UNIVERSIDADE FEDERAL FLUMINENSE
MESTRADO PROFISSIONAL EM DEFESA E SEGURANÇA CIVIL

LEONARDO ABREU DE BARROS

ESTUDO DE ESTRATÉGIAS INFORMATIZADAS DE BAIXO CUSTO APLICADAS À DEFESA CIVIL NO MUNICÍPIO DE NITERÓI/RJ EM PREVENÇÃO DE DESASTRES

Niterói – RJ

2016

LEONARDO ABREU DE BARROS

ESTUDO DE ESTRATÉGIAS INFORMATIZADAS DE BAIXO CUSTO APLICADAS À DEFESA CIVIL NO MUNICÍPIO DE NITERÓI/RJ EM PREVENÇÃO DE DESASTRES

Trabalho de Conclusão de Curso modalidade "Artigo" apresentado ao Curso de Mestrado Profissional em Defesa e Segurança Civil da Universidade Federal Fluminense como requisito parcial para obtenção de grau de Mestre em Defesa e Segurança Civil. Área de concentração: Planejamento e Gestão de Eventos Críticos. Linha de Pesquisa: Instrumentos de gestão para a redução de riscos de desastres.

Orientador: Prof. Dr. Airton Bodstein

Co-orientadora: Profª Drª Mônica de Aquino Galeano Massera da Hora

Niterói – RJ

2016

ESTUDO DE ESTRATÉGIAS INFORMATIZADAS DE BAIXO CUSTO APLICADAS À DEFESA CIVIL NO MUNICÍPIO DE NITERÓI/RJ EM PREVENÇÃO DE DESASTRES

Trabalho de Conclusão de Curso modalidade "Artigo" apresentado ao Curso de Mestrado Profissional em Defesa e Segurança Civil da Universidade Federal Fluminense como requisito parcial para obtenção de grau de Mestre em Defesa e Segurança Civil. Área de concentração: Planejamento e Gestão de Eventos Críticos. Linha de Pesquisa: Instrumentos de gestão para a redução de riscos de desastres.

Aprovado em:

Banca Examinadora

_____

Professora Mônica de Aquino Galeano Massera da Hora, D.Sc.

Universidade Federal Fluminense – UFF

_____

Professor Reiner Rosas, D.Sc.

Universidade Federal Fluminense – UFF

_____

Professor Marcelo Tílio Monteiro de Carvalho, D.Sc.

Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

XXXX Barros, Leonardo Abreu de

Estudo de Estratégias Informatizadas de Baixo Custo aplicadas à Defesa Civil no Município de Niterói/RJ em Prevenção de Desastres / Leonardo Abreu de Barros, 2016.

107 p.

Orientador: Prof. Airton Bodstein, D.Sc

Trabalho de Conclusão de Curso - Mestrado Profissional em Defesa e Segurança Civil da Universidade Federal Fluminense, Niterói, 2016.

1. defesa civil. 2. Município de Niterói. 3. Redução de riscos de desastres (RRD). 4. *software* livre.

DEDICATÓRIA

Para aqueles que fizeram do seu ofício arriscar suas vidas para salvarem desconhecidos.

AGRADECIMENTOS

Agradeço a meus pais Angela e Airton pelo apoio ao longo de todo o trabalho, mas principalmente, pelo apoio ao longo de toda a minha vida.

Sou muito grato à minha esposa Paula, pelas revisões, encorajamento nos momentos de desânimo e apoio com nossos meninos enquanto minha mente estava ausente, às voltas com este trabalho.

Obrigado a meus queridos filhotes Gabriel e Davi, por compreenderem que papai não estava todo dia disponível para contar estórias.

Agradeço à minha colega e amiga Lúcia Morais por suas revisões e dicas sempre muito pertinentes.

Sou muito grato ao meu colega e amigo Walace Medeiros, a Nathalia Moura, Felipe e demais profissionais da Defesa Civil de Niterói pela parceria entusiasta na realização deste estudo.

Muito obrigado aos Professores Reiner Rosas, Mônica da Hora e Marcelo Tílio por suas valiosas sugestões na qualificação e defesa.

Agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste projeto.

RESUMO

O presente trabalho representa a memória de um artigo, elaborado com base em um estudo de caso, a partir do qual foi elaborado um produto. A finalidade do estudo consiste em fornecer uma contribuição às defesas civis municipais brasileiras, de modo a torná-las mais eficientes e eficazes, por meio de soluções informatizadas simples e de baixo custo aplicadas à gestão e tratamento de dados no âmbito da redução de riscos de desastres (RRD). Foi feito um piloto junto à Defesa Civil Municipal da cidade de Niterói, RJ no qual foram avaliados os principais processos do órgão e sugeridas soluções de otimização. Duas dessas soluções, consideradas prioritárias, foram implementadas pelo autor: para o mapeamento de informações úteis em prevenção, preparação e resposta a emergências, foi experimentado e utilizado um software gratuito, o Google My Maps®, uma versão editável e um pouco menos conhecida do Google Maps®; já para a automatização do acompanhamento dos níveis de pluviosidade e emissão de alertas na ocorrência de níveis críticos foi aplicado um software desenvolvido pelo autor, tendo ambas as soluções apresentado resultados bastante satisfatórios em termos de ampliação da gama de informações relevantes disponíveis e otimização de recursos. O artigo foi submetido e aceito no I Congresso Brasileiro de Redução de Riscos de Desastres.

Palavras Chave: defesa civil, município de Niterói, redução de riscos de desastres (RRD), software livre

ABSTRACT

This work represents the memory of a paper, written based upon a case study, in which a product was developed. The study aims to provide a contribution to the municipal civil defenses, in order to turn them more efficient and effective through low cost IT solutions applied to the management and processing of data in the context of disaster risk reduction (DRR). A pilot study was conducted with the Municipal Civil Defense of the city of Niterói, RJ, assessing the main processes of the organization and suggesting optimization solutions. Two of these solutions, considered priority, were implemented by the author: a free software, Google My Maps®, an editable and a little less known version than Google Maps® was tried and used for mapping useful information for prevention, preparedness and response to emergencies; for automating the monitoring of rainfall and alerting in case of critical levels reached, a software developed by the author was applied. Both solutions presented satisfactory results in terms of expanding the range of available relevant information and resource optimization. The paper was submitted and accepted in the I Brazilian Congress of Disaster Risk Reduction.

Keywords: civil defense, Niterói city, disaster risk reduction (DRR), free software

LISTA DE FIGURAS

Figura 1. mapa de análise de risco da cidade de Niterói 24

Figura 2. planilha gerada pelo PLUVI, a partir de dados do *site* do Cemaden 26

Figura 3: estrutura esquemática do sistema 27

Figura 4: estrutura de diretórios do sistema 31

Figura 5: cobertura de testes com EclEmma 37

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação da Aplicação)
BD	Banco de Dados
CEMADEN	Centro Nacional de Monitoramento e Alertas de Desastres Naturais
CENAD	Centro Nacional de Gerenciamento de Riscos e Desastres
CLIN	Companhia de Limpeza de Niterói
COMDEC	Coordenadoria Municipal de Defesa Civil
CONDEC	Conselho Nacional de Defesa Civil
CSV	*Comma Separated Values* (Valores Separados por Vírgulas)
DC	Defesa Civil
DCMN	Defesa Civil Municipal da cidade de Niterói
DIRDN	Década Internacional para Redução dos Desastres Naturais
DRR	*Disaster Risk Reduction* (Redução de Riscos de Desastres)
ESRI	*Environmental System Research Institute* (Instituto de Pesquisas em Sistemas Ambientais)
FTP	*File Transfer Protocol* (Protocolo de Transferência de Arquivos)
FUNCAP	Fundo Especial para Calamidades Públicas
GM	Guarda Municipal
GMT	*Greenwich Meridian Time* (Hora do Meridiano de Greenwich)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
IBGE	Instituto Brasileiro de Geografia e Estatística
INPE	Instituto Nacional de Pesquisas Espaciais
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
IT	*Information Tecnology* (Tecnologia da Informação)

LOC	*Lines Of Code* (Linhas de Código)
MEC	Ministério da Educação
NUDEC	Núcleo Comunitário de Defesa Civil
ONG	Organização Não Governamental
ONU	Organização das Nações Unidas
PIB	Produto Interno Bruto
PM	Polícia Militar
PNDC	Política Nacional de Defesa Civil
PNPDEC	Política Nacional de Proteção e Defesa Civil
PNUD	Programa de Desenvolvimento das Nações Unidas
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
RRD	Redução de Riscos de Desastres
SAMU	Serviço de Atendimento Móvel de Urgência
SIG	Sistema de Informações Geográficas
SINDEC	Sistema Nacional de Defesa Civil
SMTP	*Simple Mail Transfer Protocol* (Protocolo Simples de Transferência de Mensagens)
SMS	*Short Messaging System* (Sistema de Mensagens Curtas)
UFF	Universidade Federal Fluminense
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)

SUMÁRIO

CAPÍTULO I – INTRODUÇÃO 1

OBJETIVOS 6
Objetivo Geral 6
Objetivos Específicos 6
RELEVÂNCIA DO ESTUDO 7
JUSTIFICATIVA 7
ORGANIZAÇÃO DO TRABALHO 8

CAPÍTULO 2 – ASPECTOS METODOLÓGICOS 10

METODOLOGIA 10
PILOTO NA DEFESA CIVIL DE NITERÓI 11
A DEFESA CIVIL MUNICIPAL DE NITERÓI 12
AVALIAÇÃO DOS PROCESSOS ATUAIS 13
Riscos 13
Monitoramento e Alerta 14
Preparação para Emergências 16
Reconstrução 16
Indicadores 16

CAPÍTULO 3 – DISCUSSÃO 19

POSSIBILIDADES DE OTIMIZAÇÃO POR MEIO DE SISTEMAS INFORMATIZADOS 19
Análise de Risco 19
Monitoramento 19
Preparação para Emergências 20
Reconstrução 20
Comunicação 21
SOLUÇÃO 1: SISTEMA DE INFORMAÇÕES GEOGRÁFICAS 22
SOLUÇÃO 2: SISTEMA DE ACOMPANHAMENTO E ALARME (PLUVI) 24
Arquitetura e Dinâmica de Execução 26
Requisitos 28
Tecnologias Utilizadas 29
Requisitos de Hardware e Software 29
Instalação 29
Estrutura de Diretórios 30
Arquivo de Propriedades 31
Leitura de Dados 32
Salvamento de Dados 33
Processamento de Alarmes 33
Envio de Mensagens 33
Tolerância a Falhas nos Dados de Entrada 33
Análise Crítica 34
Tratamento de Exceções 36
Testes Automatizados 36
Modo Offline 37

CAPÍTULO 4 – CONCLUSÕES E RECOMENDAÇÕES 38

SUGESTÕES E TRABALHOS FUTUROS 38
CAPÍTULO 5 - REFERÊNCIAS 40

APÊNDICE I - ARTIGO 42

APÊNDICE II – MANUAL DO SISTEMA 63

GUIA DE INSTALAÇÃO E USO DO SISTEMA PLUVI 64
Requisitos 64
Instalação 64
Uso do sistema 65
APÊNDICE III – CÓDIGO-FONTE 66

Classe Cemaden 67
Classe StationData 86
Classe CemadenTest 89
Classe StationDataTest 93
Classe AllTests 94
Classe Util 94
Arquivo de propriedades config.properties 95

CAPÍTULO I – INTRODUÇÃO

O presente estudo constitui um experimento realizado em parceria com a Defesa Civil de Niterói/RJ, uma contribuição que objetiva tornar as defesas civis municipais mais eficientes, seja aumentando a produtividade do seu efetivo, reduzindo custos e/ou prazos, seja automatizando o trabalho manual e ampliando a gama de informações disponíveis para auxiliar os profissionais desse setor em seu trabalho; em outras palavras, a meta do estudo é otimizar os processos da Defesa Civil Municipal. A expectativa é que a utilização de softwares gratuitos, ou mesmo o desenvolvimento de soluções informatizadas simples e de baixo custo possam contribuir significativamente para a melhoria das ações aplicadas à gestão e tratamento de dados no âmbito da redução de riscos de desastres (RRD).

Após algumas décadas de construção das noções de "redução de riscos" e "gestão de risco de desastres", de aprofundamento teórico e da tentativa de estabelecer um marco de ação específico para a gestão de risco de desastre, uma interpretação mais recente dessa gestão a compreende como a aplicação de políticas, processos e ações para prevenir novos riscos, reduzir os riscos de desastres existentes e manejar os riscos residuais, contribuindo para o fortalecimento da resiliência (UNISDR, 2015). A nova definição de gestão de risco de desastre é mais sintética do que a anterior e corresponde a um processo permanente de análise, planejamento, tomada de decisões e implementação de ações destinadas a identificar, prevenir e reduzir as possibilidades de que um fenômeno potencialmente destrutivo cause danos ou perturbações graves na vida das pessoas, nos meios de subsistência e nos ecossistemas dos territórios, assim como responder adequadamente em caso de impacto e de recuperar meios de vida, serviços e sistemas após a ocorrência do desastre (NARVAEZ et al., 2009; CIF-OIT, 2012).

Os desastres sempre estiveram presentes na História da Humanidade, em uma interpretação livre de textos religiosos e mitológicos. O Grande Dilúvio enfrentado por Noé e a erupção que destruiu as cidades gêmeas de Sodoma e Gomorra são lendas que poderiam perfeitamente ter sido inspiradas em desastres reais que assolaram civilizações muito antigas (TAVARES 2005). Estes eventos de causas naturais e decorrentes de fenômenos geodinâmicos, como sismos, erupções vulcânicas, tsunamis, deslizamentos de terra, inundações, tornados e outros foram ignorados por muito tempo, subestimados e praticamente desprovidos de prevenção. Acreditava-se que as causas dos desastres eram de natureza punitiva e que os seus efeitos eram incontestáveis, fatais e explicitamente determinados. No entanto, o avanço dos

estudos científicos e técnicos permitiu caracterizar tais eventos, prever os seus efeitos, estabelecer e distinguir as causas naturais e antropogênicas, no sentido de melhorar a prevenção e a gestão de riscos.

No contexto dos desastres reportados acima, os tornados constituem um exemplo interessante, já que foram relatados ao longo da história pelos povos de língua inglesa desde 1000 anos DC. As observações de tornados por habitantes dos hoje Estados Unidos da América remontam ao final dos anos 1500, durante aventuras expedicionárias que acabariam por levar à colonização do país. A primeira tempestade de vento relatada pelos americanos do norte data de cinco de julho de 1643, em Plymouth Rock, a qual teve um americano nativo como vítima. Provavelmente não foi um tornado, embora alguns livros o classifiquem como tal (GRAZULIS 2001). Os relatos de tornados se sucederam em 1643, 1671, ou seja, ao longo dos séculos XV, XVI até os dias de hoje. Esses eventos passaram a acontecer com frequência na Nova Inglaterra, principalmente no Vale de Ohio, já nos meados de 1800, onde vários pesquisadores começaram a se preocupar em descrever o fenômeno, e o que se acreditava estar por trás deles, através de livros didáticos, como BROCKLESBY (1851).

Em 1755, Lisboa foi seriamente atingida por tremores de terra importantes, acompanhados de tsunamis e de numerosos incêndios. O número de vítimas foi elevado, em torno de 60.000 mortos; entretanto, esse evento extremo é hoje considerado um marco não somente por sua devastação, mas por ter sido o primeiro desastre a ser descrito de forma objetiva, devido à iniciativa do Marquês de Pombal, então primeiro-ministro e responsável pela reconstrução da cidade. O Marquês instaurou um inquérito e ordenou que fossem distribuídos questionários com questões objetivas (quanto tempo durou o evento, quantas réplicas foram sentidas, que tipo de danos foram causados etc.) para apurar a ocorrência e efeitos do sismo. As respostas permitiram aos cientistas da atualidade reconstituírem o fenômeno sob uma perspectiva científica, razão pela qual este evento é considerado o nascimento da sismologia (POIRIER 2005).

As primeiras ações, estruturas e estratégias de proteção e segurança dirigidas à população, tanto no Brasil como no resto do mundo, foram realizadas nos países envolvidos na Segunda Guerra Mundial. A Inglaterra foi o primeiro país a perceber que somente as forças armadas não poderiam garantir a segurança da sua população e instituiu a *Air Raid Response Service* (Serviço de Resposta a Ataques Aéreos), a primeira instituição oficial de defesa civil de que se tem notícia, após os ataques sofridos entre 1940 e 1941, quando foram lançadas

toneladas de bombas sobre as principais cidades e centros industriais ingleses, causando milhares de perdas de vidas na população civil.

No Brasil, o tema começou a ser tratado em 1942, após o afundamento dos navios militares Baependi, Araraquara e Aníbal Benévolo, no litoral de Sergipe, e do vapor Itagiba, no litoral do estado da Bahia. A notícia dos afundamentos fez com que a população brasileira fosse às ruas, exigindo do governo uma resposta imediata aos ataques, que culminou com a declaração de guerra do Brasil contra a Alemanha e a Itália e a criação do Serviço de Defesa Passiva Antiaérea, em agosto de 1942. Em 1943, a denominação de Defesa Passiva Antiaérea foi alterada para Serviço de Defesa Civil, sob a supervisão da Diretoria Nacional do Serviço da Defesa Civil, do Ministério da Justiça e Negócios Interiores. Esse órgão foi extinto em 1946, com o fim da 2ª Guerra Mundial, bem como suas Diretorias Regionais criadas nos Estados, Territórios e no Distrito Federal.

Em função de fortes chuvas que assolaram a região Sudeste entre 1966 e 1967, provocando enchentes no Estado da Guanabara e deslizamentos na Serra das Araras/RJ e Caraguatatuba/SP, o Brasil começou a se estruturar em Defesa Civil (MIN 2012). Em 1969, foi instituído o Fundo Especial para Calamidades Públicas (FUNCAP), com o objetivo de financiar a recuperação de áreas devastadas.

As décadas de 70 e 80 foram marcadas por uma série de desastres de causas naturais, especialmente na América Latina, o que no Brasil fomentou a criação, em 1988, do Sistema Nacional de Defesa Civil (SINDEC) e do Conselho Nacional de Defesa Civil (CONDEC), para integrar a resposta a eventos extremos que desconsideravam limites territoriais. No campo internacional, a frequência de calamidades públicas levou a Organização das Nações Unidas (ONU) a proclamar a década de 1990 a Década Internacional para Redução dos Desastres Naturais (DIRDN); outra ação importante da ONU, por meio do seu Programa de Desenvolvimento das Nações Unidas (PNUD), foi lançar o conceito de "Segurança Humana" em seu relatório anual, que consistia em uma nova abordagem de segurança em nível individual, a partir de dois aspectos principais: manter as pessoas a salvo de ameaças crônicas como a fome, as doenças, a repressão e protegê-las de mudanças súbitas e nocivas nos padrões da vida cotidiana, como guerras, genocídios e desastres naturais (OLIVEIRA 2005).

Inspirado nessas resoluções, em 1995 o Brasil lançou sua Política Nacional de Defesa Civil (PNDC), a qual representou um marco histórico na Defesa Civil nacional, alinhada com o abrangente conceito de Segurança Humana proposto um ano antes pelo PNUD.

Em 2004, o furacão Catarina causou grandes danos na região sul do Brasil. Um ano depois, outro ciclone de nome semelhante, mas muito mais mortífero, o furacão Katrina, varreu a costa dos Estados Unidos. No mesmo ano de 2005 foram atualizadas a estrutura, a organização e as diretrizes para o funcionamento do SINDEC e do CONDEC, além da implantação do Centro Nacional de Gerenciamento de Riscos e Desastres (CENAD).

Em março de 2010, ano em que ocorreu um terremoto no Haiti que chocou o mundo, foi realizada, em Brasília, a I Conferência Nacional de Defesa Civil e Assistência Humanitária. Com uma participação significativa da sociedade civil organizada, o tema Defesa Civil e Assistência Humanitária foi discutido por 1.179 municípios que realizaram conferências no âmbito municipal, intermunicipal e regional, selecionando 1.495 delegados habilitados para a etapa Nacional, que aprovaram 104 proposições para reformular o sistema de Defesa Civil no país. Nesse mesmo ano, fortes chuvas ocorridas no Estado do Rio de Janeiro em janeiro e abril causaram mais de uma centena de mortos e desabrigados em função das inundações e dos deslizamentos ocorridos, dentre os quais se destaca a tragédia do Morro do Bumba, em Niterói.

Em janeiro de 2011, outro episódio de chuvas intensas assolou a Região Serrana do mesmo Estado, com centenas de perdas humanas e materiais ainda maiores, sendo considerado o pior desastre natural súbito da história do país. Apenas dois meses depois, do outro lado do planeta, o tsunami que atingiu a central nuclear de Fukushima ceifou imediatamente a vida de mais de 15.000 japoneses, sem contar com os milhares de vidas que ainda seriam perdidas nos desdobramentos do desastre.

As dificuldades enfrentadas para mitigar os efeitos das chuvas de 2010/2011 no âmbito Federal impulsionaram alterações na legislação, a fim de facilitar a transferência de recursos para ações de socorro, assistência às vítimas, restabelecimento de serviços essenciais e reconstrução de áreas atingidas por desastre, além de alterações no Fundo Especial para Calamidades Públicas (FUNCAP).

O contexto internacional representado pelo Marco de Ação de Hyogo, as proposições resultantes da Conferência Nacional de Defesa Civil e a magnitude das perdas humanas e materiais decorrentes das chuvas de 2010/2011 no Estado do Rio de Janeiro estabeleceram as bases para a substituição da então Política Nacional de Defesa Civil (PNDC) pela atual Política Nacional de Proteção e Defesa Civil (PNPDEC), por meio da Lei 12.6080/2012, e fomentaram, em julho de 2011 (Decreto n.7513/2011), a criação do Centro Nacional de Monitoramento e

Alertas de Desastres Naturais (CEMADEN), órgão ligado ao Ministério da Ciência, Tecnologia, Inovações e Comunicações.

Como se pode depreender desse histórico, a evolução da política e do sistema nacional de defesa civil sempre foi impulsionada por desastres com grandes perdas materiais e de vidas humanas, os quais geraram grande comoção e propiciaram fortes investimentos. A prioridade para ações preventivas de defesa civil, o envolvimento ativo dos três níveis da administração pública e a integração com as demais políticas setoriais pregada pela ainda recente PNPDEC depende, para se concretizar, de investimentos contínuos e significativos em capacitação e tecnologia; esses investimentos dependem de uma visão de longo prazo dos governantes e em última instância, da percepção de risco da população em geral. Entretanto, a compreensão plena dessa cadeia de causa-efeito depende de mudanças culturais drásticas que só acontecem ao longo de gerações. Para o momento presente, no qual os investimentos contínuos na infraestrutura de defesa civil ainda são escassos, são necessárias alternativas viáveis para perseguir os objetivos almejados pela Lei 12.608/12, daí a motivação para a busca de soluções efetivas que exijam baixo grau de investimento.

Em todo o mundo, durante várias décadas, os governos e as organizações não governamentais (ONGs) se apoiaram em soluções propostas por empresas do setor privado para garantir a coleta e a interpretação de dados de emergências, com a finalidade de responder às situações de crise. No entanto, há alguns anos apenas, tais instituições passaram a se interessar por tecnologias de "código aberto" (*open source*), de menor custo. Em uma entrevista com especialistas do mundo inteiro na área de informática aplicada à gestão de riscos de desastres de causas naturais (IRIN, 2013a), perguntou-se qual seria a combinação ideal de tecnologias para enfrentar tais desastres. Os *softwares* privados atendem a todas as demandas, mas apresentam um custo alto.

A ESRI, antigamente conhecida como *Environmental System Research Institute* ou Instituto de Pesquisa Ambiental, representa uma das mais importantes empresas do setor privado que trabalha com gestão de desastres de causas naturais. Essa empresa, com sede na Califórnia, vende a plataforma ArcGIS (https://www.arcgis.com), que permite criar mapas interativos, graças à tecnologia de satélite. Fundada em 1969, está avaliada em cerca de 900 milhões de dólares por ano e controla cerca da metade das tecnologias SIG (Sistemas de Informações Geográficas). No entanto, no decorrer dos últimos anos e, principalmente depois do terremoto no Haiti em 2010, um grande número de cidades e ONGs se voltaram para a

tecnologia *open source*, tendo em vista que em geral, o uso não está atrelado ao pagamento de uma licença. *Softwares* como os da Fundação Sahana (https://sahanafoundation.org/), baratos, fáceis de instalar/utilizar e poderosos, vêm ganhando espaço na confiança de governos e população. Apesar disso, esta tecnologia apresenta alguns inconvenientes, como a falta de especialistas capacitados para resolver os problemas gerados, bem como o conservadorismo ainda reinante nos governos que preferem os *softwares* privados, (geralmente) melhor elaborados.

Com o avanço da tecnologia, a preparação e prevenção às situações de urgência se transformaram significativamente. Os celulares, os sistemas de informação geográfica (SIG), *Twitter* e outras tecnologias são cada vez mais utilizadas para avisar as comunidades de um risco potencial de crise, para informá-las sobre a maneira de se preparar e para ajudar as instituições governamentais e outros setores a prever o desenvolvimento de situações de risco de desastres (IRIN, 2013a).

OBJETIVOS

Objetivo Geral

- Demonstrar, por meio de um piloto realizado em parceria com a Defesa Civil de Niterói/RJ, que o uso de softwares gratuitos, ou mesmo o desenvolvimento de soluções informatizadas simples e de baixo custo podem contribuir significativamente para tornar as defesas civis municipais mais eficientes, seja aumentando a produtividade do seu efetivo, reduzindo custos e/ou prazos, seja automatizando o trabalho manual e ampliando a gama de informações disponíveis para auxiliar os profissionais desse setor em seu trabalho.

Objetivos Específicos

- Prover uma solução para um Sistema de Informações Geográficas (SIG) que possa mapear informações de interesse da Defesa Civil Municipal para o planejamento, prevenção e resposta a desastres, como delimitação de comunidades vulneráveis, associações de moradores, órgãos de apoio etc.

- Desenvolver um produto para acompanhamento dos níveis de pluviosidade e emissão de alertas em caso do alcance de níveis críticos.

RELEVÂNCIA DO ESTUDO

A Defesa Civil de Niterói/RJ conheceu um aporte grande de recursos humanos, financeiros e materiais após o desastre do Morro do Bumba, como resultado direto da pressão popular sobre a Prefeitura da cidade, em decorrência da comoção gerada pelo desastre. Entretanto, mesmo esse reforço ainda é insuficiente para atender a toda a demanda da população, sem contar com o trabalho preventivo que a DCMN busca desenvolver. Além disso, com o distanciamento temporal da calamidade do Bumba, a percepção de risco da população e da Prefeitura tende a gradualmente diminuir, o que reduz as expectativas de fortes investimentos no setor.

Considerando esses dois aspectos, toda e qualquer iniciativa no sentido de conhecer as dificuldades, identificar as deficiências e melhorar as condições de prevenção, preparação e resposta frente a emergências, principalmente se exigir um baixo nível de investimento, será bem-vinda; e visto que as duas características citadas são comuns nas Defesas Civis Municipais do país, soluções projetadas para a Defesa Civil de Niterói têm grande possibilidade de aplicação em outros municípios.

JUSTIFICATIVA

A experiência adquirida através do curso de Bacharelado em Informática na Universidade Federal do Rio de Janeiro, bem como o diploma de mestrado profissional em Sistemas de Gestão da Universidade Federal Fluminense, Niterói, além do fato de ter nascido e morado na cidade de Niterói foram fatores que estimularam o pesquisador a escolher a Defesa Civil desse município como piloto para experimentar um processo de melhoria assistida. Além disso, o desastre do Morro do Bumba ocorrido em 2010 em Niterói, pôs em evidência a cidade no cenário nacional em razão da tragédia ocorrida, na qual foram questionados vários pontos em relação às causas do acidente, como a incorreta disposição do lixo, a geomorfologia do local, a meteorologia e mesmo a responsabilidade da gestão pública vigente à época (CUNHA, 2016). Seria potencialmente interessante para a administração atual reverter esse quadro e projetar a imagem de uma cidade resiliente a desastres, com uma Defesa Civil atuante, integrada

com os demais órgãos da administração e respaldada pela população. O presente trabalho visa contribuir com esse objetivo.

Em termos mundiais, quando se observa o aumento expressivo na frequência e intensidade dos desastres nas últimas décadas, como consequência das mudanças climáticas e outros fatores, como: aumento da densidade demográfica, ocupação desordenada do solo, provocando alterações pela formação das "ilhas de calor" nas áreas intensamente urbanizadas e potencializando a impermeabilização dos solos etc., por mais preparada que a Defesa Civil de um país esteja para a resposta, esse preparo não será eficaz se não for precedido de um forte investimento em planejamento e prevenção.

Como os desastres atingem os municípios, no caso brasileiro, cabe às Defesas Civis Municipais o primeiro combate, ações estratégicas que precisam ser disponibilizadas com os escassos recursos humanos, financeiros e tecnológicos de que dispõem. Portanto, é válida qualquer iniciativa que possa aumentar a eficiência do uso desses recursos, a um custo viável, como soluções informatizadas que contribuam para a melhoria da gestão municipal.

ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado em cinco capítulos.

O capítulo 1 aborda a apresentação do tema, seus objetivos e relevância.

O capítulo 2 descreve a metodologia utilizada para a elaboração do trabalho, bem como dá detalhes sobre o piloto realizado junto à Defesa Civil de Niterói, apresentando suas características.

O capítulo 3 aborda três esforços empreendidos no sentido de contribuir para a otimização dos processos da DC Niterói: a enumeração e descrição de possibilidades de otimização, assim como a implementação das duas soluções consideradas mais prioritárias: a indicação de um Sistema de Informações Geográficas (SIG) livre e o desenvolvimento de um Sistema para Acompanhamento de Níveis de Pluviosidade e Emissão de Alertas, nomeado neste documento como "Sistema PLUVI" para facilitar o entendimento.

O capítulo 4 aborda os resultados e conclusões deste trabalho.

O capítulo 5 retrata as referências consultadas para o desenvolvimento deste estudo.

No apêndice I consta o texto do artigo aceito no I Congresso Brasileiro de Redução de Riscos de Desastres: “Gestão Integrada em RRD no Brasil e o Marco de SENDAI para a Redução do Risco de Desastres 2015 – 2030”, Curitiba, Paraná, Brasil – 12 a 15 de Outubro de 2016.

O apêndice II traz um manual de uso e instalação do sistema PLUVI para acompanhamento dos níveis de chuva e emissão de alertas.

Por último, o apêndice III apresenta o código-fonte do sistema PLUVI.

CAPÍTULO 2 – ASPECTOS METODOLÓGICOS

METODOLOGIA

A metodologia utilizada é a pesquisa descritiva e exploratória, seguida de um piloto junto à Defesa Civil Municipal da cidade de Niterói, RJ (DCMN). O estudo inicial procurou contemplar o levantamento detalhado da estrutura atual da Defesa Civil de Niterói, priorizando as áreas a serem otimizadas. A partir da experiência do autor na área de desenvolvimento de sistemas informatizados de apoio a emergências e do conhecimento adquirido no Mestrado em Segurança e Defesa Civil da UFF, foi feita uma apresentação, com base em *softwares* para gestão de emergências existentes da indústria, sobre as possíveis áreas de atuação a serem otimizadas, a saber: Análise de Risco, Gestão de Planos, Gestão de Recursos, Gestão de Simulados, Monitoramento, Gestão de Emergências e Comunicação. Em conjunto com a coordenação do órgão, priorizou-se inicialmente a Análise de Risco. Por sua vez, a equipe técnica da prefeitura forneceu informações adicionais sobre a estrutura e funcionamento da DCMN, assim como metas e indicadores que poderiam ser melhorados. É importante salientar que várias reuniões foram realizadas com a finalidade de conhecer melhor as necessidades do órgão.

Com base na análise dos dados levantados, foram consideradas algumas possibilidades de otimização dos processos com o apoio de módulos informatizados - os quais poderiam tanto ser utilizados de forma isolada quanto combinados em um ou mais sistemas - para que estes pudessem ser priorizados. Foi definido como prioritário um Sistema de Informações Geográficas (SIG) para apoio à Análise de Risco e detalhadas as camadas que deveriam ser visualizadas nesse módulo. Após análise das soluções gratuitas disponíveis, foi selecionado o *software* Google My Maps® (https://www.google.com/mymaps), a versão editável e um pouco menos conhecida do Google Maps®, o qual foi utilizado para mapear diversas informações úteis para a Defesa Civil de Niterói.

Com esta primeira fase concluída, foi efetuada nova priorização. O módulo escolhido foi um sistema de alarme para níveis elevados de pluviosidade, da área "Monitoramento". A metodologia atualmente utilizada pela DCMN é o acompanhamento dos níveis de chuva no município, com base principalmente em informações recuperadas no *site* do Cemaden – Centro Nacional de Monitoramento e Alertas de Desastres Naturais, do Ministério da Ciência e Tecnologia (http://www.cemaden.gov.br/mapainterativo), e os critérios para disparar os estados de alerta e vigilância são regidos por protocolo do Governo do Estado do Rio de Janeiro.

Um *software* foi então desenvolvido com base em tecnologias *open source*, para: automatizar a leitura dos dados do *site* (cujos dados são atualizados a cada dez minutos); gerar um histórico das leituras em tempo real; e acompanhar os níveis de chuva e disparar alertas por *e-mail* caso níveis críticos sejam atingidos. O *software* foi demonstrado na sede da Defesa Civil de Niterói e encontra-se atualmente em fase de testes.

PILOTO NA DEFESA CIVIL DE NITERÓI

Conforme apresentado na Metodologia, para ilustrar as possibilidades de ganho em eficácia e eficiência das defesas civis municipais com investimentos em soluções informatizadas de baixo custo, foi escolhida a Defesa Civil do município de Niterói, no estado do Rio de Janeiro.

Segundo dados do IBGE (2016), a estimativa para a população de Niterói era de 496.696 habitantes em 2015, distribuídos em uma área de 133,919 km2. A cidade é considerada um dos principais centros financeiros, comerciais e industriais do Rio de Janeiro, tendo sido indicada pela Revista Exame (2014) como a 6a entre as 100 melhores cidades brasileiras para negócios. Apresenta alto índice de investimentos imobiliário e comerciário. O desenvolvimento atual justifica-se pelo fato do município estar ligado a uma série de investimentos industriais importantes nos setores ligados à cadeia produtiva de petróleo e gás, o que leva a região a responder por 70% do parque instalado do setor, concentrando estaleiros, plataformas e empresas *offshore*, além da construção de embarcações para o transporte de passageiros. O IBGE (2016) indica que o PIB nominal de Niterói é o quinto maior do estado do Rio de Janeiro, só ficando atrás da própria capital e dos municípios de Duque de Caxias, Campos dos Goytacazes e Macaé, além de ser o 45o município mais rico do Brasil. Embora ocupe o 5o lugar quanto ao número de habitantes, o que corresponde a 4,11% do total da população da Região Metropolitana do Rio de Janeiro, Niterói é o segundo maior empregador formal do Estado, apresentando o melhor Índice de Desenvolvimento Humano e o terceiro melhor do país. Além disso, a cidade está entre as mais alfabetizadas do país, sendo o melhor nível de alfabetização do Estado, apresentando também, a menor incidência de pobreza, população com maior renda mensal per capita e o maior índice de longevidade do Rio de Janeiro (IBGE 2016).

A DEFESA CIVIL MUNICIPAL DE NITERÓI

A Defesa Civil Municipal de Niterói conta hoje com aproximadamente quarenta pessoas, divididas entre as áreas de Minimização de Desastres, Operacional, Monitoramento e Administrativo. Segundo o coordenador, esse efetivo é insuficiente para atender à demanda de atendimentos que chegam por telefone, e-mail ou presencialmente, e na maior parte das vezes requerem vistoria presencial; só em 2013, por exemplo, foram 850 pedidos, dos quais 650 foram atendidos. A DCMN conta com um sistema informatizado que permite registrar esses pedidos, mas a priorização é dependente de um operador externo, e por conta do baixo efetivo, acaba sendo deficiente. A meta do órgão é conseguir atender todas as vistorias preventivas em até 15 dias, e as emergenciais em 24h. É importante destacar que esses pedidos abrangem somente as vistorias responsivas, isto é, disparadas em função da demanda da população e não consideram as vistorias preventivas, de responsabilidade do Estado para garantir a segurança dos moradores.

Outro fator limitante para a atuação da DCMN são as delicadas relações interinstitucionais que permeiam a elaboração de planos de emergência. A instituição possui o papel central de planejar e coordenar a resposta dos diferentes atores envolvidos durante a ocorrência de uma situação de emergência, como o corpo de bombeiros, as polícias civil e militar, secretarias municipais e órgãos de trânsito, mas não dispõe de uma ascendência hierárquica sobre essas entidades. Como consequência, para evitar o risco de ingerência nos processos de cada órgão, os planos de contingência do município não podem se aprofundar ao nível de detalhes, necessário para um resultado altamente otimizado, limitando-se a listar as responsabilidades dos envolvidos.

Um exemplo do impacto dessa limitação foi um caso de escorregamento ocorrido em Charitas (bairro de Niterói), em que uma escola ficou com acesso dificultado. A diretoria da escola dependia de instruções da DCMN para saber se deveria reter ou liberar as crianças, mas por sua vez dependia de informações de trânsito atualizadas do órgão responsável; como não havia planos de emergência preparados para este caso e consequentemente não havia protocolos pré-acordados, a resposta demorou muito mais do que seria desejável para a situação em questão. O caso lembra outro incidente ocorrido durante o desastre mais severo da história recente de Niterói, o deslizamento do Morro do Bumba em 2010. Na ocasião, foi solicitada a colaboração de uma empresa que possuía um maquinário pesado, uma retroescavadeira, para a remoção de entulho, a qual se encontrava em um galpão da empresa no Rio de Janeiro. A caminho do atendimento, o veículo foi barrado na entrada da ponte Presidente Costa e Silva,

popularmente conhecida como Ponte Rio - Niterói, que liga as duas cidades, por falta de protocolo prévio. Foram necessárias várias horas e o envolvimento de várias autoridades, inclusive do Governador do Estado, para que fosse permitida a passagem do veículo.

As restrições de pessoal impactam também no pós-desastre, fase na qual a DCMN se envolve para vistoriar os danos e autorizar o restabelecimento de comércio e serviços. Uma meta da DCMN seria poder registrar e tornar mais visíveis os danos secundários, como influências negativas no mercado imobiliário, impacto no comércio, absenteísmo no trabalho, doenças crônicas, etc. Infelizmente, a falta de recursos já dificulta mesmo o levantamento dos dados primários e inviabiliza completamente uma atuação mais abrangente.

Por fim, é importante citar as restrições financeiras da DCMN, comum a todos os órgãos da Defesa Civil. Estas competem pelos recursos orçamentários da prefeitura com todos os demais órgãos, e possuem a desvantagem de não gerar receita. Portanto, a saúde financeira do órgão depende muito da visão do prefeito sobre os benefícios da prevenção, a preocupação com a imagem política, etc. No caso de Niterói, o já citado desastre do Morro do Bumba foi um divisor de águas para a Defesa Civil do município, que contava com um efetivo de quatro pessoas e nenhuma viatura (o coordenador se deslocava em veículo particular). Após o desastre e a visibilidade negativa provocada pelo desastre, o efetivo decuplicou, a DCMN ganhou uma sede nova e muita infraestrutura. Conforme o tempo passa e a lembrança da tragédia fica mais distante, a tendência é que esses recursos tornem a escassear. Portanto, para qualquer solução que busque aperfeiçoar os processos de trabalho da entidade, é preciso priorizar soluções de baixo custo.

AVALIAÇÃO DOS PROCESSOS ATUAIS

Após o levantamento da estrutura atual da Defesa Civil de Niterói (já detalhado na Introdução), deu-se início ao estudo.

A Análise de Risco foi escolhida como a área prioritária a ser otimizada. A justificativa da coordenação para essa escolha partiu do fato de que essa área constitui a base para todas as demais, e que se esta for mal elaborada ou insuficiente, todas as ações seguintes seriam prejudicadas.

Riscos

A Defesa Civil Municipal considera como o maior risco para a população de Niterói os movimentos de massa. Ressacas e queimadas também são monitoradas pelo órgão, mas

apresentam impacto e incidência significativamente menores. Os movimentos de massa normalmente são deflagrados por chuvas intensas, mas geralmente, estão associados a outras variáveis, tais como:

- Declividade do terreno;
- Má ocupação do solo (moradias de baixo padrão);
- Baixa capacidade de drenagem;
- Vazamentos na rede de abastecimento de água, tanto oficial quanto clandestina, o que encharca o solo.

O mapeamento de riscos normalmente é feito *in loco*, com o deslocamento de um agente da Defesa Civil apoiado por um geólogo (atualmente só existe um profissional com essa formação atuando junto à DCMN). Devido à falta de profissionais suficientes para cobrir toda a extensão do município, o geólogo elaborou um modelo de levantamento preliminar que pode ser feito por um técnico sem formação na área de geologia, para posteriormente ser confirmado pelo referido especialista. A população - geralmente voluntários treinados pela DC - também colabora levantando os riscos nas regiões em que habitam e retornando à DCMN por diversos meios e formatos, como telefone, WhatsApp® ou presencialmente. Entretanto, por não estarem estruturados e não serem feitos por especialistas, esses relatos precisam ser confirmados e trabalhados para serem convertidos no modelo existente. Tanto os levantamentos feitos por especialistas como aqueles que partiram da população são consolidados hoje em um banco de dados na sede da DCMN, o que exige um tempo razoável de digitação dos dados.

Está em curso um trabalho conjunto com um professor e bolsistas da UFF, envolvendo um *software* livre para mapeamento de vulnerabilidades em comunidades carentes.

Monitoramento e Alerta

Existe uma área especializada em monitoramento do tempo dentro da DCMN, composta por meteorologistas e outros especialistas, responsável por acompanhar os boletins de tempo de órgãos como o Cemaden e o INPE, atenta a sinais que possam acarretar situações de emergência no município. Diversas variáveis são acompanhadas e analisadas, sendo uma das principais os níveis de precipitação acumulados.

O governo do estado apresenta um Protocolo (ver Tabela 1) que define um índice de escorregamento, ou seja, estabelece níveis de chuva acumulada, os quais, quando atingidos, aumentam significativamente a chance de escorregamento, situação em que as Defesas Civis Municipais devem ficar em alerta. É importante frisar que, sendo uma tabela para todo o estado,

é baseado em médias e, portanto, não muito preciso; a DCMN iniciou um estudo interno para refinar esse índice para o município.

Tabela 1 - protocolo estadual para acionamento de sirenes

1h s/acumulado	1h c/acumulado	24h	48h	96h	Mensal
>= 45mm	>= 40mm	>= 100mm	>= 125mm	>= 175mm	>= 300mm

No caso de acúmulo de chuva ou chuva muito intensa, planeja-se a evacuação de comunidades vulneráveis; isto é feito com até dois dias de antecedência, dependendo da previsão. Uma vez atingido um limiar de alerta, é disparado um informe para órgãos envolvidos na resposta, como Corpo de Bombeiros, Secretaria de Trânsito, Prefeitura etc., via WhatsApp®, por telefone e/ou por e-mail. Quando o índice crítico é atingido, são disparadas as sirenes nas comunidades; essa confirmação é feita com até uma hora de antecedência. Os agentes da DCMN se deslocam para bases próximas às comunidades vulneráveis, chamadas "pontos de apoio", juntamente com representantes de órgãos de resposta, e ficam de prontidão. Cerca de quinze minutos antes da deflagração do evento, a população começa a ser evacuada. Este último passo é bem delicado, pois os moradores são bem reticentes quanto a abandonar suas casas, e por isso é feito por último, quando existe certeza praticamente absoluta de um evento extremo.

No caso de queimadas (só é de competência da DCMN incêndios em florestas e vegetação urbana), o alerta inicial é recebido pelo Corpo de Bombeiros, que notifica a Defesa Civil, diferentemente do que costuma acontecer em outras ocorrências; para melhor tratar essa questão, a DCMN realizou um convênio com o Corpo de Bombeiros para troca de informações sobre queimadas e incêndios em geral, que envolve a presença de um profissional Bombeiro em tempo integral na sede da DCMN. Analogamente, para ressacas ou outros fenômenos marítimos, o aviso inicial é feito pela Marinha.

Segundo a coordenação, a DCMN faz uso de dois sistemas de comunicação para recebimento de informações e alarmes: um sistema informatizado para envio de SMS; e a ferramenta de mensagens instantâneas WhatsApp®, na qual foram criados vários grupos, alguns envolvendo autoridades e outros líderes comunitários. Esta última ferramenta apresenta como principal vantagem a praticidade, mas além da dificuldade de processamento das mensagens não estruturadas, tem enfrentado recentemente o risco de bloqueios ocasionais por iniciativa da Justiça brasileira, o que compromete sua confiabilidade para situações de emergência em que cada minuto pode fazer diferença.

Preparação para Emergências

A Defesa Civil de Niterói conta com seis planos de contingência criados para orientar os atores envolvidos na atuação contra as principais ameaças naturais ao município, a saber: alagamentos, corridas de massa, deslizamentos, enxurradas, estiagens e incêndios florestais. Cada documento apresenta uma estrutura semelhante, em que são apresentadas, de forma resumida, a legislação considerada, algumas características do relevo de Niterói e uma descrição do fenômeno estudado. O plano também enumera os órgãos envolvidos na resposta (por exemplo, secretarias de saúde, meio ambiente, departamento de trânsito, companhia de limpeza urbana etc.), citando alguns recursos de que dispõem e definindo suas responsabilidades. Embora a Defesa Civil seja o órgão responsável por coordenar as ações de resposta, a ausência de uma hierarquia direta não permite um aprofundamento maior das estratégias e ações, para evitar o risco de ingerência; em decorrência disso, os planos se limitam a definir as ações de alto nível, apresentando em torno de dez páginas cada.

Para mapear trilhas de evacuação em comunidades, a DCMN utiliza um aplicativo gratuito chamado Wikiloc® (http://pt.wikiloc.com/wikiloc/home.do). Este aplicativo utiliza mapas do Google Maps® para, com o auxílio de um celular com GPS, permitir a qualquer pessoa criar suas próprias trilhas, que podem ser compartilhadas com outros.

Reconstrução

Uma maior atuação da Defesa Civil na fase de Reconstrução constitui uma meta para a atual coordenação, atuação esta que ainda é deficiente por conta da limitação de recursos. Seria desejável dispor de uma ferramenta que permitisse e incentivasse as equipes a registrar o que se chamou de "Dados Secundários", ou seja, danos decorrentes de uma ocorrência, mas não imediatamente visíveis, como o impacto negativo no mercado imobiliário, influência no comércio da região, doenças crônicas na população da região etc. A ferramenta poderia ter um tipo de assistente que sugerisse perguntas ao agente que estivesse levantando dados. Seria também importante que esse aplicativo fosse móvel, para permitir o registro *in loco*. Isto passaria uma visão clara para a prefeitura sobre esses danos "invisíveis".

Indicadores

Como resposta ao questionamento sobre quais indicadores poderiam ser levantados e otimizados a fim de melhorar a eficiência da DCMN e a sua maior visibilidade junto à Prefeitura e à comunidade em geral, os seguintes índices foram citados como candidatos:

1. Índice de incêndios em comunidades: medido por unidade de tempo, este índice precisa ser o menor possível. A atuação da DC consiste atualmente na conscientização da população, cabendo tradicionalmente ao Corpo de Bombeiros a resposta aos eventos.
2. Nº de comunidades preparadas: uma comunidade é considerada "preparada" quando conta com um grupo de voluntários capacitados pela DC, treinados para fazer avaliação de riscos e informar ao órgão. Hoje (Junho 2016) o município conta com trinta e duas comunidades preparadas, para um total de mais de cem comunidades. A meta da DCMN para o ano de 2016 é conseguir mapear quarenta comunidades. Seria desejável que estas informações fossem disponibilizadas em um mapa georreferenciado que permitisse a visualização rápida da quantidade de comunidades preparadas.
3. Nº de ocorrências atendidas: um dos maiores desafios para a DCMN envolve a vistoria de locais sujeitos a perigos, que exige a presença de agentes. Como já mencionado anteriormente, é preciso realizar, tanto vistorias preventivas, de iniciativa do próprio órgão, a partir do seu mapeamento de áreas vulneráveis e histórico de ocorrências, quanto vistorias em resposta a demandas da população. Para as vistorias responsivas, o órgão tem uma taxa de atendimento que gira em torno de 75%, mas este número precisa ser melhorado, seja pela agregação de um maior efetivo ou otimização do processo de atendimento.
4. Nº de eventos que registraram vítimas: óbitos são sempre a pior consequência possível de um evento extremo e todos os esforços precisam ser empreendidos no sentido de reduzir seu número ao mínimo. Portanto é preciso atuar na fonte das ameaças, eliminando-as ou mitigando-as quando possível, evitando a exposição de populações vulneráveis, conduzindo orientações e remoções, ou tão somente facilitando a evacuação dessas populações em situações de emergência. Um número baixo para este indicador é a meta mais importante para qualquer Defesa Civil em nível municipal.
5. Percentual de avisos procedentes/alarmes falsos: existe sempre um grau de incerteza quanto à previsão de ocorrência de uma emergência, e isso deixa a DC em uma situação delicada quando precisa alertar a população e outras autoridades competentes da aproximação de uma ameaça, visto que a mobilização de todo um efetivo e a evacuação de habitantes gera um transtorno

só compensado pela confirmação do evento em si. Na hipótese de um "alarme falso", a credibilidade da DCMN é comprometida, portanto é necessário um controle rígido sobre a confiabilidade dos alertas antes destes serem levados adiante.

CAPÍTULO 3 – DISCUSSÃO

POSSIBILIDADES DE OTIMIZAÇÃO POR MEIO DE SISTEMAS INFORMATIZADOS

Após a análise das informações obtidas, foram discutidas com a DCMN algumas possibilidades de soluções tecnológicas com vistas a otimizar processos, chamadas de "módulos" por poderem ser utilizadas de forma isolada ou integradas em um ou mais sistemas. Os módulos propostos foram categorizados por área, conforme descrito a seguir.

Análise de Risco

Para esta área, foram propostos os seguintes módulos:

1. Sistema de Informações Geográficas (SIG) mapeando todas as informações de interesse em diferentes camadas, como: declividade do terreno, pluviosidade, moradias de risco, capacidade de drenagem do solo, nível de encharcamento do solo, vazamentos no abastecimento de água, áreas sujeitas a ressacas, áreas sujeitas a incêndios, áreas de risco etc. O mapeamento dessas informações hoje está dispersa em diversas mídias, parte em mapas de papel, parte em cadernos e documentos eletrônicos, parte na memória dos agentes da DC. É difícil consolidar essas diferentes fontes de informação em tempo útil para uma análise ampla.
2. Banco de Dados (BD) de consolidação da avaliação de riscos dos voluntários. Os voluntários treinados procuram identificar os riscos presentes em suas comunidades, e repassam essa informação à Defesa Civil por diversas vias, como WhatsApp®, SMS, telefone ou mesmo contato pessoal. Seria interessante ter uma via oficial, estruturada, de coleta e consolidação dessas informações em um banco de dados que pudesse gerar informação útil para a Defesa Civil. Este módulo também poderia potencialmente se integrar com o software livre sendo desenvolvido pela UFF para mapeamento de riscos em comunidades.

Monitoramento

Para esta área, foi proposto o seguinte módulo:

1. Alarme de ocorrências, disparado por atingimento de índices críticos de pluviosidade (para alagamentos, enxurradas e deslizamentos). Para ressacas, seria necessário monitorar o nível do mar, e para incêndios, criar uma interface

de comunicação automática com o Corpo de Bombeiros. O módulo poderia realizar notificação de ocorrências por e-mail, SMS (poderia se integrar com o sistema em uso de envio de SMS) ou aplicativos de mensagens instantâneas como o WhatsApp®. Esta ferramenta poderia também fazer um registro dos alarmes emitidos e facultar a categorização desse histórico em alarmes falsos e procedentes, o que permitiria a elaboração de relatórios. O acompanhamento desses dados hoje é feito pelo acesso manual ao *site* do Cemaden, o qual não apresenta todas as informações necessárias para uma análise completa, além de apresentar falhas ocasionais na consistência das informações fornecidas.

Preparação para Emergências

Para esta área, foram propostos os seguintes módulos:

1. Módulo para Elaboração de Planos de Emergência: poderia ser desenvolvido, ou adaptado a partir de sistemas existentes da indústria, um módulo específico para elaboração de planos de emergência, com suporte a versionamento dos documentos e disponibilização em nuvem. Este módulo incluiria desde a concepção do plano, iniciando pela identificação de cenários de emergência, passando pela elaboração de estratégias para combate que se desdobrariam em ações, até a confecção do documento em si, que poderia ser estruturado de forma a facilitar a simulação e validação das estratégias exploradas. Este módulo poderia se integrar com o Wikiloc® para o mapeamento das trilhas de evacuação. Os planos existentes hoje foram elaborados com um nível muito alto de abstração e precisam ser melhor detalhados e desdobrados em diferentes cenários emergenciais para apresentar uma melhor cobertura na resposta a situações de crise. Além disso, foram elaborados como documentos Word não versionados, o que dificulta sua recuperação e gestão de alterações.
2. Sistema de Informações Geográficas, conforme já proposto na seção anterior, adicionando uma camada para mapeamento das comunidades preparadas, apresentando a quantidade absoluta dessas comunidades e seu percentual em relação ao total. Hoje essa informação não está disponível de forma estruturada nem em formato eletrônico.

Reconstrução

Para esta área, foi proposto o seguinte módulo:

1. SIG/BD que permitisse o registro de óbitos e danos decorrentes de desastres. A entrada de dados seria feita em dispositivos móveis como celulares ou *tablets* e contaria com um assistente, que orientasse o usuário sobre as informações que precisariam ser coletadas. Este módulo poderia disponibilizar relatórios georreferenciados para apoio à decisão. Em decorrência dos recursos humanos da Defesa Civil de Niterói serem insuficientes para atender à demanda normal de chamados da população, o registro de dados obtidos em consequência de um desastre acaba sendo limitado e muita informação útil, que poderia alimentar um planejamento em prevenção e mitigação de tragédias, acaba se perdendo.
2. SIG/BD consolidando índices de ocorrências, como índices de incêndios em comunidades, com visão geográfica, tabular ou gráfica. Essa informação não é consolidada atualmente e poderia ajudar na compreensão dos mecanismos sociais que dão origem aos focos de incêndio e permitiria uma atuação mais precisa nos locais mais propensos à ocorrência de eventos críticos.

Comunicação

Para esta área, foi proposto o seguinte módulo:

1. Portal *web* para comunicação interna e externa, integrado com o *site* oficial da DCMN, contendo informações como material de treinamento para voluntários, números oficiais, sala de imprensa, informações à população, notícias, planos de emergência públicos, trilhas de evacuação, histórico de ocorrências, registro de atendimentos, etc. O *site* existente hoje é principalmente voltado para a divulgação de informações para a comunidade externa. Poderia haver uma versão de acesso interno para a gestão de informações dentro da própria Defesa Civil Municipal, e um maior investimento na interação com a comunidade para a versão externa.

De todas as possibilidades oferecidas, buscou-se implementar as duas soluções consideradas mais prioritárias, que seriam o Sistema de Informações Geográficas e o Sistema de Acompanhamento de Níveis de Pluviosidade. As seções seguintes descrevem ambas as soluções.

SOLUÇÃO 1: SISTEMA DE INFORMAÇÕES GEOGRÁFICAS

O SIG foi selecionado pela Coordenação como prioritário para apoio à Análise de Risco, cujas camadas deveriam ser:

- áreas vulneráveis e áreas com comunidades preparadas;
- associações de moradores e NUDECs (Núcleos Comunitários de Defesa Civil);
- delimitação de cada comunidade;
- pluviômetros do Cemaden e outros sensores;
- unidades de saúde e SAMU (Serviço Ambulatorial Móvel Urbano);
- unidades do Corpo de Bombeiros, da PM (Polícia Militar) e da GM (Guarda Municipal);
- escolas e creches;
- unidades prisionais;
- órgãos da Prefeitura;
- bases da CLIN (Companhia de Limpeza Urbana de Niterói).

De acordo com a Coordenação da DCMN, seria interessante o acesso móvel a esse sistema, para a utilização das equipes em deslocamento. No momento em que o levantamento dos dados para a elaboração deste trabalho estava sendo realizado, já estava em andamento o mapeamento da maior parte das camadas descritas acima por uma agente do órgão no SIG gratuito Google Earth®, que só era consultado na máquina em que havia sido instalado.

Em consulta à documentação oficial do Google Earth® (https://www.google.com.br/intl/pt-BR/earth/), verificou-se que existia a possibilidade de acesso remoto a mapas criados pela ferramenta, usando um *plug-in* (complemento instalado em navegadores *web*, permitindo estender suas funcionalidades). Entretanto, além do inconveniente de depender da instalação desse *plug-in*, e da verificação de compatibilidade entre versões de navegadores, a consulta aos endereços https://support.google.com/earth/answer/178389?hl=pt-BR&ref_topic=2376748 e https://developers.google.com/earth/ mostrou que o *plug-in* era parte de uma API (*Application Programming Interface*, isto é, uma interface criada para desenvolvedores de *software*) que estava sendo descontinuada desde 2014, por razões de segurança, e seria suspensa definitivamente em 2016. Portanto, essa não era uma opção viável e outras alternativas precisavam ser buscadas.

Como desenvolver ou mesmo adaptar um SIG existente exigiria um investimento financeiro elevado, buscou-se outras soluções gratuitas disponíveis, e a opção mais natural foi o Google My Maps® (https://www.google.com/mymaps), a versão editável e um pouco menos conhecida do Google Maps®. A ferramenta não oferece os recursos de mapa 3D e navegação rica disponíveis no Google Earth® (que não eram importantes para a utilização desejada), assim como não oferece uma ampla gama de recursos normalmente encontrados em ferramentas de SIG, mas atende às necessidades da DCMN e é simples o suficiente para ser utilizado por pessoas sem conhecimentos técnicos em geoprocessamento. Se no futuro forem necessários recursos mais avançados, podem ser utilizadas outras ferramentas de uso gratuito, como o QGIS (http://www.qgis.org), gvSIG (http://www.gvsig.com/) ou Kosmo (www.opensig.es). Os mapas gerados no Google My Maps podem ser exportados para formatos padronizados da indústria de SIG, reconhecidos por outros *softwares* dessa área.

Após a realização de alguns testes, a proposta de uso do *software* foi apresentada à DCMN, tendo sido bem recebida (ver Figura 1). A profissional responsável pelo mapeamento achou a interface mais simples e intuitiva, embora bastante poderosa, com destaque à funcionalidade de edição simultânea por usuários diferentes no mesmo mapa, o que lhe permitiu dividir o trabalho de mapeamento com outros profissionais e com isso ganhar tempo. Para a Coordenação do órgão, a possibilidade de ver o resultado do mapeamento diretamente em um aparelho celular, com a possibilidade de ligar e desligar camadas e visualizar informações relacionadas a cada local, foi considerada bastante interessante. Foi descoberta a limitação de dez camadas por mapa, a qual foi contornada criando-se outros mapas detalhados e consolidando informações no mapa original; a possibilidade de exportação de dados de um mapa para outro também ajudou a minimizar o problema.

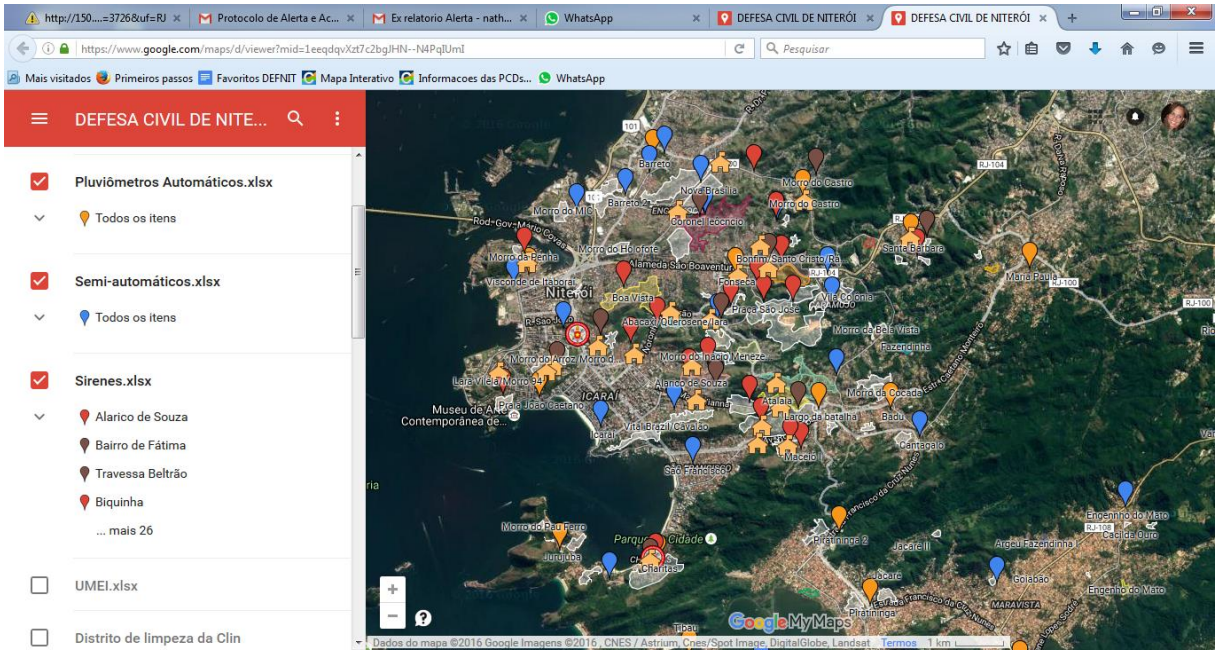


Figura 1. mapa de análise de risco da cidade de Niterói

SOLUÇÃO 2: SISTEMA DE ACOMPANHAMENTO E ALARME (PLUVI)

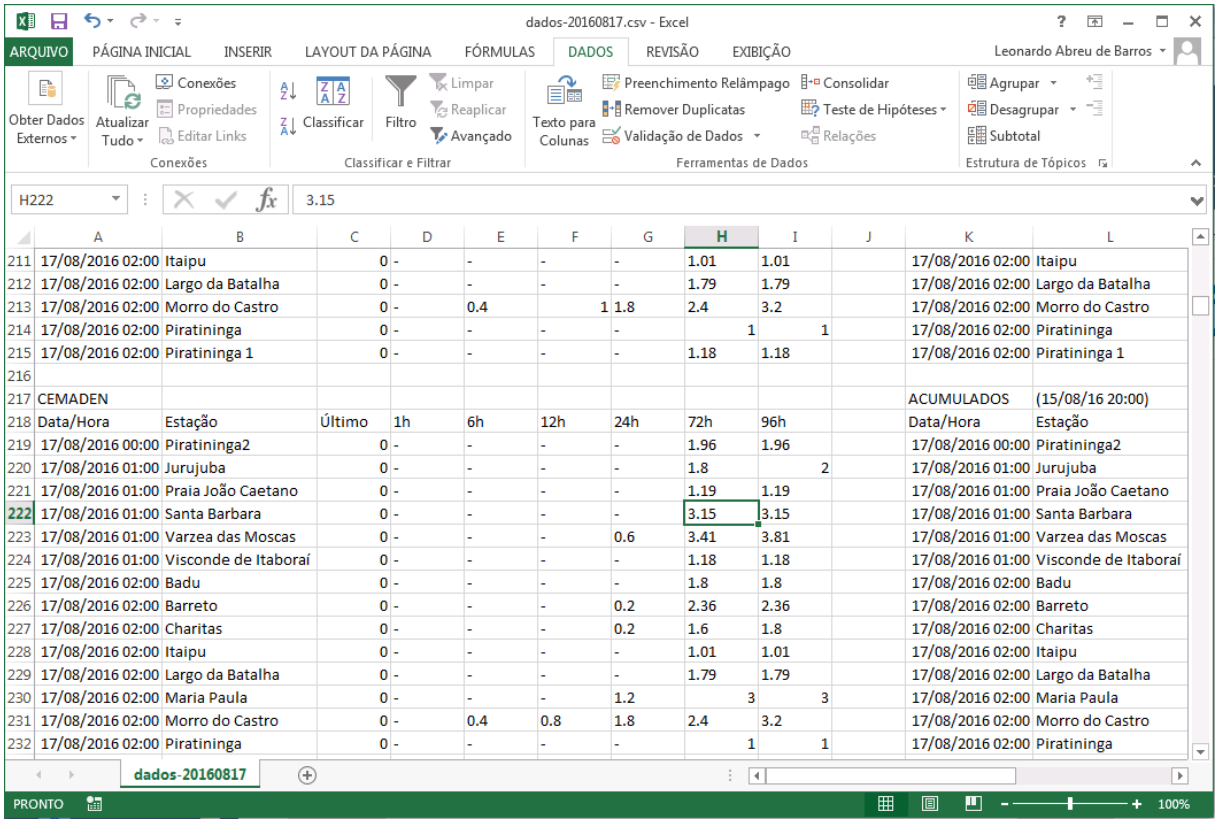
Conforme citado anteriormente, o módulo escolhido como segunda prioridade para otimização foi o de Alarmes de ocorrências, da área de Monitoramento (referenciado doravante neste documento pelo nome "PLUVI"). A meteorologista responsável explicou e demonstrou como eram hoje acompanhados os níveis de chuva no município, bem como são disparados os estados de alerta e alarmes. Existem quatro níveis no acompanhamento: o estado de “vigilância” é o padrão, alterado para “alerta” quando determinados limiares são atingidos; outros limiares determinam a passagem do estado de “alerta” para o estado de “alarme”, e por fim a iminência de deslizamentos determina a passagem do estado de “alarme” para “alarme máximo”.

Para acompanhar dados de chuva, segundo a meteorologista, só existem hoje duas fontes disponíveis: os dados de pluviômetros disponibilizados pelo Cemaden e uma estação meteorológica completa, pertencente ao Governo do Estado. Os dados do Cemaden são recuperados manualmente via *site* www.cemaden.gov.br, e atualizados a cada dez minutos. Os dados da estação são disponibilizados via FTP em um servidor para o qual a DCMN possui credenciais de acesso.

Pela facilidade de acesso e maior disponibilidade de dados, optou-se por priorizar a automatização da leitura dos dados no *site* do Cemaden. Para isso, foi desenvolvido um *software* utilizando tecnologias gratuitas (ver seção TECNOLOGIAS UTILIZADAS). A estrutura do *software* é bem simples e linear: a cada dez minutos (intervalo configurável em um arquivo de

propriedades, assim como diversas outras características), o sistema consulta os dados de pluviômetros de Niterói no *site* do Cemaden, e quando encontra dados novos, soma-os com os lidos anteriormente, salva os dados lidos e acumulados em planilha (ver Figura 2), e processa os dados, comparando-os com uma tabela de níveis críticos de chuva fornecida pelo Estado do RJ (ver Tabela 1); se encontrar algum nível acima do limite, dispara *e-mails* de alerta aos responsáveis da Defesa Civil. A geração da planilha (na verdade um arquivo .CSV, facilmente lido por planilhas eletrônicas) é interessante para consulta dos técnicos. O sistema foi preparado para ser tolerante a pequenas falhas no envio de dados do *site* do Cemaden e consegue inferir os dados corretos com base em uma análise algorítmica quando estes apresentam inconsistências.

O sistema possui pouco mais de 1000 linhas de código (LOC). Ele é instalado como um serviço Windows e roda 24/7 em um servidor dedicado, recuperando-se automaticamente em caso de queda de energia. Alterações nas propriedades são recarregadas sem necessidade de reinício do sistema e eventuais erros são enviados por *e-mail* aos administradores cadastrados. O sistema, tal como está, poderia ser reutilizado em qualquer município do país coberto pela rede de pluviômetros do Cemaden, bastando-se apenas configurar nas propriedades o endereço de acesso aos dados desejados.



	A	B	C	D	E	F	G	H	I	J	K	L
211	17/08/2016 02:00	Itaipu	0	-	-	-	-	1.01	1.01		17/08/2016 02:00	Itaipu
212	17/08/2016 02:00	Largo da Batalha	0	-	-	-	-	1.79	1.79		17/08/2016 02:00	Largo da Batalha
213	17/08/2016 02:00	Morro do Castro	0	-	0.4	1	1.8	2.4	3.2		17/08/2016 02:00	Morro do Castro
214	17/08/2016 02:00	Piratininga	0	-	-	-	-	1	1		17/08/2016 02:00	Piratininga
215	17/08/2016 02:00	Piratininga 1	0	-	-	-	-	1.18	1.18		17/08/2016 02:00	Piratininga 1
216												
217	CEMADEN										ACUMULADOS	(15/08/16 20:00)
218	Data/Hora	Estação	Último	1h	6h	12h	24h	72h	96h		Data/Hora	Estação
219	17/08/2016 00:00	Piratininga2	0	-	-	-	-	1.96	1.96		17/08/2016 00:00	Piratininga2
220	17/08/2016 01:00	Jurujuba	0	-	-	-	-	1.8	2		17/08/2016 01:00	Jurujuba
221	17/08/2016 01:00	Praia João Caetano	0	-	-	-	-	1.19	1.19		17/08/2016 01:00	Praia João Caetano
222	17/08/2016 01:00	Santa Barbara	0	-	-	-	-	3.15	3.15		17/08/2016 01:00	Santa Barbara
223	17/08/2016 01:00	Varzea das Moscas	0	-	-	-	0.6	3.41	3.81		17/08/2016 01:00	Varzea das Moscas
224	17/08/2016 01:00	Visconde de Itaboraí	0	-	-	-	-	1.18	1.18		17/08/2016 01:00	Visconde de Itaboraí
225	17/08/2016 02:00	Badu	0	-	-	-	-	1.8	1.8		17/08/2016 02:00	Badu
226	17/08/2016 02:00	Barreto	0	-	-	-	0.2	2.36	2.36		17/08/2016 02:00	Barreto
227	17/08/2016 02:00	Charitas	0	-	-	-	0.2	1.6	1.8		17/08/2016 02:00	Charitas
228	17/08/2016 02:00	Itaipu	0	-	-	-	-	1.01	1.01		17/08/2016 02:00	Itaipu
229	17/08/2016 02:00	Largo da Batalha	0	-	-	-	-	1.79	1.79		17/08/2016 02:00	Largo da Batalha
230	17/08/2016 02:00	Maria Paula	0	-	-	-	1.2	3	3		17/08/2016 02:00	Maria Paula
231	17/08/2016 02:00	Morro do Castro	0	-	0.4	0.8	1.8	2.4	3.2		17/08/2016 02:00	Morro do Castro
232	17/08/2016 02:00	Piratininga	0	-	-	-	-	1	1		17/08/2016 02:00	Piratininga

Figura 2. planilha gerada pelo PLUVI, a partir de dados do *site* do Cemaden

Arquitetura e Dinâmica de Execução

O sistema PLUVI foi desenvolvido na linguagem Java, uma linguagem de programação orientada a objetos (http://www.hardware.com.br/artigos/programacao-orientada-objetos/). Neste paradigma de programação, os principais conceitos e entidades do sistema são chamadas de "classes", as quais podem estar contidas em um ou mais arquivos do sistema operacional. O sistema de acompanhamento de pluviosidade PLUVI foi modelado com apenas quatro classes (CEMADEN, STATIONDATA, READDATATASK e PINGTASK), cuja principal é a CEMADEN. Essa classe é responsável por ler os dados das estações pluviométricas, acumulá-los, verificar condições de alerta e, se for o caso, disparar alarmes por *e-mail*. Uma única instância da classe CEMADEN é utilizada em toda a vida do sistema.

A intervalos regulares (o intervalo, assim como outras características personalizáveis do sistema, é definido pelos administradores em um arquivo de propriedades), a classe CEMADEN é acionada por uma classe auxiliar READDATATASK, cujo único propósito é agendar o processamento. A cada rodada, são criadas ou reutilizadas instâncias da classe STATIONDATA, classe que representa os dados de uma estação, para armazenar e acumular dados lidos do *site* do Cemaden (os dados de uma mesma estação pluviométrica sempre serão armazenados em um mesmo objeto do tipo STATIONDATA). Por fim, uma PINGTASK, similar à READDATATASK, é configurada no mesmo arquivo de propriedades para regularmente enviar *e-mails* aos administradores, somente para confirmar que o sistema continua executando normalmente. Somente uma instância de READDATATASK e PINGTASK são criadas ao longo de toda a vida do sistema.

O acesso ao arquivo de propriedades é feito via CEMADEN, logo os intervalos de acionamento das classes READDATATASK e PINGTASK são lidos e configurados em métodos da CEMADEN. Todo o passo-a-passo da execução (um relatório gerado pelo sistema detalhando tudo o que ocorreu ao longo da execução, conhecido no mundo computacional pelo termo "*log*") também é centralizado na CEMADEN.

Na inicialização, ambas as tarefas READDATATASK e PINGTASK são instanciadas e configuradas; a partir deste momento, elas serão acionadas em *threads* separadas, a intervalos regulares (as *threads* são linhas de execução independentes, que permitem a um único sistema realizar duas ou mais tarefas paralelamente). A classe PINGTASK solicita ao objeto CEMADEN o envio de *e-mails* aos administradores a intervalos regulares (o padrão é 24h),

somente para confirmar que o sistema está ativo. Já a READDATATASK solicita ao objeto CEMADEN as seguintes atividades, nesta ordem (a Figura 3 ilustra o processo):

1. Leitura dos dados do Cemaden;
2. Acumulação dos dados;
3. Verificação de condições para alarme;
4. Envio de *e-mails*, em caso de encontradas condições para alarme.

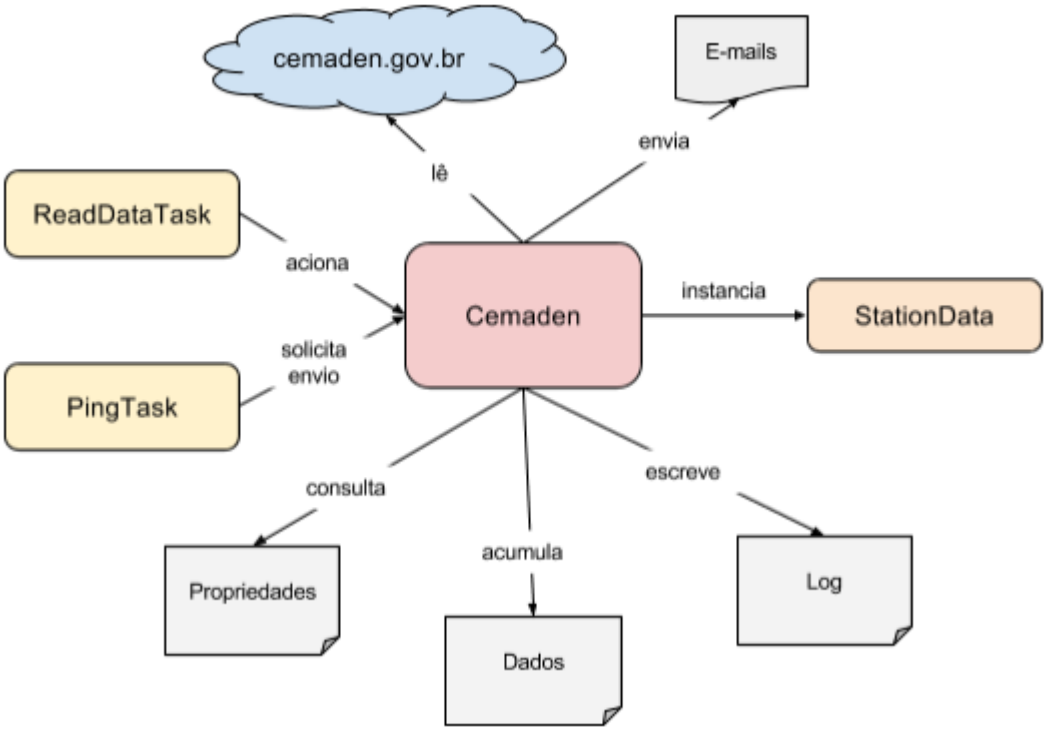


Figura 3: estrutura esquemática do sistema

Para ser instalado, o sistema é compactado em um arquivo do tipo *jar* (similar Java ao popular arquivo *zip*) e instalado como um serviço Windows (serviços são programas que se iniciam sozinhos e permanecem ativos enquanto a máquina estiver ligada), logo não possui uma camada de interface com o usuário. A configuração, ou seja, todas as características personalizáveis que os administradores precisam informar ao sistema é feita por arquivo de propriedades, o qual é recarregado dinamicamente a cada acionamento do processo de leitura e acumulação de dados (não é necessário reiniciar o sistema após alterações no arquivo de propriedades). A saída do sistema, isto é, o que ele produz, são principalmente os dados lidos e acumulados, armazenados em arquivos de extensão .CSV, que são facilmente lidos por

planilhas eletrônicas como o Microsoft Excel; além disso, são gerados *e-mails* de alertas ou erros, bem como arquivos de *log*.

Requisitos

Tradicionalmente, os requisitos de um *software* são classificados em Funcionais e Não-Funcionais (SOMMERVILLE, 2001). **Requisitos Funcionais** são as declarações das funções que o sistema deve oferecer, como o sistema se comporta com entradas particulares e como o sistema deve se comportar em situações específicas. O termo função é usado no sentido genérico da operação que pode ser realizada pelo sistema, seja por meio de comandos dos usuários, ou seja pela ocorrência de eventos internos ou externos ao sistema. Em alguns casos, os requisitos funcionais podem também explicitamente definir o que o sistema não deve fazer. Já **Requisitos Não-Funcionais** são as restrições nas funções oferecidas pelo sistema. Incluem restrições de tempo, restrições no processo de desenvolvimento e padrões de qualidade global de um *software*, como manutenibilidade, usabilidade, desempenho, custo e outros. São listados a seguir os requisitos funcionais e não-funcionais considerados no desenvolvimento do sistema PLUVI:

Requisitos Funcionais

1. Armazenar dados lidos no *site* do Cemaden em formato tabular, legível por planilhas eletrônicas.
2. Acumular os dados lidos no *site*, de forma a prover acumulados não disponíveis no *site* a todo momento.
3. Acompanhar os níveis de pluviosidade registrados, e em caso de limites atingidos, disparar alarmes em forma de *e-mails*.

Requisitos Não Funcionais

1. Enviar *e-mails* periódicos aos administradores para garantir que o sistema continua ativo.
2. Tolerância a falhas nos dados lidos do Cemaden.
3. Enviar *e-mails* aos administradores do sistema na ocorrência de falhas.
4. Recarregamento dinâmico das propriedades a cada intervalo de leitura de dados.
5. Reinício automático do sistema em caso de queda de energia.

Tecnologias Utilizadas

O sistema PLUVI foi desenvolvido utilizando a linguagem *Java Standard Edition*, versão 1.7.0_60-b19 (Java SE Development Kit 7 - Downloads | Oracle Technology Network ...). O código foi elaborado na IDE (ambiente de desenvolvimento integrado) *Eclipse* versão *Mars* (Mars Eclipse) e versionado com *GIT* versão 1.9.5 (Git - Downloads). A leitura automatizada de dados é feita com a ferramenta *Selenium WebDriver* versão 2.53.1 (http://www.seleniumhq.org), a qual utiliza a API (Interface de Programação de Aplicação) nativa do navegador Mozilla Firefox 44.0.1-32 bits. O envio de mensagens é feito com o a biblioteca *JavaMail* versão 1.5.6 ((https://java.net/projects/javamail/pages/Home) e utiliza como servidor SMTP uma conta no GMail. O sistema é instalado como serviço *Microsoft Windows 7* por meio do *Java Service Wrapper Community Edition* versão 3.5.30-32 bits (http://wrapper.tanukisoftware.com/doc/english/download.jsp). Os testes automatizados foram implementados com *JUnit* versão 4 (http://junit.org/junit4/) e a cobertura dos testes foi verificada com *EclEmma* versão 2.3.3.201602231923 (EclEmma - Java Code Coverage for Eclipse).

Requisitos de Hardware e Software

Os requisitos de instalação são bem modestos: o sistema PLUVI pode ser instalado em qualquer sistema operacional com uma máquina virtual Java (a instalação da Defesa Civil Niterói foi feita em Microsoft Windows). É obrigatória a instalação do navegador Mozilla Firefox para acesso aos dados do Cemaden (a versão atual do sistema utiliza a versão 44.0.1-32 bits do Firefox, para compatibilidade com a versão 2.53.1 do Selenium). O sistema consome pouca memória RAM e ocupa um máximo de 100Mb de espaço em disco para *logs*. Em experiência na Defesa Civil Niterói, os logs ocuparam uma média de 700Kb por mês. Os arquivos de dados (.CSV), que nunca são removidos, variam entre 20Kb a 200Kb por dia, dependendo se houver ou não registro de chuva.

Instalação

O sistema é distribuído como um único arquivo *zip*, de aproximadamente 12Mb. O processo de instalação consiste em três passos:

1. Expandir o *zip* em algum diretório do Windows.
2. Configurar os arquivos de propriedades *wrapper.conf* e *config.properties*, no diretório *conf*:

a. wrapper.conf: a única propriedade exigida é o diretório onde o Java se encontra instalado (*wrapper.java.command*). Todas as demais podem ser deixadas com valores padrão.

b. config.properties: é necessário configurar a propriedade *AMBIENTE*, para diferenciar os ambientes de desenvolvimento/homologação/produção, *URL* (dependendo da cidade onde o sistema for instalado), bem como *EMAIL.AVISOS* e *EMAIL.ERROS*, para identificar respectivamente aqueles que devem receber avisos de chuva (agentes da DC) e avisos de erros (administradores).

3. Executar o arquivo *Installcemaden-NT.bat*, dentro do diretório *bin*, para instalar o serviço. É recomendável rodar anteriormente o arquivo *cemaden.bat*, por linha de comando, para verificar se o sistema está funcionando antes de instalar o serviço propriamente dito.

Estrutura de Diretórios

A árvore de diretórios do sistema é composta de cinco pastas (ver Figura 4):

1. *bin*: arquivos binários para rodar o sistema manualmente (*cemaden.bat*), instalá-lo (*Installcemaden-NT.bat*) ou desinstalá-lo (*Uninstallcemaden-NT.bat*) como serviço. O arquivo *wrapper.exe* é de uso do Java Service Wrapper e não deve ser executado diretamente.

2. *conf*: arquivos de configuração do sistema (*config.properties*) e do Java Service Wrapper (*wrapper.conf*).

3. dados: diretório de geração de arquivos de leitura dos dados do Cemaden, no formato *dados<anomêsdia>.csv*.

4. lib: diretório onde ficam armazenados o *jar* contendo o código do sistema (*cemaden.jar*) bem como as demais bibliotecas utilizadas.

5. logs: arquivos de *log* gerados pelo sistema (*cemaden.log*) e pelo Java Service Wrapper (*wrapper.log*).

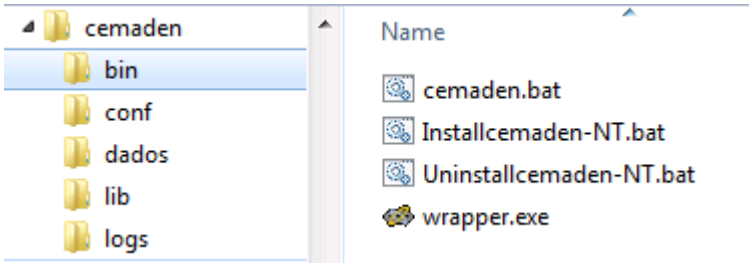


Figura 4: estrutura de diretórios do sistema

Arquivo de Propriedades

O sistema possui um arquivo de propriedades (*config.properties*, no diretório *conf*) que armazena todas as características do sistema que podem ser personalizadas em cada instalação. Como o sistema não possui uma interface gráfica, este é o mecanismo que os administradores possuem para configurar o sistema. Se desejável, é possível criar uma interface gráfica somente para atualização desse arquivo, para tornar sua manipulação mais intuitiva.

As propriedades disponíveis estão descritas a seguir:

- AMBIENTE: Ambiente onde o sistema está sendo executado. Por exemplo, "Produção DC Niterói" ou "Teste DC Niterói". Este valor é usado nos e-mails enviados diariamente pela PINGTASK para informar quais ambientes (instalações do sistema PLUVI) estão ativos.
- LOG: Nível de *log*. Quando assume o valor NORMAL, poucas mensagens serão exibidas, somente registrando os principais marcos do sistema. Já com o valor DEBUG, serão exibidas informações mais detalhadas, úteis para apoiar os desenvolvedores no diagnóstico de problemas.
- URL: endereço de acesso aos dados do Cemaden. Essa *url* deve ser recuperada a partir do endereço www.cemaden.gov.br/mapainterativo, clicando no município desejado.
- ID.TABELA: este é o identificador da tabela de dados no *site* do Cemaden. É um identificador utilizado pelo sistema para encontrar os dados, presente no código-fonte da página, o qual não deveria sofrer alteração, salvo se houver modificações no *site*.
- INTERVALO: Intervalo de leitura dos dados no site do Cemaden (em minutos). Normalmente assume o valor 10.

- EMAIL.AVISOS: E-mails para receber avisos de limites de chuva ultrapassados, direcionados aos agentes da Defesa Civil (separados por vírgulas). Exemplo: fulano@uol.com.br, ciclano@hotmail.com.
- EMAIL.ERROS: E-mails para receber notificações de erros, direcionados aos administradores do sistema (separados por vírgulas). Exemplo: fulano@uol.com.br, ciclano@hotmail.com.

As propriedades a seguir são específicas para registrar os limites de chuva (em mm) para disparar alarmes:

- 1H.SEM.ACUMULADO = 45
- 1H.COM.ACUMULADO = 40
- 24H = 100
- 48H = 125
- 96H = 175
- MENSAL = 300

Leitura de Dados

A leitura de dados é feita por acesso automatizado ao *site* do Cemaden, mais especificamente aos dados dos pluviômetros automáticos disponíveis em http://www.cemaden.gov.br/mapainterativo/. O procedimento de leitura fica encapsulado em um único método e portanto poderia ser facilmente substituído caso a fonte de dados fosse provida de outra forma. O processo de leitura é dependente da estrutura do HTML (linguagem utilizada no *site* do Cemaden), logo se houver alguma mudança na página pode ser necessário rever o código (ver seção ANÁLISE CRÍTICA).

A leitura é feita de dez em dez minutos (intervalo configurável), que corresponde ao intervalo mínimo de disponibilização de novos dados no *site* do Cemaden. Entretanto, o sistema verifica se algum novo dado foi disponibilizado ou se todas as leituras estão idênticas às últimas realizadas, isto porque em condições normais (sem chuva), o Cemaden disponibiliza dados somente de hora em hora. A partir do momento em que uma estação começa a apresentar níveis de chuva, esse intervalo diminui até o mínimo de dez minutos. Para evitar a geração de dados necessários, o sistema só armazena leituras que apresentem alguma diferença em relação às anteriores.

Salvamento de Dados

Os dados lidos do Cemaden (quando diferentes da leitura anterior), bem como os acumulados pelo sistema, são registrados em arquivos .CSV, facilmente lidos em qualquer planilha eletrônica como o MS Excel, separados por dia (ver Figura 2). São registrados a data/hora de leitura (no fuso correto - ver seção TOLERÂNCIA A FALHAS), nome da estação, os níveis de pluviosidade lidos pela última vez, assim como os acumulados nas últimas horas (e no caso do sistema, é fornecido o acumulado mensal). Esse acumulado mensal é fornecido no *site* do Cemaden após a conclusão do mês, mas ele não está disponível antes disso. Com a acumulação realizada no sistema, é possível ter esse dado atualizado a cada dez minutos.

Processamento de Alarmes

O sistema verifica, a cada nova leitura, se os níveis registrados apresentam condições para alerta. Os níveis desse protocolo do Governo do Estado do RJ (Tabela 1) foram inseridos no sistema na forma de propriedades (ver seção ARQUIVO DE PROPRIEDADES) e são comparados com cada dado lido do Cemaden e/ou acumulado pelo sistema. Quando é encontrada uma correspondência, o sistema armazena em uma estrutura de memória um novo alarme - que consiste basicamente em um texto informando o nível de pluviosidade atingido - para ser enviado posteriormente por *e-mail* (ver seção ENVIO DE MENSAGENS).

Envio de Mensagens

O sistema faz uso extensivo do envio de mensagens por *e-mail*, fundamentalmente para cumprir dois objetivos:

1. Avisar os agentes da Defesa Civil no caso de determinado limite de pluviosidade ter sido atingido.
2. Notificar os administradores do sistema da ocorrência de falhas no sistema.

Para ambos os casos, os endereços de agentes e administradores são configurados em propriedades (ver seção ARQUIVOS DE PROPRIEDADES). Foi criada uma conta no GMail (niteroidc@gmail.com) apenas para que o sistema pudesse utilizá-la como servidor SMTP para envio de mensagens.

Tolerância a Falhas nos Dados de Entrada

Foram implementadas algumas adaptações no sistema para que este pudesse contornar algumas falhas nos dados recebidos do Cemaden. As adaptações realizadas foram as seguintes:

1. **Fuso horário**: os dados do Cemaden são fornecidos com base no fuso do meridiano de Greenwich, isto é, GMT. O fuso foi corrigido para a referência do Rio de Janeiro, GMT-3.

2. **Dados incompletos/ estações desativadas:** as leituras de algumas estações podem não apresentar o nome da estação ou a hora da leitura. Em alguns casos, existem dados sendo apresentados de leituras muito antigas, de meses atrás, o que é percebido como uma estação desativada. O sistema verifica todas essas condições e ignora dados incompletos, assim como ignora dados com mais de 24h de idade.

3. **Insuficiência de dados:** o protocolo fornecido pelo Governo do Estado do Rio de Janeiro prevê uma série de condições em que alertas devem ser disparados (por exemplo, após 1h chovendo com acumulado de 40mm ou após 24h chovendo com acumulado de 100mm), sendo que o *site* do Cemaden não fornece acumulados suficientes para verificar todos os níveis de alerta (especificamente, não é fornecido o acumulado de 48h nem o mensal). Para isso, o sistema realiza sua própria acumulação, que é sempre comparada e ajustada com os acumulados do Cemaden (ver item seguinte).

4. **Ajustes na acumulação de dados:** com a observação regular dos dados disponibilizados pelo *site* é possível perceber que eventualmente existem falhas de continuidade entre os dados lidos e/ou acumulados. Por exemplo, pode ocorrer de não ter havido nenhuma leitura nas últimas duas horas no campo "Último" (reflete os dados mais recentes no pluviômetro) e o campo acumulado de 1h apresentar valores positivos. Para corrigir essa distorção, foi implementado um algoritmo que considera sempre o maior acumulado como correto, ajusta o valor acumulado anterior e corrige progressivamente os valores acumulados seguintes.

Análise Crítica

O desenho do sistema prioriza a economia de memória. Como explicado na seção ARQUITETURA E DINÂMICA DE EXECUÇÃO, somente uma instância das classes CEMADEN, READDATATASK e PINGTASK são utilizadas durante todo o ciclo de vida do sistema. Existirá uma instância de STATIONDATA para cada estação, mas elas serão reaproveitadas (os dados são acumulados) a cada nova leitura, portanto, o uso de memória é bastante limitado.

A centralização das atividades de leitura do *site* do Cemaden, salvamento e acumulação dos dados, verificação de alertas, envio de *e-mails* e escrita de *log* em na única instância CEMADEN e em uma única *thread* poderia representar um "gargalo" no desempenho

se o volume de leituras fosse muito grande e o intervalo entre elas muito reduzido. Porém, com um número de estações na ordem de dois dígitos e um intervalo de leitura de dez minutos (intervalo mínimo de publicação de novos dados no *site* do Cemaden), o desempenho do sistema é bastante satisfatório.

Os arquivos de *log*, que poderiam crescer a ponto de representar um problema de disco, apresentam três tipos de controle: primeiro, é possível definir o nível de *log* desejado, por um parâmetro no arquivo de propriedades. O valor padrão ("normal") registra muito pouca informação, já o valor de diagnóstico ("debug") apresenta informações detalhadas para investigar um problema específico e por isso deve ser utilizado somente durante o processo de investigação de falhas. Em segundo lugar, o tamanho dos arquivos é limitado a 1Mb, e seu número máximo é limitado a 100 arquivos, em política de rotação: a partir do momento em que o centésimo arquivo atinge o limite de 1Mb, o primeiro e mais antigo arquivo de *log* passa a ser sobrescrito e assim por diante, garantindo que o sistema nunca ocupará mais de 100Mb de espaço em disco. A única questão que pode a longo prazo representar um problema de armazenamento são os arquivos de dados (.CSV), que nunca são removidos automaticamente, visto que representam o produto final do sistema. Em um período de experiência de alguns meses em atividade registrando os dados de Niterói, estes arquivos variaram entre 20Kb a 200Kb por dia (dependendo se houver ou não registro de chuva), o que só seria um problema após anos, considerando a média de armazenamento dos discos atuais. Além disso, recomenda-se a utilização de unidades de armazenamento removíveis para guardar o histórico de leituras.

A leitura de dados é um tanto frágil, pois depende de dois fatores: 1) versão específica do navegador Mozilla Firefox, compatível com a versão do Selenium; e 2) formatação dos dados do *site* do Cemaden. O primeiro item representa um risco porque não se pode instalar atualizações no navegador, mesmo de segurança, sem prévia verificação de compatibilidade com a versão da biblioteca Selenium. Isso deixa o navegador vulnerável a ataques que explorem as brechas da versão, e portanto a solução de contorno para minimizar o risco é deixar essa versão do navegador dedicada somente ao sistema. A livre navegação na internet deve ser realizada em outros navegadores, ou preferencialmente, em outra máquina. O segundo ponto indica que será necessária intervenção de programadores caso o *site* do Cemaden seja modificado, visto que não há compromisso algum da fonte de dados em manter o formato existente hoje. A solução para isso seria estabelecer um acordo com o Cemaden para provimento de uma API dedicada, ou ao menos um protocolo para aviso prévio caso a página esteja em vias de sofrer modificações.

Tratamento de Exceções

"Exceções" são eventos que podem ocorrer durante a execução de um programa e interromper o fluxo normal de instruções (https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html), ou seja, são possibilidades de falhas no sistema que precisam ser tratadas.

No sistema PLUVI, todas as exceções são registradas em *log* e enviadas por *e-mail* aos administradores do sistema (cujos endereços são configurados no arquivo de propriedades). As exceções são classificadas pela linguagem Java em duas grandes categorias: *checked* e *unchecked*: as do tipo *checked* são exceções que sempre podem ocorrer, independentemente da programação, como falha no acesso ao sistema de arquivos, problema de acesso à internet etc e portanto possuem mensagens personalizadas no sistema PLUVI; já as *unchecked* não são previstas porque só ocorrem devido a falhas de programação, portanto foram tratadas com uma mensagem genérica pela sua característica inerente de imprevisibilidade.

O sistema também se comporta diferentemente dependendo se as falhas ocorridas são recuperáveis ou não. Erros desconhecidos na inicialização, falha na leitura do arquivo de propriedades ou na criação do arquivo de *log* são considerados bloqueantes e interrompem a execução do sistema; todas as demais falhas são tratadas como "avisos" aos administradores e não interrompem o funcionamento, visto que elas podem ser corrigidas independentemente de qualquer ação (como instabilidade na rede) ou após intervenção de um administrador (por exemplo, corrigindo um valor errado no arquivo de propriedades).

Testes Automatizados

Para garantir a consistência do sistema ao longo do tempo, foi criada uma série de testes unitários, distribuídos nas classes CEMADENTEST e STATIONDATATEST, apoiados por uma classe UTIL. Tais testes cobrem as principais funcionalidades do sistema, sendo executados regularmente durante o desenvolvimento e obrigatoriamente antes de cada entrega de versão à Defesa Civil. Eles poderiam ser executados periodicamente em uma solução de integração contínua como o Jenkins (https://jenkins.io) caso outros desenvolvedores de *software* fossem incorporados ao desenvolvimento do sistema, mas como o produto no estágio atual foi desenvolvido por um único profissional, esse procedimento não foi considerado necessário.

Para verificar a cobertura (percentual de código que está sendo testado), foi utilizado o EclEmma (Figura 5). Por se tratar de um protótipo, a cobertura ainda é incipiente em termos

quantitativos, mas avançada em termos qualitativos, pois o foco dos testes se deu sobre os métodos mais críticos para o sistema e/ou com a maior quantidade de lógica envolvida.

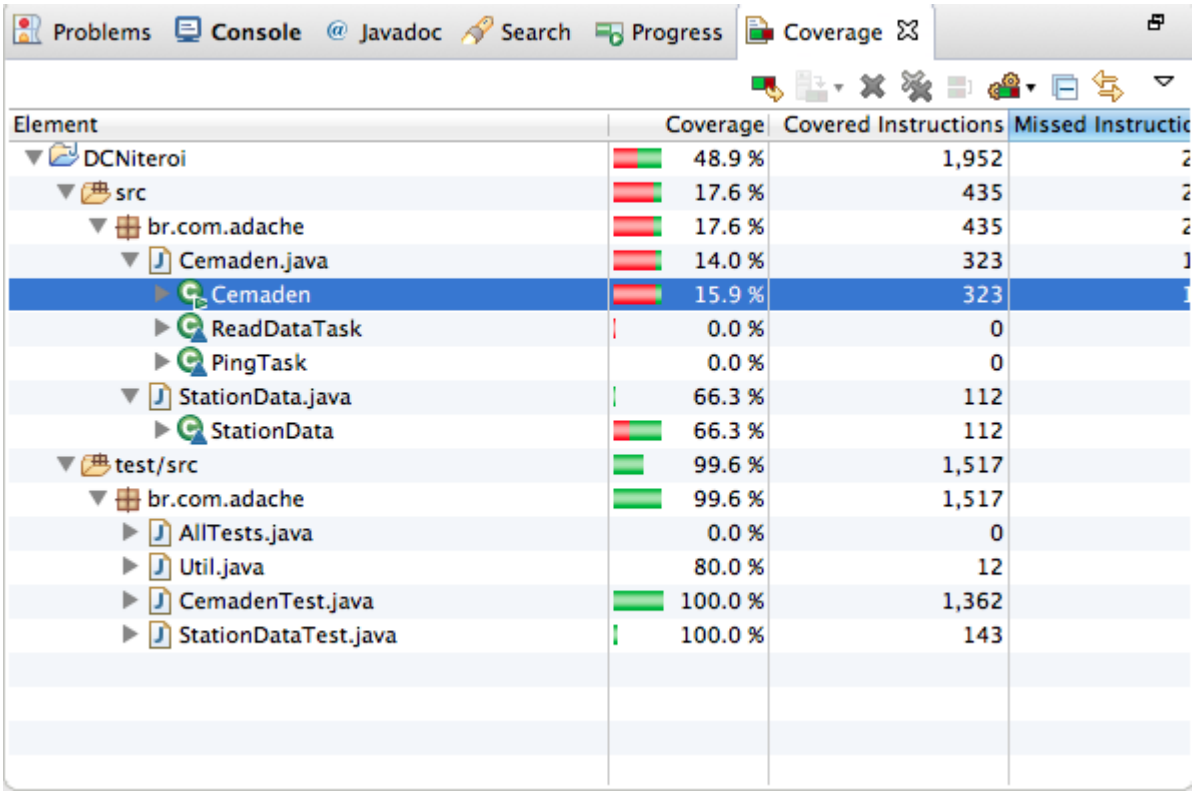


Figura 5: cobertura de testes com EclEmma

Modo Offline

O sistema possui um "modo *offline*", isto é, pode ser executado sem acesso à rede para testes da lógica. Para tal, na linha de comando deve ser passado um parâmetro "offline" ao sistema, que executará com dados de teste.

No "modo *offline*", o sistema não usa o Selenium; ao invés, utiliza dados pré-codificados para teste da lógica.

CAPÍTULO 4 – CONCLUSÕES E RECOMENDAÇÕES

Este trabalho buscou demonstrar que com uma combinação criativa de tecnologias de uso gratuito (não necessariamente *softwares* "livres" ou de código aberto, conforme definição em FREE SOFTWARE FOUNDATION, 2016) e o apoio de um profissional da área, é possível obter ganhos significativos para o desempenho das Defesas Civis Municipais. Este objetivo geral foi atingido e desdobrado em dois objetivos específicos, que foram atingidos por meio respectivamente das soluções 1 e 2. A solução 1 já beneficia a DC de Niterói a custo zero, e a solução 2 também promete cumprir com o objetivo geral, assim que forem concluídos os testes para sua implantação efetiva.

O trabalho desenvolvido em parceria com a Defesa Civil Municipal de Niterói permitiu a esse órgão realizar uma análise de risco muito mais efetiva, mapeando geograficamente com riqueza de detalhes uma ampla gama de informações úteis para a prevenção, preparação e resposta a situações de emergência, como a localização e delimitação de áreas vulneráveis, a proximidade de unidades de combate, de resgate e assistência médico-hospitalar, congregando telefones, endereços e outras informações relevantes. O acesso a essas informações é feito de forma remota por qualquer dispositivo móvel, proporcionando flexibilidade aos agentes da Defesa Civil e facilitando a coordenação de ações daquele órgão. Igualmente, a automatização do processo de acompanhamento de níveis críticos de pluviosidade permite uma maior agilidade e riqueza de informações, proporcionando um acompanhamento 24/7 efetivo, ao mesmo tempo em que faculta a alocação da força de trabalho em outras tarefas, potencializando o escopo de atendimento.

A avaliação dos processos de trabalho identificou diversas outras oportunidades de otimização em outras áreas, com aplicação direta para outras defesas civis municipais do país exigindo pouca ou nenhuma personalização, o que traria um ganho de escala capaz de reduzir os já escassos investimentos a praticamente zero, maximizando o benefício.

SUGESTÕES E TRABALHOS FUTUROS

A solução 2 (Sistema PLUVI) apresenta diversas possibilidades de melhorias, como:

1. Extração de outras informações úteis para a DC pelo processamento dos dados brutos do Cemaden, tais como o acumulado máximo de uma estação em um período longo, a média ou mediana de acumulados, a comparação com o volume total esperado para o período (série histórica, etc).

2. Recuperação de dados de outras fontes, como arquivos FTP, ou pluviômetros de propriedade do Estado.
3. Outras formas de envio de alertas, como SMS, mensagens de WhatsApp, etc.

A avaliação dos processos apontou diversas possibilidades de otimização; cada uma delas poderiam ser objeto de trabalhos futuros ou mesmo se desdobrar em diferentes trabalhos, como:

- Banco de Dados (BD) de consolidação da avaliação de riscos dos voluntários;
- Módulo para Elaboração de Planos de Emergência;
- SIG/BD para registro de óbitos e danos decorrentes de desastres;
- SIG/BD consolidando índices de ocorrências;
- Portal *web* para comunicação interna e externa.

Por fim, o mesmo procedimento de avaliação de processos poderia ser realizado junto a outras prefeituras, resultando numa ampla gama de possibilidades para serem exploradas.

CAPÍTULO 5 - REFERÊNCIAS

BROCKLESBY, J. Elements of meteorology with questions for the examination, designed for schools and academic. 4th ed. New York, 1851, Sheldon and Company, 292p.

CIF-OIT (2012): CENTRO INTERNACIONAL DE FORMAÇÃO DA OIT – CIF OIT**.** *O risco de desastres: origem, avaliação, redução e prevenção no quadro do desenvolvimento local sustentável.* Curso de Redução do Risco de Desastres e Desenvolvimento Local Sustentável, Unidade Didática 2. Programa Delnet de Apoio ao Desenvolvimento Local. Centro Internacional de Formação da OIT, 2012.

CUNHA, Bruno Pereira da. O antes do Bumba: Análise do Sistema Político sob a ótica das teorias estruturalistas-construtivistas de risco. Dissertação, Curso de Mestrado Profissional em Defesa e Segurança Civil, Niterói/2016, 109 p.

ESRI. *ArcGIS Features.* Redlands, 2016. Disponível em: https://www.arcgis.com/features/features.html . Acesso em 18 out. 2016.

FREE SOFTWARE FOUNDATION. *O que é o software livre.* Boston, 2016. Disponível em: https://www.gnu.org/philosophy/free-sw.html . Acesso em 2 out. 2016.

GRAZULIS, T.P. *The Tornado: Nature's Ultimate Windstorm*. 1th ed. Norman, 2001, University of Oklahoma Press, 324p. ISBN 0-8061-3258-2.

IBGE. *Cidades*. Rio de Janeiro, 2016. Disponível em: http://cidades.ibge.gov.br/xtras/perfil.php?lang=&codmun=330330&search=rio-de-janeiro|niteroi|infograficos:-informacoes-completas. Acesso em 19 jun. 2016.

IRIN (2013a) "Integrated Regional Information Networks", *Quelles technologies utiliser dans la gestion des catastrophes naturelles?*, Bangkok, 30 mai 2013, http://www.irinnews.org/fr/report/98172/analyse-quelles-technologies-utiliser-dans-la-gestion-des-catastrophes-naturelles. Acesso em 19 jun. 2016.

IRIN (2013b) "Integrated Regional Information Networks", How technology is transforming emergency preparedness, Dakar, 16 ago. 2013, http://www.irinnews.org/analysis/2013/08/16/how-technology-transforming-emergency-preparedness . Acesso em 19 jun. 2016.

MINISTÉRIO DA INTEGRAÇÃO NACIONAL - MIN. *Histórico da Defesa Civil.* Brasília, 2012. Disponível em: http://www.mi.gov.br/historico-sedec. Acesso em 21 jun. 2016.

NARVÁEZ, Lizardo; LAVELL, Allan; ORTEGA, Gustavo Peres. *La gestión del riesgo de desastres: um enfoque basado em procesos.* 1ª Ed. Lima, Peru: Maiteé Flores Piérola - PULL CREATIVO S.R.L. 2009.

OLIVEIRA, A. B. *O fim da guerra fria e os estudos de segurança internacional: o conceito de segurança humana*. Aurora, ano III, n.5, 2009, p.68-79. ISSN: 1982-8004.

POIRIER, J. P. *Le tremblement de terre de Lisbonne 1755*. 1ere Ed. Paris, Odile Jacob, 2005, 288p. ISBN 2-7381-1666-3.

QGIS DEVELOPMENT TEAM. *QGIS Features.* 2016. Disponível em: http://docs.qgis.org/2.14/en/docs/user_manual/preamble/features.html. Acesso em 18 out. 2016.

REVISTA EXAME. São Paulo: Ed. Abril, n. 1064, 30 abr. 2014.

SOMMERVILLE, I., 2001. *Software Engineering*. Sixth Edition, Addison Wesley.

TAVARES, M.J.F.; AMADOR, F.; PINTO, M.S. *O terramoto de Lisboa de 1755: tremores e temores.* Cuadernos Dieciochistas, Ed. Universidad de Salamanca, n.6, 2005, p.43-77. ISSN: 1576-7914. Disponível em: http://gredos.usal.es/jspui/bitstream/10366/69060/1/O_terramoto_de_Lisboa_de_1755_tremores_e.pdf. Acesso em 16 mai. 2016.

UNITED NATIONS INTERNATIONAL STRATEGY FOR DISASTER REDUCTION - UNISDR. *Working Background Text on Terminology for Disaster Risk Reduction*. Geneva, 2015. Disponível em: http://www.preventionweb.net/documents/framework/Working%20background%20text%20on%20DRR%20Terminology%202%20October%20reissued%20on%2023%20October.pdf. Acesso em 19 jun. 2016.

APÊNDICE I - ARTIGO

Artigo aceito no I Congresso Brasileiro de Redução de Riscos de Desastres, realizado em Curitiba/PR, nos dias 12-15 de outubro de 2016.

ESTUDO DE ESTRATÉGIAS INFORMATIZADAS DE BAIXO CUSTO APLICADAS À DEFESA CIVIL NO MUNICÍPIO DE NITERÓI – RJ EM PREVENÇÃO DE DESASTRES

Leonardo Abreu de Barros[1], Airton Bodstein[2]
1 Universidade Federal Fluminense, ciclano@gmail.com
2 Universidade Federal Fluminense, airton@defesacivil.uff.br

RESUMO

O presente estudo tem como finalidade fornecer uma contribuição às defesas civis municipais de modo a torná-las mais eficientes e eficazes, por meio de soluções informatizadas de baixo custo aplicadas à gestão e tratamento de dados no âmbito da redução de riscos de desastres (RRD). Foi feito um estudo de caso junto à Defesa Civil Municipal da cidade de Niterói, RJ, e priorizada a área de Análise de Risco. Para o mapeamento de informações úteis em prevenção, preparação e resposta a emergências, foi experimentado e utilizado um software gratuito, o Google My Maps®, uma versão editável e um pouco menos conhecida do Google Maps®, e para a automatização do monitoramento dos níveis de pluviosidade foi aplicado um software desenvolvido neste trabalho, tendo ambas as soluções apresentado resultados bastante satisfatórios em termos de ampliação da gama de informações relevantes disponíveis e otimização de recursos.

Palavras Chave: defesa civil, município de Niterói, redução de riscos de desastres (RRD), software livre

LOW COST COMPUTERIZED STRATEGIES STUDY APPLIED TO CIVIL DEFENSE IN THE CITY OF NITERÓI - RJ - BRAZIL FOR DISASTER PREVENTION

ABSTRACT

This study aims to provide a contribution to the municipal civil defenses, in order to turn them more efficient and effective through low cost IT solutions applied to the management and processing of data in the context of disaster risk reduction (DRR). A case study was conducted with the Municipal Civil Defense of the city of Niterói, RJ, and the Risk Analysis area was prioritized. A free software, Google My Maps®, an editable and a little less known version than Google Maps® was tried and used for mapping useful information for prevention, preparedness and response to emergencies, and for automating the monitoring of rainfall, a software developed by the author was applied. Both solutions presented satisfactory results in terms of expanding the range of available relevant information and resource optimization.

Keywords: civil defense, Niterói city, disaster risk reduction (DRR), free software

1. INTRODUÇÃO

Os desastres sempre estiveram presentes na História da Humanidade, o que pode ser facilmente aferido ao se examinar textos religiosos e mitológicos. O Grande Dilúvio enfrentado por Noé e a erupção que destruiu as cidades gêmeas de Sodoma e Gomorra são exemplos bíblicos de desastres que assolaram civilizações muito antigas (TAVARES 2005). Estes eventos de causas naturais e decorrentes de fenômenos geodinâmicos, como sismos, erupções vulcânicas, tsunamis, deslizamentos de terra, inundações, tornados e outros foram ignorados por muito tempo, subestimados e praticamente desprovidos de prevenção. Acreditava-se que as causas dos desastres eram de natureza punitiva e que os seus efeitos eram incontestáveis, fatais e explicitamente determinados. No entanto, o avanço dos estudos científicos e técnicos permitiu caracterizar tais eventos, prever os seus efeitos, estabelecer e distinguir as causas naturais e antropogênicas, no sentido de melhorar a prevenção e a gestão de riscos.

No contexto dos desastres reportados acima, os tornados constituem um exemplo interessante, já que foram observados ao longo da história pelos povos de língua inglesa desde 1000 anos DC. Nos Estados Unidos de hoje, os relatos de tornados por esses habitantes remontam ao final dos anos 1500, durante aventuras expedicionárias que acabariam por levar à colonização do país. A primeira tempestade de vento relatada pelos americanos do norte data de cinco de julho de 1643 em Plymouth Rock e teve um americano nativo como vítima. Provavelmente não foi um tornado, embora alguns livros o classifiquem como tal (GRAZULIS 2001). Os tornados se sucederam em 1643, 1671, ou seja, ao longo dos séculos XV, XVI até os dias de hoje. Esses eventos passaram a acontecer com frequência na Nova Inglaterra, principalmente no Vale de Ohio, já nos meados de 1800, onde vários pesquisadores começaram a se preocupar em descrever o fenômeno, e o que se acreditava estar por trás deles, através de livros didáticos, como BROCKLESBY (1851). Em 1755, Lisboa foi seriamente atingida por tremores de terra importantes, acompanhados de tsunamis e de numerosos incêndios. O número de vítimas foi elevado, em torno de 60 000 mortos. Este evento terrível inspirou Voltaire em seu conto "Candide" a se questionar sobre a bondade do Criador e a existência do mal. O filósofo francês fez do terremoto em Lisboa um acontecimento intelectual (POIRIER 2005).

Apesar de todas essas constatações acerca de alguns exemplos de desastres no mundo ao longo da história, a percepção de risco que as pessoas têm em relação a esses eventos é bem diferenciada. Segundo a UNISDR (2015), "Desastre é uma séria interrupção no funcionamento de uma comunidade ou sociedade devido a eventos perigosos interagindo com condições de

vulnerabilidade e de exposição, levando a perdas e impactos humanos, materiais, econômicos e ambientais generalizados".

As primeiras ações, estruturas e estratégias de proteção e segurança dirigidas à população, tanto no Brasil como no resto do mundo, foram realizadas nos países envolvidos na Segunda Guerra Mundial. A Inglaterra foi o primeiro país perceber que somente as forças armadas não poderiam garantir a segurança da sua população e instituiu a *Civil Defense* (Defesa Civil), após os ataques sofridos entre 1940 e 1941, quando foram lançadas toneladas de bombas sobre as principais cidades e centros industriais ingleses, causando milhares de perdas de vidas na população civil. No Brasil o tema começou a ser tratado em 1942, após o afundamento dos navios militares Baependi, Araraquara e Aníbal Benévolo, no litoral de Sergipe, e do vapor Itagiba, no litoral do estado da Bahia.

A notícia dos afundamentos fez com que a população brasileira fosse às ruas, exigindo do governo uma resposta imediata aos ataques, que culminou com a declaração de guerra do Brasil contra a Alemanha e a Itália e a criação do Serviço de Defesa Passiva Antiaérea, em agosto de 1942. Em 1943, a denominação de Defesa Passiva Antiaérea foi alterada para Serviço de Defesa Civil, sob a supervisão da Diretoria Nacional do Serviço da Defesa Civil, do Ministério da Justiça e Negócios Interiores. Este órgão foi extinto em 1946, com o fim da 2ª Guerra Mundial, bem como suas Diretorias Regionais criadas nos Estados, Territórios e no Distrito Federal. Hoje, em todo o mundo, a Defesa Civil se organiza em sistemas abertos, com a participação dos governos locais e da população no desencadeamento das ações preventivas e de resposta aos desastres, seguindo o princípio da Defesa Comunitária.

Em função de fortes chuvas que assolaram a região Sudeste entre 1966 e 1967, provocando enchentes no Estado da Guanabara e deslizamentos na Serra das Araras/RJ e Caraguatatuba/SP, o Brasil começou a se estruturar em Defesa Civil (MIN 2012).

As décadas de 70 e 80 foram marcadas por uma série de desastres de causas naturais, especialmente na América Latina, o que levou a Organização das Nações Unidas (ONU) a proclamar a década de 1990 a Década Internacional para Redução dos Desastres Naturais (DIRDN); outra ação importante da ONU, por meio do seu Programa de Desenvolvimento das Nações Unidas (PNUD), foi lançar o conceito de "Segurança Humana" em seu relatório anual, que consistia em uma nova abordagem de segurança em nível individual, a partir de dois aspectos principais: manter as pessoas a salvo de ameaças crônicas como a fome, as doenças, a repressão e protegê-las de mudanças súbitas e nocivas nos padrões da vida cotidiana, com guerras, genocídios e desastres naturais (OLIVEIRA 2005).

Inspirado nessas resoluções, em 1995 o Brasil lançou sua Política Nacional de Defesa Civil (PNDC), a qual representou um marco histórico na Defesa Civil nacional, alinhada com o abrangente conceito de Segurança Humana proposto um ano antes pelo PNUD.

Em 2005, foram atualizadas a estrutura, a organização e as diretrizes para o funcionamento do SINDEC e do CONDEC, com a implantação do CENAD. Em março de 2010 foi realizada, em Brasília, a I Conferência Nacional de Defesa Civil e Assistência Humanitária. Com uma participação significativa da sociedade civil organizada, o tema Defesa Civil e Assistência Humanitária foi discutido por 1.179 municípios que realizaram conferências no âmbito municipal, intermunicipal ou regional, selecionando 1.495 delegados habilitados para a etapa Nacional, que aprovaram 104 proposições para reformular o sistema de Defesa Civil no país.

Neste mesmo ano, fortes chuvas ocorridas no Estado do Rio de Janeiro em janeiro e abril causaram mais de uma centena de mortos e desabrigados em função das inundações e dos deslizamentos ocorridos. Em janeiro de 2011, outro episódio de chuvas intensas ocorreu na Região Serrana deste mesmo Estado, com centenas de perdas humanas e materiais ainda maior, sendo considerado o pior desastre natural súbito da história do país.

As dificuldades enfrentadas para mitigar os efeitos das chuvas de 2010/2011 no âmbito Federal impulsionaram alterações na legislação, a fim de facilitar a transferência de recursos para ações de socorro, assistência às vítimas, restabelecimento de serviços essenciais e reconstrução de áreas atingidas por desastre, além de alterações no Fundo Especial para Calamidades Públicas (FUNCAP).

O contexto internacional representado pelo Marco de Ação de Hyogo, as proposições resultantes da Conferência Nacional de Defesa Civil e a magnitude das perdas humanas e materiais decorrentes das chuvas de 2010/2011 no Estado do Rio de Janeiro estabeleceram as bases para a substituição da Política Nacional de Defesa Civil e fomentaram, em julho de 2011 (Decreto n.7513/2011), a criação do Centro Nacional de Monitoramento e Alertas de Desastres Naturais (CEMADEN).

Como se pode depreender desse histórico, a evolução da política e do sistema nacional de defesa civil nacional sempre foi impulsionada por desastres com grandes perdas materiais e de vidas humanas, os quais geraram grande comoção e propiciaram fortes investimentos. A prioridade para ações preventivas de defesa civil, o envolvimento ativo dos três níveis da administração pública e a integração com as demais políticas setoriais pregada pela recente

(2012) PNPDEC depende, para se concretizar, de investimentos contínuos e significativos em capacitação e tecnologia; esses investimentos dependem de uma visão de longo prazo dos governantes e em última instância, da percepção de risco da população em geral. Entretanto, a compreensão plena dessa cadeia de causa-efeito depende de mudanças culturais drásticas que só acontecem ao longo de gerações. Para o momento presente, no qual os investimentos contínuos na infraestrutura de defesa civil ainda são escassos, são necessárias alternativas viáveis para perseguir os objetivos almejados pela Lei 12.608/12, daí a motivação para a busca de soluções efetivas que exijam baixo grau de investimento.

Objetivos

O objetivo deste trabalho consiste em mostrar, por meio de um estudo de caso, que o uso de *softwares* gratuitos, ou mesmo o desenvolvimento de soluções informatizadas simples e de baixo custo podem contribuir significativamente para tornar as defesas civis municipais mais eficientes, seja aumentando a produtividade do seu efetivo, reduzindo custos e/ou prazos, seja automatizando o trabalho manual e ampliando a gama de informações disponíveis para auxiliar a defesa civil em seu trabalho.

2. METODOLOGIA

A metodologia utilizada é a pesquisa descritiva e exploratória seguida de um estudo de caso junto à Defesa Civil Municipal da cidade de Niterói, RJ (DCMN). O estudo inicial procurou contemplar o levantamento detalhado da estrutura atual da Defesa Civil de Niterói, priorizando as áreas a serem otimizadas. A partir da experiência do autor na área de informática, foi feita uma apresentação sobre as possíveis áreas de atuação a serem otimizadas, com base em softwares para gestão de emergências existentes da indústria, e em conjunto com a coordenação do órgão, priorizou-se a Análise de Risco. Por sua vez, a equipe técnica da prefeitura forneceu as informações adicionais sobre a estrutura e funcionamento da DCMN, assim como metas e indicadores que poderiam ser melhorados.

Com base na análise dos dados levantados, foram consideradas algumas possibilidades de informatização de processos, genericamente chamadas de "módulos" - os quais poderiam tanto ser utilizados de forma isolada quanto integrados em um ou mais sistemas - para que estes fossem priorizados. Foi definido como prioritário um Sistema de Informações Geográficas (SIG) para apoio à Análise de Risco e detalhadas as camadas que deveriam ser visualizadas nesse módulo. Após análise das soluções gratuitas disponíveis, foi selecionado o *software* Google My Maps® (https://www.google.com/mymaps), a versão editável e um pouco menos

conhecida do Google Maps®, o qual foi utilizado para mapear diversas informações úteis para a Defesa Civil de Niterói.

Com esta primeira fase concluída, foi efetuada nova priorização. O módulo escolhido foi um sistema de alarme para níveis elevados de pluviosidade. A metodologia atualmente utilizada é o monitoramento dos níveis de chuva no município, com base principalmente em informações recuperadas no *site* do Cemaden – Centro Nacional de Monitoramento e Alertas de Desastres Naturais, do Ministério da Ciência e Tecnologia (http://www.cemaden.gov.br/mapainterativo), e os critérios para disparar os estados de alerta e vigilância são regidos por protocolo do Governo do Estado do Rio de Janeiro.

Um *software* foi então desenvolvido pelo autor com base em tecnologias *open source*, para: automatizar a leitura dos dados do *site* (cujos dados são atualizados a cada dez minutos); gerar um histórico das leituras em tempo real; e monitorar os níveis de chuva e disparar alertas por *e-mail* caso níveis críticos sejam atingidos. O *software* foi instalado na sede da Defesa Civil de Niterói e encontra-se atualmente em fase de testes.

3. ESTUDO DE CASO

Conforme apresentado na Metodologia, para ilustrar as possibilidades de ganho em eficácia e eficiência das defesas civis municipais com investimentos em soluções informatizadas de baixo custo, foi escolhida a Defesa Civil do município de Niterói, no estado do Rio de Janeiro.

Segundo dados do IBGE (2016), a estimativa para a população de Niterói era de 496.696 habitantes em 2015, distribuídos em uma área de 133,919 km^2. A cidade é considerada um dos principais centros financeiros, comerciais e industriais do Rio de Janeiro, tendo sido indicada pela Revista Exame (2014) como a 6ª entre as 100 melhores cidades brasileiras para negócios. Apresenta alto índice de investimentos imobiliário e comerciário. O desenvolvimento atual justifica-se pelo fato do município estar ligado a uma série de investimentos industriais importantes nos setores ligados à cadeia produtiva de petróleo e gás, o que leva a região a responder por 70% do parque instalado do setor, concentrando desde empresas *offshore*, com destaque para a reinauguração de estaleiros, bem como com a reforma e a manutenção de plataformas e estruturas *offshore*, além da construção de embarcações para o transporte de passageiros. O IBGE (2016) indica que o PIB nominal de Niterói é o quinto maior do estado do Rio de Janeiro, só ficando atrás da própria capital e dos municípios de Duque de Caxias, Campos dos Goytacazes e Macaé, além de ser o 45º município mais rico do Brasil. Embora

ocupe o 5º lugar quanto ao número de habitantes, o que corresponde a 4,11% do total da população da Região Metropolitana do Rio de Janeiro, Niterói é o segundo maior empregador formal do Estado, apresentando o melhor Índice de Desenvolvimento Humano e o terceiro melhor do país. Além disso, a cidade está entre as mais alfabetizadas do país, sendo o melhor nível de alfabetização do estado, apresentando também, a menor incidência de pobreza, população com maior renda mensal per capita e o maior índice de longevidade do Rio de Janeiro (IBGE 2016).

1.1. A Defesa Civil Municipal de Niterói

A Defesa Civil Municipal de Niterói conta hoje com aproximadamente quarenta pessoas, divididas entre as áreas de Minimização de Desastres, Operacional, Monitoramento e Administrativo. Segundo o coordenador, esse efetivo é insuficiente para atender à demanda de atendimentos que chegam por telefone, e-mail ou presencialmente, e na maior parte das vezes requerem vistoria presencial; só em 2013, por exemplo, foram 850 pedidos, dos quais 650 foram atendidos. A DCMN conta com um sistema informatizado que permite registrar e priorizar esses pedidos, mas a priorização é manual, e por conta do baixo efetivo, acaba sendo deficiente. A meta do órgão é conseguir atender todas as vistorias preventivas em até 15 dias, e as emergenciais em 24h. É importante destacar que esses pedidos abrangem somente as vistorias responsivas, isto é, disparadas em função da demanda da população e não consideram as vistorias preventivas, de responsabilidade do Estado para garantir a segurança dos moradores.

Outro fator limitante para a atuação da DCMN são as delicadas relações interinstitucionais que permeiam a elaboração de planos de emergência. A instituição possui o papel central de planejar e coordenar a resposta dos diferentes atores envolvidos durante a ocorrência de uma situação de emergência, como o corpo de bombeiros, as polícias civil e militar, secretarias municipais e órgãos de trânsito, mas não dispõe de uma ascendência hierárquica sobre essas entidades. Como consequência, para evitar o risco de ingerência nos processos de cada órgão, os planos de contingência do município não podem se aprofundar ao nível de detalhes, necessário para um resultado altamente otimizado, limitando-se a listar as responsabilidades dos envolvidos.

Um exemplo do impacto dessa limitação foi um caso de escorregamento ocorrido em Charitas (bairro de Niterói), em que uma escola ficou com acesso dificultado. A diretoria da escola dependia de instruções da DCMN para saber se deveria reter ou liberar as crianças, mas por sua vez dependia de informações de trânsito atualizadas do órgão responsável; como não havia planos de emergência preparados para este caso e consequentemente não havia protocolos

pré-acordados, a resposta demorou muito mais do que seria desejável para a situação em questão. O caso lembra outro incidente ocorrido durante o desastre mais severo da história recente de Niterói, o deslizamento do Morro do Bumba em 2010. Na ocasião, foi solicitada a colaboração de uma empresa que possuía um maquinário pesado, uma retroescavadeira, para a remoção de entulho, a qual se encontrava em um galpão da empresa no Rio de Janeiro. A caminho do atendimento, o veículo foi barrado na entrada da ponte Presidente Costa e Silva, popularmente conhecida como Ponte Rio - Niterói, que liga as duas cidades, por falta de protocolo prévio. Foram necessárias várias horas e o envolvimento de várias autoridades, inclusive do Governador do Estado, para que fosse permitida a passagem do veículo.

As restrições de pessoal impactam também no pós-desastre, fase na qual a DCMN se envolve para vistoriar os danos e autorizar o restabelecimento de comércio e serviços. Uma meta da DCMN seria poder registrar e tornar mais visíveis os danos secundários, como influências negativas no mercado imobiliário, impacto no comércio, absenteísmo no trabalho, doenças crônicas, etc. Infelizmente, a falta de recursos já dificulta mesmo o levantamento dos dados primários e inviabiliza completamente uma atuação mais abrangente.

Por fim, é importante citar as restrições financeiras da DCMN, comum a todos os órgãos da Defesa Civil. Estas competem pelos recursos orçamentários da prefeitura com todos os demais órgãos, e possuem a desvantagem de não gerar receita. Portanto, a saúde financeira do órgão depende muito da visão do prefeito sobre os benefícios da prevenção, a preocupação com a imagem política, etc. No caso de Niterói, o já citado desastre do Morro do Bumba foi um divisor de águas para a Defesa Civil do município, que contava com um efetivo de quatro pessoas e nenhuma viatura (o coordenador se deslocava em veículo particular). Após o desastre e a visibilidade negativa provocada pelo desastre, o efetivo decuplicou, a DCMN ganhou uma sede nova e muita infraestrutura. Com o distanciamento temporal do evento, a tendência é que esses recursos tornem a escassear. Portanto, para qualquer solução que busque aperfeiçoar os processos de trabalho da entidade, é preciso priorizar soluções de baixo custo.

4. RESULTADOS E DISCUSSÃO

1.2. Levantamento dos processos atuais

Após o levantamento da estrutura atual da Defesa Civil de Niterói (já detalhado na Introdução) e da priorização de áreas a serem otimizadas, deu-se início ao estudo.

A Análise de Risco foi escolhida como a área prioritária a ser otimizada. A justificativa da coordenação para essa escolha partiu do fato de que essa área constitui a base para todas as

demais, e que se esta for mal elaborada ou insuficiente, todas as ações seguintes seriam prejudicadas.

1.1.1. Riscos

A Defesa Civil Municipal considera como o maior risco para a população de Niterói os movimentos de massa. Ressacas e queimadas também são monitoradas pelo órgão, mas apresentam impacto e incidência significativamente menores. Os movimentos de massa normalmente são deflagrados por chuvas intensas mas que, geralmente, estão associados a outras variáveis, tais como:

- Declividade do terreno;
- Má ocupação do solo (moradias de baixo padrão);
- Baixa capacidade de drenagem;
- Vazamentos na rede de abastecimento de água, tanto oficial quanto clandestina, o que encharca o solo.

O mapeamento de riscos normalmente é feito *in loco*, com o deslocamento de um agente da Defesa Civil apoiado por um geólogo (atualmente só existe um profissional com essa formação atuando junto à DCMN). Devido à falta de profissionais suficientes para cobrir toda a extensão do município, o geólogo elaborou um modelo de levantamento preliminar que pode ser feito por um técnico sem formação na área de geologia, para posteriormente ser confirmado pelo referido especialista. A população - geralmente voluntários treinados pela DC - também colabora levantando os riscos nas regiões em que habitam e retornando à DCMN por diversos meios e formatos, como telefone, WhatsApp® ou presencialmente. Entretanto, por não estarem estruturados e não serem feitos por especialistas, esses relatos precisam ser confirmados e trabalhados para serem convertidos no modelo existente. Tanto os levantamentos feitos por especialistas como aqueles que partiram da população são consolidados hoje em um banco de dados na sede da DCMN, o que exige um tempo razoável de digitação dos dados.

Está em curso um trabalho conjunto com um professor e bolsistas da UFF, envolvendo um software livre para mapeamento de vulnerabilidades em comunidades carentes.

1.1.2. Monitoramento e Alerta

Existe uma área especializada em monitoramento do tempo dentro da DCMN, composta por meteorologistas e outros especialistas, responsável por monitorar os boletins de tempo de órgãos como o Cemaden e o INPE, atenta a sinais que possam acarretar situações de

emergência no município. Diversas variáveis são monitoradas e analisadas, sendo uma das principais os níveis de precipitação acumulados.

O governo do estado apresenta um Protocolo (ver Tabela 1) que define um índice de escorregamento, ou seja, estabelece níveis de chuva acumulada, os quais, quando atingidos, aumentam significativamente a chance de escorregamento, situação em que as Defesas Civis Municipais devem ficar em alerta. É importante frisar que, sendo uma tabela para todo o estado, é baseado em médias e, portanto não muito preciso; a DCMN iniciou um estudo interno para refinar esse índice para o município.

Tabela 1 - protocolo estadual para acionamento de sirenes

1h s/acumulado	1h c/acumulado	24h	48h	96h	Mensal
>= 45mm	>= 40mm	>= 100mm	>= 125mm	>= 175mm	>= 300mm

No caso de acúmulo de chuva ou chuva muito intensa, planeja-se a evacuação de comunidades vulneráveis; isto é feito com até dois dias de antecedência, dependendo da previsão. Uma vez atingido um limiar de alerta, é disparado um informe para órgãos envolvidos na resposta, como Corpo de Bombeiros, Secretaria de Trânsito, Prefeitura etc., via WhatsApp®, por telefone e/ou por e-mail. Quando o índice crítico é atingido, são disparadas as sirenes nas comunidades; essa confirmação é feita com até uma hora de antecedência. Os agentes da DCMN se deslocam para bases próximas às comunidades vulneráveis, chamadas "pontos de apoio", juntamente com representantes de órgãos de resposta, e ficam de prontidão. Cerca de quinze minutos antes da deflagração do evento, a população começa a ser evacuada. Este último passo é bem delicado, pois os moradores são bem reticentes quanto a abandonar suas casas, e por isso é feito por último, quando existe certeza praticamente absoluta de um evento extremo.

No caso de queimadas (só é de competência da DCMN incêndios em florestas e vegetação urbana), o alerta inicial é recebido pelo Corpo de Bombeiros, que notifica a Defesa Civil, diferentemente do que costuma acontecer em outras ocorrências; para melhor tratar essa questão, a DCMN realizou um convênio com o Corpo de Bombeiros para troca de informações sobre queimadas e incêndios em geral, que envolve a presença de um profissional Bombeiro em tempo integral na sede da DCMN. Analogamente, para ressacas ou outros fenômenos marítimos, o aviso inicial é feito pela Marinha.

Segundo a coordenação, a DCMN faz uso de dois sistemas de comunicação para recebimento de informações e alarmes: um sistema informatizado para envio de SMS, o qual

funciona razoavelmente bem, mas apresenta algumas falhas e limitações; e a ferramenta de mensagens instantâneas WhatsApp®, na qual foram criados vários grupos, alguns envolvendo autoridades e outros líderes comunitários. Esta última ferramenta apresenta como principal vantagem a praticidade, mas além da dificuldade de processamento das mensagens não estruturadas, tem enfrentado recentemente o risco de bloqueios ocasionais por iniciativa da Justiça brasileira, o que compromete sua confiabilidade para situações de emergência em que cada minuto pode fazer diferença.

1.1.3. Preparação para Emergências

Para mapear trilhas de evacuação em comunidades, a DCMN utiliza um aplicativo gratuito chamado Wikiloc® (http://pt.wikiloc.com/wikiloc/home.do). Este aplicativo utiliza mapas do Google Maps® para, com o auxílio de um celular com GPS, permitir a qualquer pessoa criar suas próprias trilhas, que podem ser compartilhadas com outros.

1.1.4. Reconstrução

Uma maior atuação da Defesa Civil na fase de Reconstrução constitui uma meta para a atual coordenação, atuação esta que ainda é deficiente por conta da limitação de recursos. Seria desejável dispor de uma ferramenta que permitisse e incentivasse as equipes a registrar o que se chamou de "Dados Secundários", ou seja, danos decorrentes de uma ocorrência, mas não imediatamente visíveis, como o impacto negativo no mercado imobiliário, influência no comércio da região, doenças crônicas na população da região etc. A ferramenta poderia ter um tipo de assistente que sugerisse perguntas ao agente que estivesse levantando dados. Seria também importante que esse aplicativo fosse móvel, para permitir o registro *in loco*. Isto passaria uma visão clara para a prefeitura sobre esses danos "invisíveis".

1.1.5. Indicadores

Como resposta ao questionamento sobre quais indicadores poderiam ser levantados e otimizados a fim de melhorar a eficiência da DCMN e a sua maior visibilidade junto à Prefeitura e à comunidade em geral, os seguintes índices foram citados como candidatos:

1. Índice de incêndios em comunidades: medido por unidade de tempo, este índice precisa ser o menor possível. A atuação da DC consiste atualmente na conscientização da população, cabendo tradicionalmente ao Corpo de Bombeiros a resposta aos eventos.
2. Nº de comunidades preparadas: uma comunidade é considerada "preparada" quando conta com um grupo de voluntários capacitados pela DC, treinados para fazer avaliação de riscos e informar ao órgão. Hoje (Junho 2016) o município conta com trinta e duas comunidades preparadas, para um total de mais de cem comunidades. A meta da DCMN

para o ano de 2016 é conseguir mapear quarenta comunidades. Seria desejável que estas informações fossem disponibilizadas em um mapa georreferenciado que permitisse a visualização rápida da quantidade de comunidades preparadas.

3. Nº de ocorrências atendidas: um dos maiores desafios para a DCMN envolve a vistoria de locais sujeitos a perigos, que exige a presença de agentes. Como já mencionado anteriormente, é preciso realizar, tanto vistorias preventivas, de iniciativa do próprio órgão, a partir do seu mapeamento de áreas vulneráveis e histórico de ocorrências, quanto vistorias em resposta a demandas da população. Para as vistorias responsivas, a o órgão tem uma taxa de atendimento que gira em torno de 75%, mas este número precisa ser melhorado, seja pela agregação de um maior efetivo ou otimização do processo de atendimento.
4. Nº de eventos que registraram vítimas: óbitos são sempre a pior consequência possível de um evento extremo e todos os esforços precisam ser empreendidos no sentido de reduzir seu número ao mínimo. Portanto é preciso atuar na fonte das ameaças, eliminando-as ou mitigando-as quando possível, evitando a exposição de populações vulneráveis, conduzindo orientações e remoções, ou tão somente facilitando a evacuação dessas populações em situações de emergência. Um número baixo para este indicador é a meta mais importante para qualquer Defesa Civil em nível municipal.
5. Percentual de avisos procedentes/alarmes falsos: existe sempre um grau de incerteza quanto à previsão de ocorrência de uma emergência, e isso deixa a DC em uma situação delicada quando precisa alertar a população e outras autoridades competentes da aproximação de uma ameaça, visto que a mobilização de todo um efetivo e a evacuação de habitantes gera um transtorno só compensado pela confirmação do evento em si. Na hipótese de um "alarme falso", a credibilidade da DCMN é comprometida, portanto é necessário um controle rígido sobre a confiabilidade dos alertas antes destes serem levados adiante.

1.3. Possibilidades de otimização por meio de Sistemas Informatizados

Após a análise dos dados levantados, foram repassadas à DCMN algumas possibilidades de informatização de processos, genericamente chamadas de "módulos" - os quais poderiam tanto ser utilizados de forma isolada quanto integrados em um ou mais sistemas - para que estes fossem priorizados. Os módulos foram categorizados por fase de atuação, conforme descrito a seguir.

1.1.6. Análise de Risco

Para esta fase, foram propostos os seguintes módulos:

1. Sistema de Informações Geográficas (SIG) mapeando todas as informações de interesse em diferentes camadas, como: declividade do terreno, pluviosidade, moradias de risco, capacidade de drenagem do solo, nível de encharcamento do solo, vazamentos no abastecimento de água, áreas sujeitas a ressacas, áreas sujeitas a incêndios, áreas de risco etc.

2. Alarme de ocorrências, disparado por atingimento de índices críticos de pluviosidade (para alagamentos, enxurradas e deslizamentos). Para ressacas, seria necessário monitorar o nível do mar, e para incêndios, criar uma interface de comunicação automática com o Corpo de Bombeiros. O módulo poderia realizar notificação de ocorrências por e-mail, SMS (poderia se integrar com o sistema em uso de envio de SMS) ou aplicativos de mensagens instantâneas como o WhatsApp®. Esta ferramenta poderia também fazer um registro dos alarmes emitidos e facultar a categorização desse histórico em alarmes falsos e procedentes, o que permitiria a elaboração de relatórios.
3. Banco de Dados (BD) de consolidação da avaliação de riscos dos voluntários. Os voluntários treinados procuram identificar os riscos presentes em suas comunidades, e repassam essa informação à Defesa Civil por diversas vias, como WhatsApp®, SMS, telefone ou mesmo contato pessoal. Seria interessante ter uma via oficial, estruturada, de coleta e consolidação dessas informações em um banco de dados que pudesse gerar informação útil para a Defesa Civil. Este módulo também poderia potencialmente se integrar com o software livre sendo desenvolvido pela UFF para mapeamento de riscos em comunidades.

1.1.7. Preparação para Emergências

Para esta fase, foram propostos os seguintes módulos:

1. Módulo para Elaboração de Planos de Emergência: poderia ser desenvolvido, ou adaptado a partir de sistemas existentes da indústria, um módulo específico para elaboração de planos de emergência. Este módulo incluiria desde a concepção do plano, iniciando pela identificação de cenários de emergência, passando pela elaboração de estratégias para combate que se desdobrariam em ações, até a confecção do documento em si, que poderia ser estruturado de forma a facilitar a simulação e validação das estratégias exploradas. Este módulo poderia se integrar com o Wikiloc® para o mapeamento das trilhas de evacuação.
2. Sistema de Informações Geográficas, conforme já proposto na seção anterior, adicionando uma camada para mapeamento do número de comunidades preparadas, apresentando a quantidade absoluta dessas comunidades e seu percentual em relação ao total.

1.1.8. Resposta a Emergências

Para esta fase, foi proposto o seguinte módulo:

1. SIG/BD consolidando índices de ocorrências, como índices de incêndios em comunidades, com visão geográfica, tabular ou gráfica.

1.1.9. Reconstrução

Para esta fase, foi proposto o seguinte módulo:

1. SIG/BD que permitisse o registro de óbitos e danos decorrentes de desastres. A entrada de dados seria feita em dispositivos móveis como celulares ou *tablets* e contaria com um assistente, que orientasse o usuário sobre as informações que precisariam ser coletadas. Este módulo poderia disponibilizar relatórios georreferenciados para apoio à decisão.

1.1.10. Comunicação

Para esta fase, foi proposto o seguinte módulo:

1. Portal web para comunicação interna e externa, integrado com site oficial, contendo informações como material de treinamento para voluntários, números oficiais, sala de imprensa, Informações à população, notícias, planos de emergência públicos, trilhas de evacuação, histórico de ocorrências, registro de atendimentos, etc.

1.4. Otimização dos processos

1.1.11. Sistema de Informações Geográficas

O SIG foi selecionado pela Coordenação como prioritário para apoio à Análise de Risco, cujas camadas deveriam ser:

- áreas vulneráveis e áreas com comunidades preparadas;
- associações de moradores e NUDECs (Núcleos Comunitários de Defesa Civil);
- delimitação de cada comunidade;
- pluviômetros do Cemaden e outros sensores;
- unidades de saúde e SAMU (Serviço Ambulatorial Móvel Urbano);
- unidades do Corpo de Bombeiros, da PM (Polícia Militar) e da GM (Guarda Municipal);
- escolas;
- unidades prisionais;
- órgãos da Prefeitura;
- bases da CLIN (Companhia de Limpeza Urbana de Niterói).

De acordo com a Coordenação da DCMN, seria interessante o acesso móvel a esse sistema, para a utilização das equipes em deslocamento. No momento em que o levantamento dos dados para a elaboração deste trabalho estava sendo realizado, já estava em andamento o mapeamento da maior parte das camadas descritas acima por uma agente do órgão no SIG gratuito Google Earth®, que só era consultado na máquina em que havia sido instalado.

Em consulta à documentação oficial do Google Earth® (https://www.google.com.br/intl/pt-BR/earth/), verificou-se que existia a possibilidade de acesso remoto a mapas criados pela ferramenta, usando um *plug-in* (complemento instalado em navegadores *web*, permitindo estender suas funcionalidades). Entretanto, além do inconveniente de depender da instalação desse *plug-in*, e da verificação de compatibilidade entre versões de navegadores, a consulta aos endereços

https://support.google.com/earth/answer/178389?hl=pt-BR&ref_topic=2376748 e https://developers.google.com/earth/ mostrou que o *plug-in* era parte de uma API (*Application Programming Interface*, isto é, uma interface criada para desenvolvedores de *software*) que estava sendo descontinuada desde 2014, por razões de segurança, e seria suspensa definitivamente em 2016. Portanto, essa não era uma opção viável e outras alternativas precisavam ser buscadas.

Como desenvolver ou mesmo adaptar um SIG existente exigiria um investimento financeiro elevado, buscou-se outras soluções gratuitas disponíveis, e a opção mais natural foi o Google My Maps® (https://www.google.com/mymaps), a versão editável e um pouco menos conhecida do Google Maps®. A ferramenta não oferece os recursos de mapa 3D e navegação rica disponíveis no Google Earth® (que não eram importantes para a utilização desejada), mas apresenta a vantagem de não depender de instalação, sendo acessível por qualquer navegador, tanto em dispositivos fixos quanto móveis.

Após a realização de alguns testes, a proposta de uso do novo *software* foi apresentada à DCMN, tendo sido bem recebida (ver Figura 1). A profissional responsável pelo mapeamento achou a interface mais simples e intuitiva, embora bastante poderosa, com destaque à funcionalidade de edição simultânea por usuários diferentes no mesmo mapa, o que lhe permitiu dividir o trabalho de mapeamento com outros profissionais e com isso ganhar tempo. Para a Coordenação do órgão, a possibilidade de ver o resultado do mapeamento diretamente em um aparelho celular, com a possibilidade de ligar e desligar camadas e visualizar informações relacionadas a cada local, foi considerada bastante interessante. Foi descoberta a limitação de

dez camadas por mapa, a qual foi contornada criando-se outros mapas; a possibilidade de exportação de dados de um mapa para outro também ajudou a minimizar o problema.

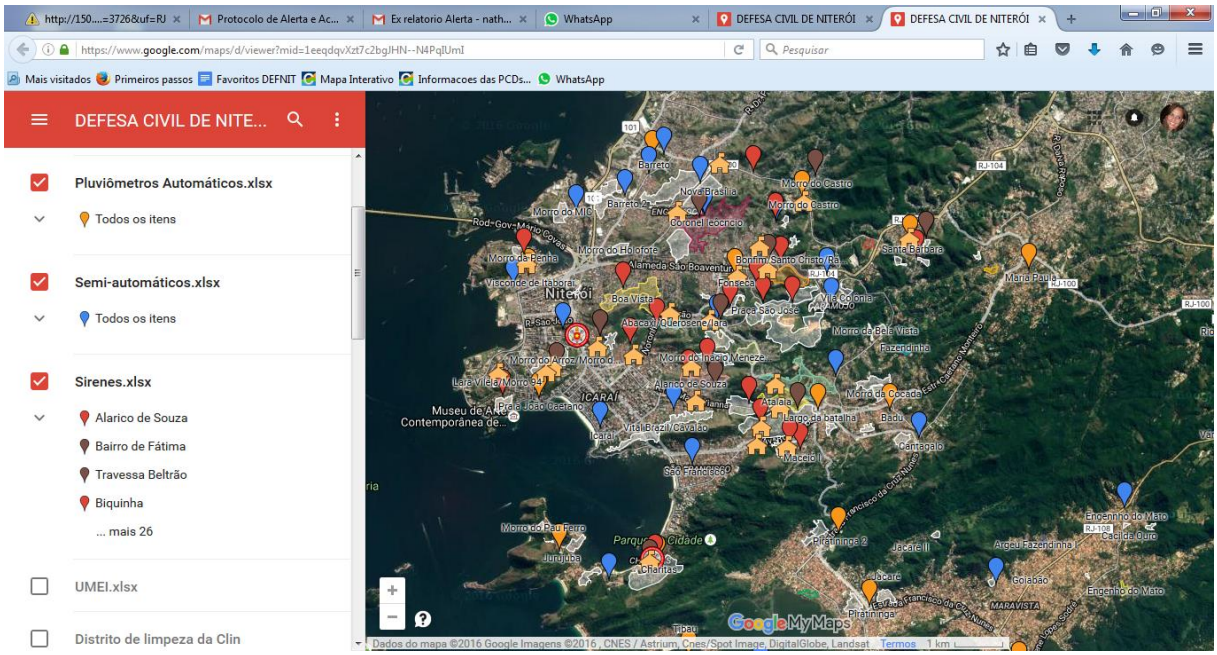


Figura 2. mapa de análise de risco da cidade de Niterói

1.1.12. Sistema de Monitoramento e Alarme

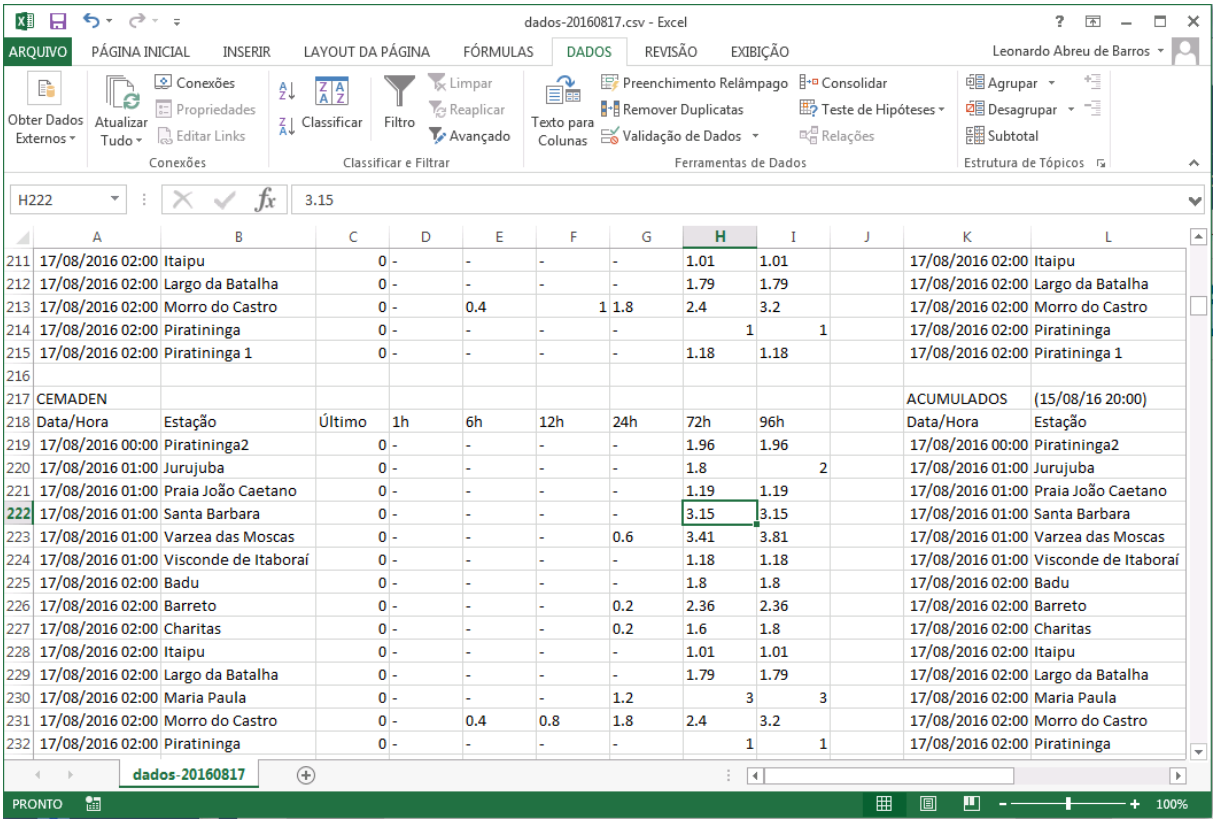
Conforme citado anteriormente, o módulo escolhido como segunda prioridade para otimização foi o de Alarmes de ocorrências. A meteorologista responsável explicou e demonstrou como eram hoje monitorados os níveis de chuva no município, bem como são disparados os estados de alerta e alarmes. Existem quatro níveis no monitoramento: o estado de "vigilância" é o padrão, alterado para "alerta" quando determinados limiares são atingidos; outros limiares determinam a passagem do estado de "alerta" para o estado de "alarme", e por fim a iminência de deslizamentos determina a passagem do estado de "alarme" para "alarme máximo".

Para monitorar dados de chuva, segundo a meteorologista, só existem hoje duas fontes disponíveis: os dados de pluviômetros disponibilizados pelo Cemaden e uma estação meteorológica completa, pertencente ao Governo do Estado. Os dados do Cemaden são recuperados manualmente via *site* www.cemaden.gov.br, e atualizados a cada dez minutos. Os dados da estação são disponibilizados via FTP em um servidor para o qual a DCMN possui credenciais de acesso.

Pela facilidade de acesso e maior disponibilidade de dados, optou-se por priorizar a automatização da leitura dos dados no *site* do Cemaden. Para isso, foi desenvolvido um *software* utilizando tecnologias gratuitas, como a linguagem Java (http://www.java.com), o

editor Eclipse (http://www.eclipse.org) e a ferramenta de automação Selenium (http://www.seleniumhq.org). A estrutura do *software* é bem simples e linear: a cada dez minutos (intervalo configurável em um arquivo de propriedades, assim como diversas outras características), o sistema consulta os dados de pluviômetros de Niterói no *site* do Cemaden, e quando encontra dados novos, soma-os com os lidos anteriormente, salva os dados lidos e acumulados em planilha (ver Figura 2), e processa os dados, comparando-os com uma tabela de níveis críticos de chuva fornecida pelo Estado do RJ (ver Tabela 1); se encontrar algum nível acima do limite, dispara *e-mails* de alerta aos responsáveis da Defesa Civil. O sistema foi preparado para ser tolerante a pequenas falhas no envio de dados do *site* do Cemaden e consegue inferir os dados corretos com base em uma análise algorítmica quando estes apresentam inconsistências.

O sistema possui pouco mais de 1000 linhas de código (LOC) e foi desenvolvido pelo autor. Ele é instalado como um serviço Windows e roda 24/7 em um servidor dedicado, recuperando-se automaticamente em caso de queda de energia. Alterações nas propriedades são recarregadas sem necessidade de reinício do sistema e eventuais erros são enviados por *e-mail* aos administradores cadastrados. O sistema, tal como está, poderia ser reutilizado em qualquer município do país coberto pela rede de pluviômetros do Cemaden, bastando-se apenas configurar nas propriedades o endereço de acesso aos dados desejados.



	A	B	C	D	E	F	G	H	I	J	K	L
211	17/08/2016 02:00	Itaipu	0	-	-	-	-	1.01	1.01		17/08/2016 02:00	Itaipu
212	17/08/2016 02:00	Largo da Batalha	0	-	-	-	-	1.79	1.79		17/08/2016 02:00	Largo da Batalha
213	17/08/2016 02:00	Morro do Castro	0	-	0.4	1	1.8	2.4	3.2		17/08/2016 02:00	Morro do Castro
214	17/08/2016 02:00	Piratininga	0	-	-	-	-	1	1		17/08/2016 02:00	Piratininga
215	17/08/2016 02:00	Piratininga 1	0	-	-	-	-	1.18	1.18		17/08/2016 02:00	Piratininga 1
216												
217	CEMADEN										ACUMULADOS	(15/08/16 20:00)
218	Data/Hora	Estação	Último	1h	6h	12h	24h	72h	96h		Data/Hora	Estação
219	17/08/2016 00:00	Piratininga2	0	-	-	-	-	1.96	1.96		17/08/2016 00:00	Piratininga2
220	17/08/2016 01:00	Jurujuba	0	-	-	-	-	1.8	2		17/08/2016 01:00	Jurujuba
221	17/08/2016 01:00	Praia João Caetano	0	-	-	-	-	1.19	1.19		17/08/2016 01:00	Praia João Caetano
222	17/08/2016 01:00	Santa Barbara	0	-	-	-	-	3.15	3.15		17/08/2016 01:00	Santa Barbara
223	17/08/2016 01:00	Varzea das Moscas	0	-	-	-	0.6	3.41	3.81		17/08/2016 01:00	Varzea das Moscas
224	17/08/2016 01:00	Visconde de Itaboraí	0	-	-	-	-	1.18	1.18		17/08/2016 01:00	Visconde de Itaboraí
225	17/08/2016 02:00	Badu	0	-	-	-	-	1.8	1.8		17/08/2016 02:00	Badu
226	17/08/2016 02:00	Barreto	0	-	-	-	0.2	2.36	2.36		17/08/2016 02:00	Barreto
227	17/08/2016 02:00	Charitas	0	-	-	-	0.2	1.6	1.8		17/08/2016 02:00	Charitas
228	17/08/2016 02:00	Itaipu	0	-	-	-	-	1.01	1.01		17/08/2016 02:00	Itaipu
229	17/08/2016 02:00	Largo da Batalha	0	-	-	-	-	1.79	1.79		17/08/2016 02:00	Largo da Batalha
230	17/08/2016 02:00	Maria Paula	0	-	-	-	1.2	3	3		17/08/2016 02:00	Maria Paula
231	17/08/2016 02:00	Morro do Castro	0	-	0.4	0.8	1.8	2.4	3.2		17/08/2016 02:00	Morro do Castro
232	17/08/2016 02:00	Piratininga	0	-	-	-	-	1	1		17/08/2016 02:00	Piratininga

Figura 2. amostra de dados coletados e acumulados do *site* do Cemaden

5. CONCLUSÃO

Este trabalho buscou demonstrar que com um baixo nível de investimento em tecnologia e o apoio de um profissional da área, com poucas horas de consultoria técnica, é possível obter ganhos significativos para o desempenho das Defesas Civis Municipais.

O trabalho desenvolvido em parceria com a Defesa Civil Municipal de Niterói permitiu a esse órgão realizar uma análise de risco muito mais efetiva, mapeando geograficamente com riqueza de detalhes uma ampla gama de informações úteis para a prevenção, preparação e resposta a situações de emergência, como a localização e delimitação de áreas vulneráveis, a proximidade de unidades de combate, de resgate e assistência médico-hospitalar, congregando telefones, endereços e outras informações relevantes. O acesso a essas informações é feito de forma remota por qualquer dispositivo móvel, proporcionando flexibilidade aos agentes da Defesa Civil e facilitando a coordenação de ações daquele órgão. Igualmente, a automatização do processo de monitoração de níveis críticos de pluviosidade permite uma maior agilidade e riqueza de informações, proporcionando um monitoramento 24/7 efetivo, ao mesmo tempo em que faculta a alocação da força de trabalho em outras tarefas, potencializando o escopo de atendimento.

O levantamento dos processos de trabalho identificou diversas outras oportunidades de aplicação das mesmas técnicas e tecnologias em outras frentes, com aplicação direta para outras defesas civis municipais do país exigindo pouca ou nenhuma personalização, o que traria um ganho de escala capaz de reduzir os já escassos investimentos a praticamente zero, maximizando o benefício.

6. **REFERÊNCIAS**

BROCKLESBY, J. *Elements of meteorology with questions for the examination, designed for schools and academic*. 4th ed. New York, 1851, Sheldon and Company, 292p.

GRAZULIS, T.P. *The Tornado: Nature's Ultimate Windstorm*. 1th ed. Norman, 2001, University of Oklahoma Press, 324p. ISBN 0-8061-3258-2.

IBGE. *Cidades*. Rio de Janeiro, 2016. Disponível em: http://cidades.ibge.gov.br/xtras/perfil.php?lang=&codmun=330330&search=rio-de-janeiro|niteroi|infograficos:-informacoes-completas. Acesso em 19 jun. 2016.

MINISTÉRIO DA INTEGRAÇÃO NACIONAL - MIN. *Histórico da Defesa Civil*. Brasília, 2012. Disponível em: http://www.mi.gov.br/historico-sedec. Acesso em 21 jun. 2016.

OLIVEIRA, A. B. *O fim da guerra fria e os estudos de segurança internacional: o conceito de segurança humana*. Aurora, ano III, n.5, 2009, p.68-79. ISSN: 1982-8004.

POIRIER, J. P. *Le tremblement de terre de Lisbonne 1755*. 1ere Ed. Paris, Odile Jacob, 2005, 288p. ISBN 2-7381-1666-3.

REVISTA EXAME. São Paulo: Ed. Abril, n. 1064, 30 abr. 2014.

TAVARES, M.J.F.; AMADOR, F.; PINTO, M.S. *O terramoto de Lisboa de 1755: tremores e temores*. Cuadernos Dieciochistas, Ed. Universidad de Salamanca, n.6, 2005, p.43-77. ISSN: 1576-7914. Disponível em: http://gredos.usal.es/jspui/bitstream/10366/69060/1/O_terramoto_de_Lisboa_de_1755_tremores_e.pdf. Acesso em 16 mai. 2016.

UNITED NATIONS INTERNATIONAL STRATEGY FOR DISASTER REDUCTION - UNISDR. *Working Background Text on Terminology for Disaster Risk Reduction*. Geneva, 2015. Disponível em: http://www.preventionweb.net/documents/framework/Working%20background%20text%20on%20DRR%20Terminology%202%20October%20reissued%20on%2023%20October.pdf. Acesso em 19 jun. 2016.

APÊNDICE II – MANUAL DO SISTEMA

Guia de instalação e uso do sistema “PLUVI”.

GUIA DE INSTALAÇÃO E USO DO SISTEMA PLUVI

Requisitos

Hardware mínimo

- Pentium 4 ou mais recente que suporte SSE2
- 512 MB de RAM
- 200 MB de espaço em disco

Software

- Microsoft Windows XP SP2 ou superior
- Mozilla Firefox 44.0.1-32 bits (somente esta versão)
- Java SE Runtime Environment (JRE) 1.7.0_79-b15 ou superior
- Microsoft Excel (qualquer versão)

Instalação

1. Expandir o arquivo *pluvi.zip* em uma pasta qualquer do Windows.
2. Na pasta *conf*, editar o arquivo *wrapper.conf*:

    a. Procurar a propriedade *wrapper.java.command* e inserir o caminho absoluto onde o Java se encontra instalado.
3. Na pasta *conf*, editar o arquivo *config.properties*:

    a. Procurar a propriedade *AMBIENTE* e inserir o nome do local onde o sistema esté sendo instalado (exemplo: “Defesa Civil Niterói”)

    b. Procurar a propriedade *URL* e inserir o caminho da página do Cemaden com os dados dos pluviômetros de interesse. Esse caminho deve ser recuperado a partir do endereço www.cemaden.gov.br/mapainterativo, clicando no município desejado;

    c. Procurar a propriedade *EMAIL.AVISOS* e inserir uma lista de e-mails, separados por vírgulas, de agentes a serem avisados em caso de níveis críticos de chuva terem sido ultrapassados;

    d. Procurar a propriedade *EMAIL.ERROS* e inserir uma lista de e-mails, separados por vírgulas, de administradores a serem avisados em caso de falhas encontradas no sistema.

4. Na pasta *bin*, executar o arquivo (dois-cliques) *cemaden.bat* para verificar se o sistema está funcionando corretamente (o navegador deverá ser aberto automaticamente, os dados lidos e salvos em arquivos CSV na pasta *dados*).

5. Na pasta *bin*, executar o arquivo (dois-cliques) *Installcemaden-NT.bat* para instalar o serviço.

Uso do sistema

1. O sistema PLUVI é automático, portanto, não é necessário acioná-lo. Para recuperar o consolidado dos dados lidos, basta acessar a pasta *dados* e procurar o arquivo com o dia desejado.
2. O arquivo de dados (extensão CSV) pode ser aberto com duplo-clique no Windows Explorer ou diretamente pelo Microsoft Excel. Após aberto, deve-se selecionar a primeira coluna ("A"), clicar no menu "Dados" e no submenu "Texto para Colunas". Será então aberto um diálogo com o título "Assistente para conversão de texto para colunas".
3. Na tela do Assistente, deve-se escolher o tipo de dados "Delimitado" e clicar no botão "Avançar".
4. Na tela seguinte, marcar o delimitador "Vírgula" e clicar no botão "Concluir".
5. Se aparecerem os símbolos "######" na planilha, basta expandir a coluna em que se encontram para visualizar os dados.

APÊNDICE III – CÓDIGO-FONTE

Código-fonte do Sistema “PLUVI”, na linguagem Java.

Classe Cemaden

```
package br.com.adache;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.text.NumberFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import java.util.SortedSet;
import java.util.TimeZone;
import java.util.Timer;
import java.util.TimerTask;
import java.util.TreeSet;
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

import javax.mail.Address;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.swing.JOptionPane;

import org.openqa.selenium.By;
import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import br.com.adache.Cemaden.ErrorType;

public class Cemaden  {

	private Timer readDataTimer;
	private Properties props;
	private long lastInterval;
	private SimpleDateFormat fileNameFormatter;
	private boolean online;
	private SimpleDateFormat timestampFormatter;
	private SimpleDateFormat timestampParser;
	private NumberFormat floatFormatter;
```

```
	Logger logger;
	private TreeSet<StationData> accumulatedData;

	public static void main(String[] args) {
		boolean online = true;
		if (args.length == 1 && args[0].equalsIgnoreCase("OFFLINE")) online =
false;
		Cemaden c = new Cemaden(online);
		c.logSetup();
		try {
			ReadDataTask readDataTask = new ReadDataTask(c);
			readDataTask.run();
			PingTask pingTask = new PingTask(c);
			long day = 24 * 60 * 60 * 1000;
			new Timer().scheduleAtFixedRate(pingTask, 0, day);
		} catch (Exception e) {
			c.logAndSendError("Erro na inicialização do sistema", "Erro
inesperado", ErrorType.BLOCKING, e);
		}
	}

	String getStackTrace(Exception e) {
		StringBuilder message = new StringBuilder(e.toString());
		StackTraceElement[] stackTrace = e.getStackTrace();
		for (StackTraceElement stackTraceElement : stackTrace) {
			message.append("\n    at
"+stackTraceElement.getClassName()+"."+stackTraceElement.getMethodName());

	message.append("("+stackTraceElement.getFileName()+":"+stackTraceElement.get
LineNumber()+")");
		}
		return message.toString();
	}

	enum ErrorType {
		BLOCKING,
		WARNING
	}

	Cemaden(boolean online) {
		this.online = online;
		fileNameFormatter = new SimpleDateFormat("yyyyMMdd");
		timestampParser = new SimpleDateFormat("dd/MM/yy HH:mm");
		timestampParser.setTimeZone(TimeZone.getTimeZone("GMT"));
		timestampFormatter = new SimpleDateFormat("dd/MM/yy HH:mm");
		floatFormatter = NumberFormat.getInstance(Locale.ENGLISH);
		floatFormatter.setMaximumFractionDigits(2);
	}

	void startReadDataTimer(ReadDataTask task, long interval) {
		if (readDataTimer != null) {
			readDataTimer.cancel();
		}
		readDataTimer = new Timer();
		Date nextDataReleasedDate = getNextDataReleasedDate(new Date());
		readDataTimer.scheduleAtFixedRate(task, nextDataReleasedDate,
interval);
	}
```

```
	/**
	 * <p>
	 * Retorna a data em que (provavelmente) haverá novos dados disponíveis para leitura no Cemaden.
	 * </p>
	 * Os dados são disponibilizados de hora em hora, e quando chove de dez em dez minutos. O método retorna sempre o próximo
	 * sexto de hora, isto é: se agora for 12h, retorna 12h10; se for 12h9, também retorna 12h10. A partir de 12h10, já
	 * retorna 12h20. 12h59 já retorna 13h, e assim por diante.
	 *
	 * @param now agora.
	 * @return próximo sexto de hora.
	 */
	Date getNextDataReleasedDate(Date now) {
		Calendar cal = Calendar.getInstance();
	    cal.setTime(now);
	    int minute = cal.get(Calendar.MINUTE);
	    int ten = minute / 10;
	    cal.set(Calendar.MINUTE, (ten+1)*10);
	    return cal.getTime();
	}

	void setLastInterval(long interval) {
		this.lastInterval = interval;
	}

	/**
	 * Configura o arquivo de log
	 */
	private void logSetup() {
		try {
			logger = Logger.getLogger(Cemaden.class.getName());
			logger.info("Diretório raiz: "+new File(".").getCanonicalPath());
			FileHandler fileHandler = new FileHandler("../logs/cemaden%g.log", 1048576, 100, true);
			fileHandler.setFormatter(new SimpleFormatter());
			logger.addHandler(fileHandler);

		} catch (IOException e) {
			String title = "Erro ao criar arquivo de log";
			String message = "Não foi possível criar o arquivo de log CEMADEN.LOG.\n" +
					"Por favor, verifique se existe espaço em disco e permissão para escrita no diretório.\n\n" +
					"O sistema será DESATIVADO até que o problema seja resolvido.";
			logAndSendError(title, message, ErrorType.BLOCKING);
			System.exit(1);
		}
	}

	/**
	 * Carrega o arquivo de propriedades
	 */
	void loadProperties() {
		try {
			props = new Properties();
```

```
                    props.load(new FileInputStream(new
File("../conf/config.properties")));
             } catch (IOException e) {
                    String title = "Arquivo CONFIG.PROPERTIES não encontrado";
                    String message = "Não foi encontrado o arquivo
CONFIG.PROPERTIES.\n"+
                           "Por favor, verifique se existe um arquivo com esse nome
na pasta onde está o arquivo CEMADEN.JAR.\n\n" +
                           "O sistema será DESATIVADO até que o problema seja
resolvido.";
                    logAndSendError(title, message, ErrorType.BLOCKING);
                    System.exit(1);
             }
      }

      void logAndSendError(final String title, final String message, ErrorType
type, Exception... exception) {
             if (type == ErrorType.BLOCKING) {
                    JOptionPane.showMessageDialog(null, message, title,
JOptionPane.ERROR_MESSAGE);
             }
             else {
                    new Thread() {
                           public void run() {
                                  JOptionPane.showMessageDialog(null, message,
title, JOptionPane.ERROR_MESSAGE);
                           }
                    }.start();
             }
             if (exception == null || exception.length == 0) {
                    logger.severe(message);
             }
             else {
                    logger.log(Level.SEVERE, message, exception[0]);
             }
             sendError(title, message, type, exception);
      }

      void sendError(String title, String message, ErrorType type, Exception...
exception) {
             if (exception != null && exception.length > 0) {
                    message += "\n"+getStackTrace(exception[0]);
             }
             String addressList = getProperty("EMAIL.ERROS");
             if (addressList == null) {
                    title = "Erro ao ler a propriedade EMAIL.ERROS";
                    message += "\n\nNão foi possível ler a propriedade EMAIL.ERROS
no arquivo CONFIG.PROPERTIES.\n" +
                                  "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                                  "Por enquanto, será assumido o valor padrão
ciclano@gmail.com.";
                    addressList = "ciclano@gmail.com";
             }
             sendEmail("Erro do Leitor de Dados do CEMADEN - "+title, message,
addressList);
      }

      /**
```

```
	 * @return intervalo de leitura dos dados do CEMADEN.
	 */
	long getInterval() {
		long interval = 60 * 60 * 1000;
		try {
			interval = Integer.parseInt(getProperty("INTERVALO")) * 60 *
1000;
		} catch (NumberFormatException|NullPointerException e) {
			String title  = "Erro ao ler propriedade INTERVALO";
			String message = "Não foi possível ler a propriedade INTERVALO
no arquivo CONFIG.PROPERTIES.\n" +
					"Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo config.properties.\n" +
					"Por enquanto, será assumido o intervalo padrão de
60 minutos.";
			logAndSendError(title, message, ErrorType.WARNING);
		}
		return interval;
	}

	/**
	 * @param key chave da propriedade desejada.
	 * @return valor da propriedade com a chave especificada EM MAIÚSCULAS.
	 */
	String getProperty(String key) {
		if (props == null) return null;
		String property = props.getProperty(key);
		if (property == null) return null;
		return property.trim();
	}

	 SortedSet<StationData> readData() {
		SortedSet<StationData> data = new TreeSet<>();
		if (online) {
			String url = getURL();
			String tableId = getTableId();
			try {
				WebDriver driver = new FirefoxDriver();
				driver.get(url);
				int TIMEOUT_IN_SECONDS = 30;
				new WebDriverWait(driver,
TIMEOUT_IN_SECONDS).until(ExpectedConditions.presenceOfElementLocated(By.id(tableI
d)));
				WebElement table = driver.findElement(By.id(tableId));
				List<WebElement> names =
table.findElements(By.tagName("td"));
				final int OFFSET_LINE = 13;
				for (int i = 0; i < names.size(); i++) {
					String stationName = names.get(i+2).getText();
					Date timestamp =
getDateTime(names.get(i+3).getText());
					if (timestamp == null || stationName == null ||
stationDeactivated(timestamp)) {
						i += OFFSET_LINE;
						continue;
					}
					Float lastValue =
getFloat(names.get(i+4).getText());
					Float acc1h = getFloat(names.get(i+5).getText());
```

```
                                    Float acc6h = getFloat(names.get(i+6).getText());
                                    Float acc12h = getFloat(names.get(i+7).getText());
                                    Float acc24h = getFloat(names.get(i+8).getText());
                                    Float acc72h = getFloat(names.get(i+9).getText());
                                    Float acc96h =
getFloat(names.get(i+10).getText());
                                    data.add(new StationData(timestamp, stationName,
lastValue, acc1h, acc6h, acc12h, acc24h, acc72h, acc96h));
                                    i += OFFSET_LINE;
                              }
                              driver.quit();
                        } catch (TimeoutException e) {
                              String title = "Erro na leitura dos dados do CEMADEN";
                              String message = "Não foram encontrados os dados do
CEMADEN (tabela \""+tableId+"\") no endereço:\n" +
                                          url + "\n"+
                                          "Por favor, verifique se a conexão com a
Internet está ativa, e se o endereço e o identificador para os dados estão
corretos.";
                              logAndSendError(title, message, ErrorType.WARNING, e);
                        }
                  }
                  else {
                        // Dados de histórico
//                      stations.add(new String[]{"Charitas","06/08/2016 01:00","0","-
","-","45","-","-","-"});
//                      stations.add(new String[] {"Piratininga 1","06/08/2016
01:00","0","-","-","0.2","-","-","-"});
//                      stations.add(new String[] {"Santa Barbara","06/08/2016
01:00","0","-","-","-","-","0.5","-"});
//                      stations.add(new String[]{"Charitas","06/08/2016 02:00","0","-
","-","45","-","-","-"});
//                      stations.add(new String[] {"Piratininga 1","06/08/2016
02:00","3","-","-","0.2","-","-","-"});
//                      stations.add(new String[] {"Santa Barbara","06/08/2016
02:00","4","-","-","-","-","0.5","-"});
//                      stations.add(new String[]{"Charitas","06/08/2016 02:00","0","-
","-","45","-","-","-"});
//                      stations.add(new String[] {"Piratininga 1","06/08/2016
02:10","3","-","-","0.2","-","-","-"});
//                      stations.add(new String[] {"Santa Barbara","06/08/2016
02:10","4","-","-","-","-","0.5","-"});
//                      stations.add(new String[]{"Charitas","06/08/2016 02:00","0","-
","-","45","-","-","-"});
//                      stations.add(new String[] {"Piratininga 1","06/08/2016
02:20","3","-","-","0.2","-","-","-"});
//                      stations.add(new String[] {"Santa Barbara","06/08/2016
02:20","4","-","-","-","-","0.5","-"});
//                      stations.add(new String[]{"Charitas","06/08/2016 02:00","0","-
","-","45","-","-","-"});
//                      stations.add(new String[] {"Piratininga 1","06/08/2016
02:30","2","-","-","0.2","-","-","-"});
//                      stations.add(new String[] {"Santa Barbara","06/08/2016
02:30","1","-","-","-","-","0.5","-"});
//                      stations.add(new String[]{"Charitas","06/08/2016 02:00","0","-
","-","45","-","-","-"});
//                      stations.add(new String[] {"Piratininga 1","06/08/2016
02:40","0","-","-","0.2","-","-","-"});
```

```
//                  stations.add(new String[] {"Santa Barbara","06/08/2016
02:40","0","-","-","-","-","0.5","-"});
//                  stations.add(new String[]{"Charitas","06/08/2016 03:00","0","-
","-","45","-","-","-"});
//                  stations.add(new String[] {"Piratininga 1","06/08/2016
03:00","0","-","-","0.2","-","-","-"});
//                  stations.add(new String[] {"Santa Barbara","06/08/2016
03:00","0","-","-","-","-","0.5","-"});
//                  stations.add(new String[]{"Charitas","06/08/2016 04:00","0","-
","-","45","-","-","-"});
//                  stations.add(new String[] {"Piratininga 1","06/08/2016
04:00","0","-","-","0.2","-","-","-"});
//                  stations.add(new String[] {"Santa Barbara","06/08/2016
04:00","0","-","-","-","-","0.5","-"});
//                  stations.add(new String[]{"Charitas","06/08/2016 05:00","0","-
","-","45","-","-","-"});
//                  stations.add(new String[] {"Piratininga 1","06/08/2016
05:00","0","-","-","0.2","-","-","-"});
//                  stations.add(new String[] {"Santa Barbara","06/08/2016
05:00","0","-","-","-","-","0.5","-"});
//                  stations.add(new String[]{"Charitas","06/08/2016 06:00","0","-
","-","45","-","-","-"});
//                  stations.add(new String[] {"Piratininga 1","06/08/2016
06:00","0","-","-","0.2","-","-","-"});
//                  stations.add(new String[] {"Santa Barbara","06/08/2016
06:00","0","-","-","-","-","0.5","-"});
//                  this.getCityStations().recordData(stations);
//                  stations.clear();
//
//                  stations.add(new String[]{"Charitas","06/08/2016 02:50","0","-
","-","45","-","-","-"});
//                  stations.add(new String[] {"Piratininga 1","06/08/2016
02:50","0","-","-","0.2","-","-","-"});
//                  stations.add(new String[] {"Santa Barbara","06/08/2016
02:50","0","-","-","-","-","0.5","-"});
            }
            return data;
      }

      private boolean stationDeactivated(Date lastReadTime) {
            if (lastReadTime == null) return true;
            Date limitDateTime = addHours(new Date(), -24);
            return lastReadTime.before(limitDateTime);
      }

      /**
       * @return endereço para recuperação dos dados do Cemaden.
       */
      private String getURL() {
            String url = getProperty("URL");
            if (url == null || url.isEmpty()) {
                  url =
"http://150.163.255.234/salvar/mapa_interativo/interativo/grafico_CEMADEN.php?idpc
d=3727&uf=RJ";
                  String title = "Erro ao ler propriedade URL";
                  String message = "Não foi possível ler a propriedade URL no
arquivo CONFIG.PROPERTIES.\n" +
```

```
                                "Por favor, verifique se existe uma propriedade
com esse nome e o caminho para os dados do CEMADEN no arquivo
CONFIG.PROPERTIES.\n" +
                                "Por enquanto, será assumido o endereço
original.";
                        logAndSendError(title, message, ErrorType.WARNING);
                }
                return url;
        }

         /**
          * @return identificador da tabela no HTML da página de dados do Cemaden.
          */
         private String getTableId() {
                String tableId = getProperty("ID.TABELA");
                if (tableId == null || tableId.isEmpty()) {
                        String title = "Erro ao ler propriedade ID.TABELA";
                        String message = "Não foi possível ler a propriedade ID.TABELA
no arquivo CONFIG.PROPERTIES.\n" +
                                "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n" +
                                "Por enquanto, será assumido o valor original.";
                        logAndSendError(title, message, ErrorType.WARNING);
                        tableId = "infopcds";
                }
                return tableId;
        }

         /**
          * Registra os dados lidos em um arquivo.
          * @param stations dados lidos das estações meteorológicas.
          */
        void saveData(SortedSet<StationData> newData) {
                Date now = new Date();
                String date = fileNameFormatter.format(now);
                String fileName = "dados-"+date+".csv";
                String firstTimestamp = getFormattedTimestamp(getFirstTimestamp());
                BufferedWriter out = null;
                try {
                        out = new BufferedWriter(new FileWriter(new
File("../dados/"+fileName), true));

        out.write("CEMADEN,,,,,,,,,ACUMULADOS,("+firstTimestamp+"),,,,,,,,,\n");
                        out.write("Data/Hora,Estação,Último,1h,6h,12h,24h,72h,96h,,"+

        "Data/Hora,Estação,Último,1h,6h,12h,24h,48h,72h,96h,mês\n");
                        for (StationData data : newData) {
                                Date timestamp = data.getTimestamp();
                                String formattedTimestamp =
getFormattedTimestamp(timestamp);
                                String stationName = data.getStationName();
                                String cemLastStr =
getFormattedFloat(data.getCemLast());
                                String cemAcc1hStr =
getFormattedFloat(data.getCemAcc1h());
                                String cemAcc6hStr =
getFormattedFloat(data.getCemAcc6h());
                                String cemAcc12hStr =
getFormattedFloat(data.getCemAcc12h());
```

```
                        String cemAcc24hStr =
getFormattedFloat(data.getCemAcc24h());
                        String cemAcc72hStr =
getFormattedFloat(data.getCemAcc72h());
                        String cemAcc96hStr =
getFormattedFloat(data.getCemAcc96h());

      out.write(formattedTimestamp+","+stationName+","+cemLastStr+","+cemAcc1hStr+
","+cemAcc6hStr+","+

      cemAcc12hStr+","+cemAcc24hStr+","+cemAcc72hStr+","+cemAcc96hStr+",,");
                        String myLastStr = getFormattedFloat(data.getMyLast());
                        String myAcc1hStr =
getFormattedFloat(data.getMyAcc1h());
                        String myAcc6hStr =
getFormattedFloat(data.getMyAcc6h());
                        String myAcc12hStr =
getFormattedFloat(data.getMyAcc12h());
                        String myAcc24hStr =
getFormattedFloat(data.getMyAcc24h());
                        String myAcc48hStr =
getFormattedFloat(data.getMyAcc48h());
                        String myAcc72hStr =
getFormattedFloat(data.getMyAcc72h());
                        String myAcc96hStr =
getFormattedFloat(data.getMyAcc96h());
                        String myAccMonthStr =
getFormattedFloat(data.getMyAccMonth());

      out.write(formattedTimestamp+","+stationName+","+myLastStr+","+myAcc1hStr+",
"+myAcc6hStr+","+myAcc12hStr+","+

      myAcc24hStr+","+myAcc48hStr+","+myAcc72hStr+","+myAcc96hStr+","+myAccMonthSt
r+"\n");
                  }
                  out.newLine();
            } catch (IOException e) {
                  String title = "Erro ao gravar o arquivo "+fileName;
                  String message = "Não foi possível escrever no arquivo
"+fileName+".\n" +
                              "Se algum programa estiver com esse arquivo
aberto, por favor feche.\n" +
                              "Outra possibilidade é não ter mais espaço em
disco para gravá-lo.";
                  logAndSendError(title, message, ErrorType.WARNING);
            } finally {
                  try {
                        if (out != null) out.close();
                  } catch (IOException e) {
                        String title = "Erro ao fechar o arquivo "+fileName;
                        String message = "Não foi possível fechar o arquivo
"+fileName+".\n" +
                                    "Se algum programa estiver com esse arquivo
aberto, por favor feche.\n" +
                                    "Verifique posteriormente se o arquivo foi
gravado corretamente.";
                        logAndSendError(title, message, ErrorType.WARNING);
                  }
            }
```

```
	}

	private String getFormattedFloat(Float number) {
		if (number == null) {
			return "-";
		}
		return floatFormatter.format(number);
	}

	public Date addHours(Date date, int hours) {
	    Calendar cal = Calendar.getInstance();
	    cal.setTime(date);
	    cal.add(Calendar.HOUR_OF_DAY, hours);
	    return cal.getTime();
	}

	public Date addMonths(Date date, int months) {
		Calendar cal = Calendar.getInstance();
		cal.setTime(date);
		cal.add(Calendar.MONTH, months);
		return cal.getTime();
	}

	private Date getDateTime(String dateStr) {
		if (dateStr == null || dateStr.trim().isEmpty() || dateStr.equals("-
")) {
			return null;
		}
		try {
			return timestampParser.parse(dateStr);
		} catch (ParseException e) {
			Cemaden.this.logAndSendError("Erro na conversão de data", "Não
foi possível converter a data/hora "+dateStr, ErrorType.WARNING);
		}
		return null;
	}

	private String getFormattedTimestamp(Date timestamp) {
		if (timestamp == null) return "-";
		return timestampFormatter.format(timestamp);
	}

	private Float getFloat(String value) {
		value = value.trim();
		if (value.isEmpty() || value.equals("-")) {
			return null;
		}
		try {
			return Float.parseFloat(value);
		} catch (NumberFormatException e) {
			// Não há necessidade de logar erro neste caso
			return null;
		}
	}

	Date getFirstTimestamp() {
		if (accumulatedData == null) return null;
		StationData firstData = accumulatedData.first();
```

```
            return firstData.getTimestamp();
      }

      /**
       * Recupera o volume acumulado em determinada estação, até X horas antes da
hora especificada.
       *
       * @param timestamp horário atual.
       * @param stationName nome da estação.
       * @param hours horas para trás (1h, 6h, 24h, 48h, 72h ou 96h).
       *
       * @return volume acumulado na estação.
       */
      public Float getHoursAccumulated(Date timestamp, String stationName, int
hours) {
            StationData from = new StationData(addHours(timestamp, -hours),
stationName);
            return getAccumulated(timestamp, stationName, from);
      }

      /**
       * Recupera o volume acumulado em determinada estação, exatamente um mês
antes da hora especificada.
       *
       * @param timestamp horário atual.
       * @param stationName nome da estação.
       *
       * @return volume acumulado na estação em um mês.
       */
      public Float getMonthAccumulated(Date timestamp, String stationName) {
            StationData from = new StationData(addMonths(timestamp, -1),
stationName);
            return getAccumulated(timestamp, stationName, from);
      }

      /**
       * Retorna o volume de chuva acumulado em uma determinada estação,
       *
       * @param timestamp
       * @param stationName
       * @param from
       * @return
       */
      private Float getAccumulated(Date timestamp, String stationName, StationData
from) {
            StationData to = new StationData(timestamp, stationName);
            SortedSet<StationData> subData = accumulatedData.subSet(from, false,
to, true);
            if (subData.isEmpty()) return null;
            Float accumulated = 0f;
            for (StationData stationData : subData) {
                  if (stationData.getStationName().equals(stationName)) {
                        accumulated += stationData.getCemLast();
                  }
            }
            return accumulated;
      }

      List<String> processAlarms(SortedSet<StationData> stations) {
```

```
		List<String> alarms = new ArrayList<String>();
		for (StationData data : stations) {
			String alarm = processLastData(data);
			if (alarm != null) alarms.add(alarm);
			alarm = processAcc1hData(data);
			if (alarm != null) alarms.add(alarm);
			alarm = processOtherAccumulatedData(data, 24, "24H");
			if (alarm != null) alarms.add(alarm);
			alarm = processOtherAccumulatedData(data, 48, "48H");
			if (alarm != null) alarms.add(alarm);
			alarm = processOtherAccumulatedData(data, 96, "96H");
			if (alarm != null) alarms.add(alarm);
			alarm = processAccumulatedMonth(data);
			if (alarm != null) alarms.add(alarm);
		}
		return alarms;
	}

	private String processLastData(StationData data) {
		Float volume = data.getMyLast();
		if (volume == null) return null;
		float alertLimit = 0;
		try {
			alertLimit =
Float.parseFloat((String)getProperty("1H.SEM.ACUMULADO"));
		} catch (NumberFormatException|NullPointerException e) {
			String title = "Erro ao ler propriedade 1H.SEM.ACUMULADO";
			String message = "Não foi possível ler a propriedade
1H.SEM.ACUMULADO no arquivo CONFIG.PROPERTIES.\n" +
					"Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
					"Este alerta está DESATIVADO até que o problema
seja resolvido.";
			logAndSendError(title, message, ErrorType.WARNING);
			return null;
		}
		if (volume >= alertLimit) {
			return "Alarme "+data.getStationName()+": "+volume+"mm sem
acumulado.";
		}
		return null;
	}

	private String processOtherAccumulatedData(StationData data, int hours,
String alertLimitProperty) {
		Date timestamp = data.getTimestamp();
		String stationName = data.getStationName();
		if (timestamp == null || stationName == null) return null;
		Float accVolume = null;
		switch (hours) {
		case 24:
			accVolume = data.getMyAcc24h();
			break;
		case 48:
			accVolume = data.getMyAcc48h();
			break;
		case 96:
			accVolume = data.getMyAcc96h();
			break;
```

```
            }
            if (accVolume == null) { // Verifica se existe dado do Cemaden
                  switch (hours) {
                  case 24:
                        accVolume = data.getCemAcc24h();
                        break;
                  case 96:
                        accVolume = data.getCemAcc96h();
                        break;
                  }
                  if (accVolume == null) return null;
            }
            float alertLimit = 0;
            try {
                  alertLimit =
Float.parseFloat((String)getProperty(alertLimitProperty));
            } catch (NumberFormatException|NullPointerException e) {
                  String title = "Erro ao ler propriedade "+alertLimitProperty;
                  String message = "Não foi possível ler a propriedade "+
alertLimitProperty +" no arquivo CONFIG.PROPERTIES.\n" +
                              "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                              "Este alerta está DESATIVADO até que o problema
seja resolvido.";
                  logAndSendError(title, message, ErrorType.WARNING);
                  return null;
            }
            if (accVolume >= alertLimit) {
                  return "Alarme "+data.getStationName()+": " + accVolume + "mm
em " + hours + "horas.";
            }
            return null;
      }

      private String processAcc1hData(StationData data) {
            Float volume = data.getMyLast();
            if (volume == null) return null;

            Date timestamp = data.getTimestamp();
            String stationName = data.getStationName();
            if (timestamp == null || stationName == null) return null;
            Float myAcc1h = getHoursAccumulated(timestamp, stationName, 1);
            if (myAcc1h == null) return null;

            float alertLimit = 0;
            try {
                  alertLimit =
Float.parseFloat((String)getProperty("1H.COM.ACUMULADO"));
            } catch (NumberFormatException|NullPointerException e) {
                  String title = "Erro ao ler propriedade 1H.COM.ACUMULADO";
                  String message = "Não foi possível ler a propriedade
1H.COM.ACUMULADO no arquivo CONFIG.PROPERTIES.\n" +
                              "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                              "Este alerta está DESATIVADO até que o problema
seja resolvido.";
                  logAndSendError(title, message, ErrorType.WARNING);
                  return null;
            }
```

```
            if (myAcc1h > 0 && volume >= alertLimit) {
                  return "Alarme "+data.getStationName()+": "+volume+"mm em 1h
com acumulado.";
            }
            return null;
      }

      private String processAccumulatedMonth(StationData data) {
            Date timestamp = data.getTimestamp();
            String stationName = data.getStationName();
            if (timestamp == null || stationName == null) return null;
            Float myAccMonth = getMonthAccumulated(timestamp, stationName);
            if (myAccMonth == null) return null;

            float alertLimit = 0;
            try {
                  alertLimit = Float.parseFloat((String)getProperty("MENSAL"));
            } catch (NumberFormatException|NullPointerException e) {
                  String title = "Erro ao ler propriedade MENSAL";
                  String message = "Não foi possível ler a propriedade MENSAL no
arquivo CONFIG.PROPERTIES.\n" +
                              "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                              "Este alerta está DESATIVADO até que o problema
seja resolvido.";
                  logAndSendError(title, message, ErrorType.WARNING);
                  return null;
            }
            if (myAccMonth >= alertLimit) {
                  return "Alarme "+data.getStationName() +": "+myAccMonth+"mm em
um mês.";
            }
            return null;
      }

       void sendAlarm(List<String> alarms) {
             if (alarms.isEmpty()) {
                   return;
             }
             StringBuilder text = new StringBuilder();
             for (String alarme : alarms) {
                   text.append(alarme);
             }
             String addressList = getProperty("EMAIL.AVISOS");
             if (addressList == null) {
                  String title = "Erro ao ler a propriedade EMAIL.AVISOS";
                  String message = "Não foi possível ler a propriedade
EMAIL.AVISOS no arquivo CONFIG.PROPERTIES.\n" +
                              "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                              "Os alarmes NÃO SERÃO NOTIFICADOS até que o
problema seja resolvido.";
                  logAndSendError(title, message, ErrorType.WARNING);
             }
             sendEmail("Alarme de chuva", text.toString(), addressList);
      }

      void sendEmail(String subject, String text, String targetList) {
            Properties props = new Properties();
```

```
			props.put("mail.smtp.host", "smtp.gmail.com");
			props.put("mail.smtp.socketFactory.port", "465");
			props.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
			props.put("mail.smtp.auth", "true");
			props.put("mail.smtp.port", "465");
			Session session = Session.getDefaultInstance(props, new
Authenticator() {
				protected PasswordAuthentication getPasswordAuthentication() {
					//TODO Encriptar e O2Auth
					return new PasswordAuthentication();
				}
			});
			session.setDebug(true);
			try {
				Message message = new MimeMessage(session);
				message.setFrom(new InternetAddress("niteroidc@gmail.com"));
				Address[] toUser = InternetAddress.parse(targetList);
				message.setRecipients(Message.RecipientType.TO, toUser);
				message.setSubject(subject);
				message.setText(text);
				Transport.send(message);
				logger.fine("e-mail enviado.");
			}
			catch (MessagingException e) {
				String title = "Erro no envio de e-mail";
			   String message = "Não foi possível enviar e-mail. Verifique se
a conexão com a internet está ativa.";
				logAndSendError(title , message , ErrorType.WARNING, e);
			}
		}

		public long getLastInterval() {
			return lastInterval;
		}

		public void updateLevelLog() {
			String level = getProperty("LOG");
			if (level == null) {
				String title = "Erro na recuperação da propriedade LOG";
				String message = "Não foi possível encontrar a propriedade LOG
no arquivo CONFIG.PROPERTIES.\n"+
					"Por favor, verifique se existe uma propriedade com esse
nome e um valor no arquivo CONFIG.PROPERTIES.\n" +
					"Por enquanto, será assumido o valor DEBUG.";
				logAndSendError(title, message, ErrorType.WARNING);
				level = "DEBUG";
			}
			level = level.toUpperCase();
			if (!level.equals("NORMAL") && !level.equals("DEBUG")) {
				String title = "Erro na leitura da propriedade LOG";
				String message = "Não foi reconhecido o valor da propriedade
LOG no arquivo CONFIG.PROPERTIES.\n"+
					"Para essa propriedade, só são admitidos os
valores NORMAL e DEBUG.\n"+
					"Por favor, verifique o valor da propriedade LOG
no arquivo CONFIG.PROPERTIES.\n" +
					"Por enquanto, será assumido o valor DEBUG.";
				logAndSendError(title, message, ErrorType.WARNING);
```

```
                level = "DEBUG";
            }
            if (level.equals("NORMAL")) {
                logger.setLevel(Level.INFO);
            }
            else {
                logger.setLevel(Level.ALL);
            }
        }

        /**
         * Acumula os dados lidos, isto é, soma os valores acumulados com os recém-
lidos.
         *
         * @param newData dados recém-lidos.
         */
        /**
         * Verifica se os dados lidos são novos, isto é, se já haviam sido lidos
antes ou não. Retorna somente, dentre o conjunto
         * de dados recebidos, aqueles efetivamente novos.
         *
         * @param readData conjunto de dados lidos na última leitura.
         * @return
         * @return subconjunto dos dados novos.
         */
        public SortedSet<StationData> accumulateData(SortedSet<StationData>
readData) {
            SortedSet<StationData> newData;
            if (accumulatedData == null) {
                accumulatedData = new TreeSet<>();
                accumulatedData.addAll(readData);
                newData = readData;
            }
            else {
                newData = new TreeSet<>();
                for (StationData datum : readData) {
                    if (!accumulatedData.contains(datum)) {
                        accumulatedData.add(datum);
                        newData.add(datum);
                    }
                }
            }
                // TODO testar esta limpeza de lixo
//                  SortedSet<StationData> oldData =
//                      stationDataSet.headSet(new
StationData(addMonths(timestamp, -1), stationName), true);
//                  stationDataSet.removeAll(oldData);
            for (StationData data : newData) {
                Date timestamp = data.getTimestamp();
                String stationName = data.getStationName();
                Float myAcc1h = getHoursAccumulated(timestamp, stationName, 1);
                myAcc1h = getAdjustedMyAcc1h(data, myAcc1h);
                Float myAcc6h = getHoursAccumulated(timestamp, stationName, 6);
                Float myAcc12h = getHoursAccumulated(timestamp, stationName,
12);
                Float myAcc24h = getHoursAccumulated(timestamp, stationName,
24);
                Float myAcc48h = getHoursAccumulated(timestamp, stationName,
48);
```

```
                        Float myAcc72h = getHoursAccumulated(timestamp, stationName,
72);
                        Float myAcc96h = getHoursAccumulated(timestamp, stationName,
96);
                        Float myAccMonth = getMonthAccumulated(timestamp, stationName);
                        data.set(myAcc1h, myAcc6h, myAcc12h, myAcc24h, myAcc48h,
myAcc72h, myAcc96h, myAccMonth);
                }
                return newData;
        }

        /**
         * Ajuste no cálculo do acumulado (e possivelmente no valor lido por
último), por conta de eventuais falhas no repasse
         * de informações pelo Cemaden. Podem ocorrer situações em que o acumulado
de 1h reflete dados que não foram exibidos
         * no campo "Último", e portanto devem ser ajustados. O algoritmo deste
método faz o seguinte:
         * <ul>
         * <li>Se o acumulado de 1h do cemaden for maior que o meu acumulado:</li>
         * <li>Ultimo += cemAcc1h - Ultimo</li>
         * <li>Reprocessa os acumulados</li>
         * </ul>
         * @param data dados lidos pelo Cemaden
         * @param myAcc1h acumulado calculado.
         * @return acumulado ajustado (o dado "Ultimo" também pode ser atualizado).
         */
        Float getAdjustedMyAcc1h(StationData data, Float myAcc1h) {
                if (data.getCemAcc1h() == null || data.getCemAcc1h() == myAcc1h) {
                        return myAcc1h;
                }
                else if (data.getCemAcc1h() > myAcc1h) {
                        float missingLast = data.getCemAcc1h() - myAcc1h;
                        data.setMyLast(data.getCemLast() + missingLast);
                        return data.getCemAcc1h();
                }
                else {
                        float exceedingLast = myAcc1h - data.getCemAcc1h();
                        data.setMyLast(data.getCemLast() - exceedingLast);
                        return data.getCemAcc1h();
                }
        }

}

class ReadDataTask extends TimerTask {

        private Cemaden cemaden;

        public ReadDataTask(Cemaden c) {
                this.cemaden = c;
        }

        @Override
        public void run() {
                try {
                        cemaden.logger.info("Iniciando a leitura de dados do
CEMADEN...");
                        cemaden.loadProperties();
```

```
                    cemaden.updateLevelLog();
                    cemaden.logger.fine("Vai começar a ler as estações...");
                    SortedSet<StationData> validData = cemaden.readData();
                    cemaden.logger.fine("Leu. Vai acumular...");
                    SortedSet<StationData> newData =
cemaden.accumulateData(validData);
                    cemaden.logger.fine("Acumulou. Vai salvar...");
                    if (!newData.isEmpty()) {
                          cemaden.saveData(validData);
                          cemaden.logger.fine("Salvou. Vai processar alarmes...");
                          List<String> alarms = cemaden.processAlarms(newData);
                          cemaden.logger.fine("Processados. Vai enviar
alarmes...");
                          cemaden.sendAlarm(alarms);
                          cemaden.logger.fine("Alarmes enviados.");
                    }
                    long interval = cemaden.getInterval();
                    if (cemaden.getLastInterval() != interval) {
                          cemaden.logger.fine("Intervalo modificado de
"+cemaden.getLastInterval()+" para "+interval+". Reiniciando o timer...");
                          cemaden.setLastInterval(interval);
                          cemaden.startReadDataTimer(this, interval);
                    }
                    cemaden.logger.info("Concluída a leitura e processamento dos
dados.");
              } catch (Exception e) {
                    cemaden.logAndSendError("Erro ao executar a ReadDataTask",
"Erro inesperado", ErrorType.WARNING, e);
              }
        }
}

class PingTask extends TimerTask {

        private Cemaden cemaden;

        public PingTask(Cemaden c) {
              this.cemaden = c;
        }

        @Override
        public void run() {
              String title = "Serviço ativo no ambiente ";
              String message = "Esta mensagem só confirma que o serviço está de pé,
mas não verifica se houve erros.";
              String environment = cemaden.getProperty("AMBIENTE");
              if (environment == null) {
                    environment = "INDEFINIDO";
                    message += "\n\nNão foi possível ler a propriedade EMAIL.ERROS
no arquivo CONFIG.PROPERTIES.\n" +
                                "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                                "Por enquanto, será assumido o valor padrão
ciclano@gmail.com.";
              }
              title += environment;
              String addressList = cemaden.getProperty("EMAIL.ERROS");
              if (addressList == null) {
```

```
                    message += "\n\nNão foi possível ler a propriedade EMAIL.ERROS
no arquivo CONFIG.PROPERTIES.\n" +
                                "Por favor, verifique se existe uma propriedade
com esse nome e um valor no arquivo CONFIG.PROPERTIES.\n\n"+
                                "Por enquanto, será assumido o valor padrão
ciclano@gmail.com.";
                    addressList = "ciclano@gmail.com";
            }
            cemaden.sendEmail(title, message, addressList);
      }
}
```

Classe StationData

```
package br.com.adache;

import java.util.Date;

class StationData implements Comparable<StationData>{
	private Date timestamp;
	private String stationName;
	private Float cemLast, cemAcc1h, cemAcc6h, cemAcc12h, cemAcc24h, cemAcc72h,
cemAcc96h;
	private Float myLast, myAcc1h, myAcc6h, myAcc12h, myAcc24h, myAcc48h,
myAcc72h, myAcc96h, myAccMonth;

	StationData(Date timestamp, String stationName, Float cemLast) {
		this.timestamp = timestamp;
		this.stationName = stationName;
		this.cemLast = cemLast;
		this.myLast = cemLast;
	}

	public StationData(Date timestamp, String stationName, Float cemLast, Float
cemAcc1h, Float cemAcc6h, Float cemAcc12h,
			Float cemAcc24h, Float cemAcc72h, Float cemAcc96h) {
		this(timestamp, stationName, cemLast);
		this.cemAcc1h = cemAcc1h;
		this.cemAcc6h = cemAcc6h;
		this.cemAcc12h = cemAcc12h;
		this.cemAcc24h = cemAcc24h;
		this.cemAcc72h = cemAcc72h;
		this.cemAcc96h = cemAcc96h;
	}

	StationData(Date timestamp, String stationName) {
		this(timestamp, stationName, 0f);
	}

	public void set(Float myAcc1h, Float myAcc6h, Float myAcc12h, Float
myAcc24h, Float myAcc48h, Float myAcc72h,
			Float myAcc96h, Float myAccMonth) {
		this.myAcc1h = myAcc1h;
		this.myAcc6h = myAcc6h;
		this.myAcc12h = myAcc12h;
		this.myAcc24h = myAcc24h;
		this.myAcc48h = myAcc48h;
		this.myAcc72h = myAcc72h;
		this.myAcc96h = myAcc96h;
		this.myAccMonth = myAccMonth;
	}

	@Override
	public int compareTo(StationData other) {
		if (!this.timestamp.equals(other.timestamp)) {
			return this.timestamp.compareTo(other.timestamp);
		}
		return this.stationName.compareTo(other.stationName);
	}

	@Override
```

```
	public String toString() {
		return "StationData [stationName=" + stationName + ", timestamp=" +
timestamp + ", value=" + cemLast + "]";
	}

	public String getStationName() {
		return stationName;
	}

	public Date getTimestamp() {
		return timestamp;
	}

	public Float getCemLast() {
		return cemLast;
	}

	public Float getCemAcc1h() {
		return cemAcc1h;
	}

	public Float getCemAcc6h() {
		return cemAcc6h;
	}

	public Float getCemAcc12h() {
		return cemAcc12h;
	}

	public Float getCemAcc24h() {
		return cemAcc24h;
	}

	public Float getCemAcc72h() {
		return cemAcc72h;
	}

	public Float getCemAcc96h() {
		return cemAcc96h;
	}

	public Float getMyLast() {
		return myLast;
	}

	public Float getMyAcc1h() {
		return myAcc1h;
	}

	public Float getMyAcc6h() {
		return myAcc6h;
	}

	public Float getMyAcc12h() {
		return myAcc12h;
	}

	public Float getMyAcc24h() {
		return myAcc24h;
```

```
	}

	public Float getMyAcc48h() {
		return myAcc48h;
	}

	public Float getMyAcc72h() {
		return myAcc72h;
	}

	public Float getMyAcc96h() {
		return myAcc96h;
	}

	public Float getMyAccMonth() {
		return myAccMonth;
	}

	public void setMyLast(float myLast) {
		this.myLast = myLast;
	}

}
```

Classe CemadenTest

```
package br.com.adache;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNull;
import static org.junit.Assert.assertTrue;

import java.io.IOException;
import java.util.Iterator;
import java.util.SortedSet;
import java.util.TreeSet;

import org.junit.Before;
import org.junit.Test;

public class CemadenTest {

	private Cemaden c;

	@Before
	public void setup() throws IOException {
		c = new Cemaden(false);
	}

	@Test
	public void testGetNextReleasedDate() {
		assertEquals(Util.date(2016,8,1,0,10),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 0)));
		assertEquals(Util.date(2016,8,1,0,10),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 1)));
		assertEquals(Util.date(2016,8,1,0,10),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 9)));
		assertEquals(Util.date(2016,8,1,0,20),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 10)));
		assertEquals(Util.date(2016,8,1,0,20),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 11)));
		assertEquals(Util.date(2016,8,1,0,20),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 19)));
		assertEquals(Util.date(2016,8,1,0,30),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 20)));
		assertEquals(Util.date(2016,8,1,0,30),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 29)));
		assertEquals(Util.date(2016,8,1,0,40),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 39)));
		assertEquals(Util.date(2016,8,1,0,50),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 49)));
		assertEquals(Util.date(2016,8,1,1,0),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 0, 50)));
		assertEquals(Util.date(2016,8,2,0,0),
c.getNextDataReleasedDate(Util.date(2016, 8, 1, 23, 59)));
		assertEquals(Util.date(2016,9,1,0,0),
c.getNextDataReleasedDate(Util.date(2016, 8, 31, 23, 59)));
		assertEquals(Util.date(2017,1,1,0,0),
c.getNextDataReleasedDate(Util.date(2016, 12, 31, 23, 59)));
		assertEquals(Util.date(2016,12,31,15,0),
c.getNextDataReleasedDate(Util.date(2016, 12, 31, 14, 54)));
	}
```

```
	@Test
	public void testGetHoursAccumulated() {
		SortedSet<StationData> readData = new TreeSet<>();
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 0), "b", 1f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 10), "b", 2f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 20), "b", 2f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 30), "b", 3f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 40), "b", 3f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 50), "b", 4f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 5, 2, 0), "b", 3f));
		readData.add(new StationData(Util.date(2016, 8, 5, 7, 0), "b", 5f));
		c.accumulateData(readData);
		assertNull(c.getHoursAccumulated(Util.date(2016, 8, 5, 0, 0), "b",
1));
		assertEquals((Float)1f, c.getHoursAccumulated(Util.date(2016, 8, 5,
1, 0), "b", 1));
		assertEquals((Float)8f, c.getHoursAccumulated(Util.date(2016, 8, 5,
1, 30), "b", 1));
		assertEquals((Float)17f, c.getHoursAccumulated(Util.date(2016, 8, 5,
2, 0), "b", 1));
		assertEquals((Float)10f, c.getHoursAccumulated(Util.date(2016, 8, 5,
2, 30), "b", 1));
		assertEquals((Float)22f, c.getHoursAccumulated(Util.date(2016, 8, 5,
7, 0), "b", 6));
	}

	@Test
	public void testGetMonthAccumulated() {
		SortedSet<StationData> readData = new TreeSet<>();
		readData.add(new StationData(Util.date(2016, 8, 5, 1, 0), "b", 1f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 15, 1, 10), "b",
2f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 25, 1, 20), "b",
2f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 8, 30, 1, 30), "b",
3f));
		c.accumulateData(readData);

		readData.add(new StationData(Util.date(2016, 8, 30, 7, 10), "b",
3f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 9, 1, 0, 0), "b", 5f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 9, 5, 1, 40), "b", 3f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 9, 15, 1, 50), "b",
4f));
		c.accumulateData(readData);
		readData.add(new StationData(Util.date(2016, 9, 25, 2, 0), "b", 3f));
		readData.add(new StationData(Util.date(2016, 9, 30, 7, 0), "b", 5f));
```

```
            c.accumulateData(readData);
            assertEquals((Float)23f, c.getMonthAccumulated(Util.date(2016, 9, 30,
7, 0), "b"));
      }

      @Test
      public void testCheckForNewData() {
            SortedSet<StationData> readData = new TreeSet<>();
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "a", 1f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "b", 2f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "c", 3f));
            SortedSet<StationData> newData = c.accumulateData(readData);
            assertEquals(readData, newData);

            readData.clear();
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "a", 1f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "b", 2f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "c", 3f));
            newData = c.accumulateData(readData);
            assertTrue(newData.isEmpty());

            readData.clear();
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "a", 1f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 1), "b", 2f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 2), "c", 4f));
            newData = c.accumulateData(readData);
            SortedSet<StationData> expected = new TreeSet<>();
            expected.add(new StationData(Util.date(2016, 8, 1, 1, 1), "b"));
            expected.add(new StationData(Util.date(2016, 8, 1, 1, 2), "c"));
            assertEquals(expected, newData);
      }

      @Test
      public void accumulateData() {
            SortedSet<StationData> readData = new TreeSet<>();
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "x", 1f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "y", 2f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "z", 3f));
            SortedSet<StationData> expected = new TreeSet<>();
            expected.addAll(readData);
            SortedSet<StationData> newData = c.accumulateData(readData);
            assertEquals(expected, newData);
            Iterator<StationData> iterator = newData.iterator();
            StationData x = iterator.next();
            assertEquals((Float)1f, x.getMyLast());
            assertEquals((Float)1f, x.getMyAcc1h());
            StationData y = iterator.next();
            assertEquals((Float)2f, y.getMyLast());
            assertEquals((Float)2f, y.getMyAcc1h());
            StationData z = iterator.next();
            assertEquals((Float)3f, z.getMyLast());
            assertEquals((Float)3f, z.getMyAcc1h());

            readData.clear();
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "x", 4f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "y", 3f));
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "z", 2f));
            newData = c.accumulateData(readData);
            assertTrue(newData.isEmpty());
```

```
            readData.clear();
            readData.add(new StationData(Util.date(2016, 8, 1, 1, 0), "x", 1f));
            readData.add(new StationData(Util.date(2016, 8, 1, 2, 10), "y", 2f));
            readData.add(new StationData(Util.date(2016, 8, 1, 2, 10), "z", 4f));
            newData = c.accumulateData(readData);
            expected.clear();
            expected.add(new StationData(Util.date(2016, 8, 1, 2, 10), "y"));
            expected.add(new StationData(Util.date(2016, 8, 1, 2, 10), "z"));
            assertEquals(expected, newData);
            iterator = newData.iterator();
            y = iterator.next();
            assertEquals((Float)2f, y.getMyLast());
            assertEquals((Float)2f, y.getMyAcc1h());
            z = iterator.next();
            assertEquals((Float)4f, z.getMyLast());
            assertEquals((Float)4f, z.getMyAcc1h());
      }

      @Test
      public void testGetAdjustedMyAcc1h() {
            StationData data = new StationData(Util.date(2016, 8, 1, 1, 0), "x",
0.2f, 2.0f, 0f, 0f, 0f, 0f, 0f);
            Float myAcc1h = 0.2f;
            Float expectedAcc1h = 2.0f;
            assertEquals(expectedAcc1h, c.getAdjustedMyAcc1h(data, myAcc1h));
            assertEquals((Float)2.0f, data.getMyLast());

            data = new StationData(Util.date(2016, 8, 1, 1, 0), "x", 0f, 0.2f,
0f, 0f, 0f, 0f, 0f);
            myAcc1h = 0f;
            expectedAcc1h = 0.2f;
            assertEquals(expectedAcc1h, c.getAdjustedMyAcc1h(data, myAcc1h));
            assertEquals((Float)0.2f, data.getMyLast());

            data = new StationData(Util.date(2016, 8, 1, 1, 0), "x", 0f, 1.2f,
0f, 0f, 0f, 0f, 0f);
            myAcc1h = 0f;
            expectedAcc1h = 1.2f;
            assertEquals(expectedAcc1h, c.getAdjustedMyAcc1h(data, myAcc1h));
            assertEquals((Float)1.2f, data.getMyLast());

            data = new StationData(Util.date(2016, 8, 1, 1, 0), "x", 0f, 2.0f,
0f, 0f, 0f, 0f, 0f);
            myAcc1h = 1.2f;
            expectedAcc1h = 2.0f;
            assertEquals(expectedAcc1h, c.getAdjustedMyAcc1h(data, myAcc1h));
            assertEquals((Float)0.8f, data.getMyLast(), 0.00001);
      }
}
```

Classe StationDataTest

```
package br.com.adache;

import static org.junit.Assert.*;

import java.util.SortedSet;
import java.util.TreeSet;

import org.junit.Test;

public class StationDataTest {

	@Test
	public void checkContainsDate() {
		SortedSet<StationData> data = new TreeSet<>();
		data.add(new StationData(Util.date(2016, 8, 1, 7, 20), "Badu"));
		assertTrue(data.contains(new StationData(Util.date(2016, 8, 1, 7,
20), "Badu")));
		assertFalse(data.contains(new StationData(Util.date(2016, 8, 1, 8,
20), "Badu")));
	}

	@Test
	public void checkContainsName() {
		SortedSet<StationData> data = new TreeSet<>();
		data.add(new StationData(Util.date(2016, 8, 1, 7, 20), "Badu"));
		data.add(new StationData(Util.date(2016, 8, 1, 7, 20), "Itaipu"));
		data.add(new StationData(Util.date(2016, 8, 1, 8, 20), "Itaipu"));
		assertTrue(data.contains(new StationData(Util.date(2016, 8, 1, 7,
20), "Badu")));
		assertFalse(data.contains(new StationData(Util.date(2016, 8, 1, 8,
20), "Badu")));
		assertTrue(data.contains(new StationData(Util.date(2016, 8, 1, 7,
20), "Itaipu")));
		assertTrue(data.contains(new StationData(Util.date(2016, 8, 1, 8,
20), "Itaipu")));
	}

}
```

Classe AllTests

```
package br.com.adache;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ CemadenTest.class, StationDataTest.class })
public class AllTests {

}
```

Classe Util

```
package br.com.adache;

import java.util.Date;
import java.util.GregorianCalendar;

public class Util {

	public static Date date(int year, int month, int day, int hour, int minute)
{
		return new GregorianCalendar(year,month-1,day,hour,minute).getTime();
	}
}
```

Arquivo de propriedades config.properties

```
####################################
# Arquivo de configuração do sistema
####################################

# Ambiente onde o sistema está sendo executado
AMBIENTE = MacBook Leonardo

# Nível log (pode ser NORMAL ou DEBUG)
LOG = NORMAL

# Endereço dados Niterói
URL =
http://150.163.255.234/salvar/mapa_interativo/interativo/grafico_CEMADEN.php?idpcd
=3727&uf=RJ
#URL =
http://150.163.255.234/salvar/mapa_interativo/interativo/grafico_CEMADEN.php?idpcd
=6788&uf=PA
#URL =
http://150.163.255.234/salvar/mapa_interativo/interativo/grafico_CEMADEN.php?idpcd
=6716&uf=AL
#URL =
http://150.163.255.234/salvar/mapa_interativo/interativo/grafico_CEMADEN.php?idpcd
=6422&uf=BA

# Identificador da tabela de dados no site do Cemaden
ID.TABELA = infopcds

# Intervalo de leitura dos dados no site do Cemaden (em minutos)
INTERVALO = 10

# Limites de chuva (em mm) para disparar alarmes
1H.SEM.ACUMULADO = 45
1H.COM.ACUMULADO = 40
24H = 100
48H = 125
96H = 175
MENSAL = 300

# E-mails para receber avisos (separados por vírgulas). Exemplo:
fulano@uol.com.br,ciclano@hotmail.com
EMAIL.AVISOS = ciclano@gmail.com,fulano@hotmail.com

# E-mails para receber erros (separados por vírgulas). Exemplo:
fulano@uol.com.br,ciclano@hotmail.com
EMAIL.ERROS = ciclano@gmail.com
```