvolvido envio de áudio e vídeo por falta do suporte de rede para os dispositivos móveis.

Referente ao atendimento social, o sistema possuirá a limitação para Lages - SC e região.

No tocante ao *hardware*, o GPS do celular não possui exata precisão da localização do dispositivo, podendo variar em alguns metros.

Com o auxílio da metodologia Scrum, as funcionalidades dos sistemas foram limitadas até a quantidade máxima suportada pelas *sprints* até o prazo final da entrega do trabalho, demais ações dos *softwares* serão abordadas em projetos futuros.

9 RESULTADOS

Após a criação do primeiro protótipo apresentado aos oficias do corpo de bombeiros, foi dado o início a criação do *software* para aplicação final.

Com a aplicação do sistema foi obtido melhora na compreensão de todos os envolvidos no incidente pelo auxílio na distribuição da informação fornecido pelo sistema.

Em relação à comunidade, foi possível reduzir as ligações referentes ao mesmo incidente através do acompanhamento por parte do cidadão da viatura encaminhada a atendê-lo, como também pelo sistema ter a capacidade de identificar tentativas de envios do mesmo incidente.

No tocante a central de operações, tornou-se mais simples o gerenciamento das viaturas e incidentes, além de proporcionar um melhor acompanhamento das ocorrências.

A respeito do sistema localizado nas viaturas, foi proporcionado melhora no conhecimento prévio da situação do incidente, visto que o sistema disponibiliza o local exato, assim como a imagem fornecida pelo cidadão. Além disso, a rota criada a partir do local atual da viatura até o incidente auxiliou no deslocamento dos veículos até o local.

REFERÊNCIAS

AGUILAR, L. J. **Fundamentos de programação:** Algoritmos, Estruturas de dados e Objetos. 3. ed. São Paulo: Mcgraw-hill, 2008.

AHMAD, G.; SOOMRO, T. R.; BROHI, M. N. XSR: Novel Hybrid Software Development Model (Integrating XP, Scrum & RUP). **International Journal Of Soft Computing And Engineering.** Dubai, Karachi, p. 126-130. Apr. 2014. Disponível em: <http://ijsce.org/attachments/File/v4i2/B2228054214.pdf>. Acesso em: 28 abr. 2016.

ALMEIDA, M. B. Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares. **Ci. Inf.**, Brasília , v. 31, n. 2, p. 5-13, Ago. 2002 . Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652002000200001&lng=en&nrm=iso>. Acesso em: 02 abr. 2016.

ANDROID DEVELOPER. **Android Studio:** The Official IDE for Android. Disponível em: <http://developer.android.com/intl/pt-br/sdk/index.html>. Acesso em: 06 mai. 2016.

ANICHE, M. **Test-Driven Development:** Teste e Design no Mundo Real. São Paulo: Casa do Código, 2012.

BARRETT, D.; KING, T. **Redes de computadores.** Rio de Janeiro: LTC, 2010.

BECK, K et al. **Manifesto para desenvolvimento ágil de software**. 2001. Disponível em: <http://www.manifestoagil.com.br/>. Acesso em: 25 abr. 2016.

BECK, K. **TDD:** Desenvolvimento Guiado por Testes. São Paulo: Bookman, 2010.

BERNERS-LEE, T. **The WorldWideWeb browser.** Disponível em: <https://www.w3.org/People/Berners-Lee/WorldWideWeb.html>. Acesso em: 30 abr. 2016.

BLANKENSHIP, Ed et al. **Professional Team Foundation Server 2012.** John Wiley & Sons, 2012.

BONAN, A. R. **Java:** Fundamentos, Prática e Cerficação. Rio de Janeiro: Alta Books, 2008.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** Guia do usuário. Rio de Janeiro: Elseiver, 2006.

BORGES, E. N. **Conceitos e Benefícios do Test Driven Development**. Universidade Federal do Rio Grande do Sul / Instituto de Informática, 2006. Disponível em: <http://www.inf.ufrgs.br/~cesantin/TDD-Eduardo.pdf>. Acesso em: 04 abr. 2016.

CANALYS. **Media alert:** Over 1.5 billion smart phones to ship worldwide in 2016. Disponível em: <http://www.canalys.com/newsroom/media-alert-over-15-billion-smart-phones-ship-worldwide-2016>. Acesso em: 27 mar. 2016.

CARVALHO, M. S. R. M. de. **A trajetória da internet no brasil: do surgimento das redes de computadores à instituição dos mecanismos de governança.** 2006. 259 f. Dissertação (Mestrado) - Curso de Ciências de Engenharia de Sistemas e Computação, Universidade

Federal do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <http://www.cos.ufrj.br/uploadfile/1430748034.pdf>. Acesso em: 30 abr. 2016.

CBMSC – Corpo de Bombeiros Militar de Santa Catarina. **História do Corpo de Bombeiros**. Disponível em: < https://portal.cbm.sc.gov.br/index.php/historia>. Acesso em: 28 mar. 2016a.

CBMSC – Corpo de Bombeiros Militar de Santa Catarina. **Aplicativo FireCast.** Disponível em: <http://www.cbm.sc.gov.br/ccecobom/index.php/aplicativo-e-bombeiro>. Acesso em: 23 mai. 2016b.

CERN. **The birth of the web.** Disponível em: <http://home.cern/topics/birth-web>. Acesso em: 30 abr. 2016.

COMER, D. E. **Interligação de redes com TCP/IP.** 5. ed. Rio de Janeiro: Elseiver, 2006.

COMER, D. E. **Redes de computadores e internet.** 4. ed. Porto Alegre: Bookman, 2007.

CORPO DE BOMBEIROS PARANÁ. **História do Corpo de Bombeiros no Mundo.** Disponível em: <http://www.bombeiros.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1>. Acesso em: 28 fev. 2016a.

CORPO DE BOMBEIROS PARANÁ. **Histórico do Corpo de Bombeiros no Brasil**. Disponível em: <http://www.bombeiros.pr.gov.br/modules/conteudo/conteudo.php?conteudo=2>. Acesso em: 28 fev Bookman. 2016b.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos:** Conceitos e projetos. 4. ed. São Paulo: Bookman, 2007.

DEITEL, P. et al. **Android para programadores:** Uma Abordagem Baseada em Aplicativos. Porto Alegre: Bookman, 2013. 512 p.

DIAS, J. A. S.; BORGES, C. L. T. Modelo orientado a objetos para avaliação da confiabilidade composta por simulação monte carlo com representação da geração eólica. **Revista Controle & Automação**, v. 20, n. 3, p.359-372, jul./set. 2009. Disponível em: <http://www.scielo.br/pdf/ca/v20n3/07.pdf>. Acesso em: 18 abr. 2016.

FARINELLI, F. **Conceitos básicos de programação orientada a objetos**. p. 16-23, 2007. Acesso em 20 abr. 2016. Disponível em: <http://www.marcelohsantos.com.br/aulas/downloads/2Semestre_2013/poo/Apostila_Java_B.pdf>

FOWLER, M. **Refatoração**: Aperfeiçoando o projeto de código existente. Porto Alegre: Bookman, 2004.

GIBBS, Graham. **Analise de dados qualitativos.** Porto Alegre: Artmed, 2009.

GODOY NETO, M. **XDTv:** um Método Ágil para o Desenvolvimento de Aplicações para TV Digital. 2014. 229 f. Tese (Doutorado) - Curso de Ciência da Computação, Universidade

Federal de Pernambuco, Recife, 2014. Disponível em: <http://repositorio.ufpe.br/bitstream/handle/123456789/12173/TESE Mário Godoy Neto.pdf?sequence=1&isAllowed=y>. Acesso em: 24 abr. 2016.

GUEDES, G. T.a. **UML 2:** Guia prático. São Paulo: Novatec, 2007.

GUERRA, E. **Design patterns com Java:** Projeto orientado a objetos guiado por padrões. São Paulo: Casa do Código, 2013.

HAUGHEE, Eric. **Instant Sublime Text Starter.** Birmingham: Packt Publishing, 2013.

IDC. **Smartphone OS Market Share, 2015 Q2.** 2015. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 01 mai. 2016.

JSON. **Introducing JSON.** Disponível em: <http://www.json.org/>. Acesso em: 03 abr. 2016.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a Internet:** Uma abordagem top-down. 3. ed. São Paulo: Pearson, 2006

LACERDA, F. C. de. **Uma proposta de arquitetura para o protocolo NETCONF sobre SOAP.** 2007. 154 f. Tese (Doutorado) - Curso de Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2007. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000431499>. Acesso em: 26 mar. 2016.

LECHETA, R. R. **Google Android:** Aprenda a Criar Aplicações para Dispositivos Móveis com o Android SDK. 3. ed. São Paulo: Novatec, 2013.

LIBARDI, P. L. O.; BARBOSA, V. **Métodos Ágeis.** 2010. 35 f. Monografia (Especialização) - Curso de Computação, Universidade Estadual de Campinas, Limeira, 2010. Disponível em: <http://www.ft.unicamp.br/liag/Gerenciamento/monografias/monogafia_metodos_ageis.pdf>. Acesso em: 24 abr. 2016.

LODDI, S. A. et al. Metodologias Ágeis: Um exemplo de aplicação da eXtreme Programming (XP). **Fasci-tech,** São Caetano do Sul, v. 1, n. 3, p.163-177, jul./dez. 2010. Disponível em: <http://fatecsaocaetano.edu.br/fascitech/index.php/fascitech/article/view/34>. Acesso em: 27 abr. 2016.

MACHADO JUNIOR, R. R. **Desenvolvimento de um middleware para comunicação via web services e sua aplicação em sistemas de aquisição de dados industriais**. 2014. 104 f. Dissertação (Mestrado em Automação e Sistemas; Engenharia de Computação; Telecomunicações) - Universidade Federal do Rio Grande do Norte, Natal, 2014.

MAGRI, J. A. Criando e usando web service. **Augusto Guzzo Revista Acadêmica**, São Paulo, n. 11, p. 166-183, jun. 2013. ISSN 2316-3852. Disponível em: <http://www.fics.edu.br/index.php/augusto_guzzo/article/view/160>. Acesso em: 25 Mar. 2016.

MARÇAL, A. S. C. **SCRUMMI:** Um processo de gestão ágil baseado no SCRUM e aderente ao CMMI. 2009. 205 f. Dissertação (Mestrado) - Curso de Informática Aplicada, Universidade de Fortaleza, Fortaleza, 2009. Disponível em: <http://www.cin.ufpe.br/~if720/downloads/SCRUMMI - AnaSofia.pdf>. Acesso em: 24 abr. 2016.

MARCONI, M. de A.; LAKATOS, E. M. **Fundamentos de metodologia científica.** 7. ed. São Paulo: Atlas, 2010.

MELO, A. C. **Desenvolvendo Aplicações com UML 2.2:** Do conceitual à implementação. 3. ed. Rio de Janeiro: Brasport, 2010.

MICROSOFT. **Polymorphism (C# Programming Guide).** Disponível em: <https://msdn.microsoft.com/pt-br/library/ms173152.aspx>. Acesso em: 23 abr. 2016a.

MICROSOFT. **An Introduction to JavaScript Object Notation (JSON) in JavaScript and .NET.** Disponível em: < https://msdn.microsoft.com/en-us/library/bb299886.aspx>. Acesso em: 03 abr. 2016b.

MICROSOFT. **Visual Studio IDE.** Disponível em: <https://msdn.microsoft.com/en-us/library/dn762121(v=vs.140).aspx>. Acesso em: 07 mai 2016c.

MÜLLER NETO, G. U. **Métodos tradicionais versus ágeis:** Um estudo comparativo através do trainingcad. 2009. 58 f. Monografia (Especialização) - Curso de Sistemas de Informação, Universidade de Pernambuco, Caruaru, 2009. Disponível em: <http://www.cin.ufpe.br/~gumn/files/monography>. Acesso em: 24 abr. 2016.

MUNDACA, I. L.; ABARCA, M. V. Método ágil híbrido para desarrollar software en dispositivos móviles. **Ingeniare. Revista Chilena de Ingeniería,** Arica, v. 23, n. 3, p.473-488, set. 2015. Disponível em: <http://www.scielo.cl/pdf/ingeniare/v23n3/art16.pdf>. Acesso em: 28 abr. 2016.

MUSHTAQ, Z.; QURESHI, M. R. J. Novel Hybrid Model: Integrating Scrum and XP. **International Journal Of Information Technology And Computer Science,** Lahore, Pakistan, v. 4, n. 6, p.39-44, 1 jun. 2012. MECS Publisher. http://dx.doi.org/10.5815/ijitcs.2012.06.06. Disponível em: < http://www.mecs-press.org/ijitcs/ijitcs-v4-n6/IJITCS-V4-N6-6.pdf>. Acesso em: 24 abr. 2016.

NARDON, F. B. N. **Utilizando XML para representação de informação em Saúde.** Unidade de Pesquisa e Desenvolvimento, Instituto do Coração do Hospital das Clínicas da Faculdade de Medicina da USP, São Paulo, 2002. Disponível em: <http://www.tridedalo.com.br/fabiane/publications/XML-SBISNews.pdf >. Acesso em: 03 abr. 2016.

OPEN HANDSET ALLIANCE. **Building a better phone for consumers.** Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 28 fev. 2016.

PACHECO JÚNIOR, M. A.; CASTRO, R. de O. Um estudo de caso da plataforma Android com Interfaces Adaptativas. **Alumni**, Itu, v. 1, n. 1, p.51-66, jul. 2011. Disponível em:

<http://www.ceunsp.edu.br/revistas/alumni/revistaalumni/artigos/Artigo_Marco Antonio.pdf>. Acesso em: 27 mar. 2016.

PAULA FILHO, W. de P. **Engenharia de software:** Fundamentos, métodos e padrões. 2. ed. Rio de Janeiro: Ltc, 2005.

POLÍCIA MILITAR DO ESTADO DE SÃO PAULO. **História do Corpo de Bombeiros**. Disponível em: <http://www.corpodebombeiros.sp.gov.br/internetcb/site/corpo_bombeiros.php>. Acesso em: 5 mar. 2016.

SÁ-SILVA, J. R.; ALMEIDA C. D. de; GUIDANI, J. F. Pesquisa documental: pistas teóricas e metodológicas. Em: **Revista Brasileira de História e Ciências Sociais.** Ano 1, n. 1, jul. 2009, 15 p. Disponível em: <http://www.unisc.br/portal/upload/com_arquivo/pesquisa_documental_pistas_teoricas_e_metodologicas.pdf>. Acesso em: 15 mai. 2016

SBROCCO, J. H. T. de C. **UML 2.3:** Teoria e prática. São Paulo: Érica, 2011.

SCHACH, S. R. **Engenharia de software:** Os paradigmas clássico e orientado a objetos. 7. ed. São Paulo: Mcgraw-hill, 2009.

SEBESTA, R. W. **Conceitos de linguagens de programação.** 9. ed. Porto Alegre: Bookman, 2011.

SILVA, M. S. **HTML 5**: A Linguagem de Marcação que Revolucionou a Web. São Paulo: Novatec, 2011.

STALLINGS, W. **Redes e sistemas de comunicação de dados:** Teoria e aplicações corporativas. 5. ed. Rio de Janeiro: Elsevier, 2005.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas distribuídos:** Princípios e paradigmas. 2. ed. São Paulo: Pearson Prentice Hall, 2007. 402 p

TELES, V. M. **Extreme Programming:** Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. 2. ed. São Paulo: Novatec, 2014.

TUCKER, A. B.; NOONAN, R. E. **Linguagens de programação:** Princípios e paradigmas. 2. ed. São Paulo: Mcgraw-hill, 2008.

W3C. **Extensible Markup Language (XML) 1.0 (Fifth Edition).** 2008. Disponível em: <https://www.w3.org/TR/REC-xml/>. Acesso em: 03 abr. 2016.

WINDER, R.; ROBERTS, G. **Desenvolvendo Software em Java.** 3. ed. Rio de Janeiro: Ltc, 2009. 716 p.

APÊNDICE A – Classe de recepção de evento de cancelamento de incidente

```
public class CancelIncidentEvent : EventBase

   {
      private CancelIncidentDao incCancDao;

      public CancelIncidentEvent( ComponentContext context) : base(context)
      {
         incCancDao = new CancelIncidentDao(context.DataAccess);
      }

      public override ResponseBase Process()
      {
         Logger.Info("Iniciando cancelamento do incidente");

         try
         {
            var parameters = Body.Parameters.FromJson<IncidentCancelationParams>();

            Validate(incCancDao, parameters.IncidentId);

            incCancDao.CancelIncident(parameters);

            Logger.Info("incidente com o id: {0} - cancelado com sucesso",
parameters.IncidentId);

            return new IncidentCancelationResponse((int)ComponentState.SUCCESS,
"Incidente cancelado com sucesso.");
         }
         catch (EventException ex)
         {
            Logger.Info("Não foi possível cancelar o incidente");
            Logger.Info("Não foi possível cancelar o incidente: {0}", ex.Message);

            return new IncidentCancelationResponse((int)ComponentState.ERROR,
ex.Message);
         }
         catch (Exception ex)
         {
            Logger.ErrorInfo("Ocorreu um erro ao cancelar o incidente");
            Logger.Error("Ocorreu um erro ao cancelar o incidente: {0}", ex.ToString());

            return new IncidentCancelationResponse((int)ComponentState.ERROR, "Ocorreu
um erro ao cancelar o incidente.");
         }
      }
   }
```

APÊNDICE B – Classe de recepção de evento de encaminhamento de viaturas

```
public class ForwardEvent : EventBase
  {
    ForwardDao _dao;

    public ForwardEvent(ComponentContext context) : base(context)
    {
      _dao = new ForwardDao(context.DataAccess);
    }

    public override ResponseBase Process()
    {
      Logger.Info("Iniciando processo de encaminhamento de viatura");

      try
      {
        ForwardModel model = Body.Parameters.FromJson<ForwardModel>();

        Validate(_dao, model.IncidentId);

        Logger.Info("Iniciando execução da query no banco de dados");

        _dao.Forward(model);

        Logger.Info("Encaminhamento da(s) viatura(s) com o(s) id(s): {0} para o incidente
com o id: {1} realizado com sucesso"
          , String.Join(",", model.TeamMarkerIds), model.IncidentId);

        return new ForwardResponse((int)ComponentState.SUCCESS, "Viatura(s)
encaminhadas com sucesso.");
      }
      catch (EventException ex)
      {
        Logger.Info("Não foi possível encaminhar a viatura");
        Logger.Info("Não foi possível encaminhar a viatura: {0}", ex.Message);

        return new ForwardResponse((int)ComponentState.ERROR, ex.Message);
      }
      catch (Exception ex)
      {
        Logger.ErrorInfo("Ocorreu um erro ao encaminhar a viatura");
        Logger.Error("Ocorreu um erro ao encaminhar a viatura: {0}", ex.ToString());

        return new ForwardResponse((int)ComponentState.ERROR, "Ocorreu um erro ao
encaminhar a(s) viatura(s).");
      }
    }
  }
```

APÊNDICE C – Classe de recepção de evento de atualização de incidentes

```
 public class UpdateIncidentEvent : EventBase
  {
     private UpdateIncidentDao _dao;

     public UpdateIncidentEvent(ComponentContext context) : base(context)
     {
        _dao = new UpdateIncidentDao(Context.DataAccess);
     }

     public override ResponseBase Process()
     {
        Logger.Info("Iniciando processo de atualização de incidente");

        try
        {
           UpdateIncidentParams model =
Body.Parameters.FromJson<UpdateIncidentParams>();

           Validate(_dao, model.IncidentId);

           if (model.ShouldUpdateImage())
           {
              Logger.Info("Iniciando inserção de nova imagem para o incidente : {0}",
model.IncidentId);

              _dao.InsertImage(model);

              Logger.Info("Inserção de nova imagem realizada com sucesso");
           }

           if (model.ShouldUpdateDescription())
           {
              Logger.Info("Iniciando atualização da descrição incidente: {0}",
model.IncidentId);

              _dao.UpdateDescription(model.IncidentId, model.Description);

              Logger.Info("Atualização da descrição realizada com sucesso");
           }

           if (model.ShouldUpdatePosition())
           {
              Logger.Info("Iniciando atualização da localização do incidente: {0}",
model.IncidentId);

              _dao.UpdateLocation(model);
```

```
                Logger.Info("Atualização da localização realizada com sucesso");
            }

            Logger.Info("Processamento referente a atualização do incidente realizado com
sucesso");

            return new UpdateIncidentResponse((int)ComponentState.SUCCESS, "Incidente
atualizado com sucesso");
        }
        catch (EventException ex)
        {
            Logger.Info("Não foi possível atualizar o incidente");
            Logger.Info("Não foi possível atualizar o incidente: {0}", ex.Message);

            return new UpdateIncidentResponse((int)ComponentState.ERROR, ex.Message);
        }
        catch (Exception ex)
        {
            Logger.ErrorInfo("Ocorreu um erro ao atualizar o incidente");
            Logger.Error("Ocorreu um erro ao atualizar o incidente: {0}", ex.ToString());

            return new UpdateIncidentResponse((int)ComponentState.ERROR, "Ocorreu um
erro ao atualizar o incidente.");
        }
    }
}
```

APÊNDICE D – Classe dos aplicativos *mobiles* responsável por requisitar o Web Service

```
public class GmsWebService
{
   Gson _gson = new Gson();

   public CrossTalkResponse sendMessage(String message)
   {
      BaseServerConfig config = null;
      try
      {
         config = BaseServerConfig.getInstance(ConfigurationType.LOCAL);

      } catch (Exception e)
      {
         e.printStackTrace();
      }
      try
      {
         SoapObject soap = new SoapObject(config.getNamespace(), "SendMessage");

         soap.addProperty("message", message);

         SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);

         envelope.setOutputSoapObject(soap);

         envelope.dotNet = true;

         HttpTransportSE httpTransportSe = new HttpTransportSE("http://"
               + config.getServerIp()+ "/"
               + config.getPath()
               + "/" + config.getAsmxName() + ".asmx");

         httpTransportSe.debug = true;

         httpTransportSe.call(config.getNamespace() + "SendMessage", envelope);

         Object result = envelope.getResponse();

         return _gson.fromJson(result.toString(), CrossTalkResponse.class);
      } catch (Exception ex)
      {
         return new CrossTalkResponse(ResponseStateConst.ERROR
               , "Conexão perdida, tentando reconectar...", null);
      }
   }
}
```

APÊNDICE E – XML da tela de login do aplicativo FireWorker

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:app="http://schemas.android.com/apk/res-auto"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:weightSum="1">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical">

        <EditText
            android:id="@+id/txtUser"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_column="0"
            android:layout_marginTop="0dp"
            android:layout_row="0"
            android:layout_weight="1"
            android:ems="10"
            android:hint="Usuário"
            android:inputType="textPersonName"/>

        <EditText
            android:id="@+id/txtPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_below="@+id/txtUser"
            android:layout_column="0"
            android:layout_marginTop="0dp"
            android:layout_row="1"
            android:layout_weight="1"
            android:ems="10"
            android:hint="Senha"
            android:inputType="textPassword"/>


        <com.dd.CircularProgressButton
            android:id="@+id/btnLogin"
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:textColor="#ffffff"
            android:textSize="18sp"
            app:cpb_cornerRadius="48dp"
            app:cpb_selectorIdle="@drawable/pb_idle_state_selector"
            app:cpb_selectorComplete="@drawable/pb_complete_state_selector"
            app:cpb_selectorError="@drawable/pb_error_state_selector"
            app:cpb_textComplete="@string/loginComplete"
            app:cpb_textError="@string/loginError"
            app:cpb_textIdle="@string/loginDefault"
            android:layout_weight="2"
            />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="8"
            android:orientation="vertical">

        </LinearLayout>

    </LinearLayout>

</LinearLayout>
```

APÊNDICE F – XML da tela principal do aplicativo FireHunter

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:app="http://schemas.android.com/apk/res-auto"
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1">

        <Spinner
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/typeSpiner"
            android:layout_weight="5"/>
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="9">

        <EditText
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:inputType="text|textMultiLine"
            android:ems="10"
            android:textSize="25dp"
            android:id="@+id/txtDescription"
            android:hint="Descrição do incidente"
            android:padding="0dp"
            android:layout_margin="0dp"
            android:gravity="top"/>
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical|center_horizontal"
        android:layout_weight="5">
        <at.markushi.ui.CircleButton
            android:id="@+id/btnCloseApp"
            android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:layout_gravity="left"
            android:gravity="left"
            android:src="@android:drawable/ic_menu_save"
            app:cb_color="@android:color/holo_blue_dark"
            android:layout_weight="100"/>

        <at.markushi.ui.CircleButton
        android:id="@+id/btnSendIncident"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@android:drawable/stat_sys_upload_done"
        android:layout_gravity="right"
        android:gravity="right"
        app:cb_color="#99CC00"
        android:layout_weight="50"/>
    </LinearLayout>

</LinearLayout>
```