Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Utilização de Técnicas de Inspeção e Monitoramento no Observatório da Web

Leonardo de Sousa Melo
Felipe Carvalho Gules

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Genaina Nunes Rodrigues

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenadora: Prof.ª Dr.ª Maristela Holanda

Banca examinadora composta por:

Prof.ª Dr.ª Genaina Nunes Rodrigues (Orientadora) — CIC/UnB
Prof. Me. Walter dos Santos Filho — DCC/UFMG
Prof. Dr. Rodrigo Bonifácio — CIC/UnB

CIP — Catalogação Internacional na Publicação

Melo, Leonardo de Sousa.

Utilização de Técnicas de Inspeção e Monitoramento no Observatório da Web / Leonardo de Sousa Melo, Felipe Carvalho Gules. Brasília : UnB, 2013.
219 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2013.

1. Rastreabilidade, 2. Falha, 3. Erro, 4. Defeito, 5. Inspeção, 6. Manutenção corretiva, 7. Manutenção preventiva, 8. Observatório da Dengue

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Utilização de Técnicas de Inspeção e Monitoramento no Observatório da Web

Leonardo de Sousa Melo
Felipe Carvalho Gules

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Genaina Nunes Rodrigues (Orientadora)
CIC/UnB

Prof. Me. Walter dos Santos Filho
DCC/UFMG

Prof. Dr. Rodrigo Bonifácio
CIC/UnB

Prof.ª Dr.ª Maristela Holanda
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 27 de Setembro de 2013

Dedicatória

Felipe - Primeiramente dedico este trabalho à toda a minha família e namorada pelo apoio e carinho durante todo o meu trajeto. Gostaria também de ressaltar a importante contribuição de todos as amizades que cultivei durante esta jornada e os bons momentos que passamos juntos.

Leonardo - Primeiramente, dedico este trabalho aos meus pais e meu irmão que sempre me passaram a confiança necessária para seguir em frente. Dedico este trabalho à minha namorada, que sempre esteve presente nos momentos mais difíceis de toda minha graduação, me apoiando e dando a atenção que sempre precisei.

Agradecimentos

Agradecemos todo o apoio dado por nossas famílias e namoradas durante essa etapa de nossas vidas.

Agradecemos à nossa orientadora, Genaína Nunes Rodrigues, que sempre esteve do nosso lado para a conclusão deste trabalho. Agradecemos pela liberdade, confiança e paciência que nos foi dada durante todo este longo percurso. Uma pessoa que possui luz própria, sendo não só um exemplo profissional como um exemplo de caráter, perseverança e carisma.

Agradecemos ao Walter dos Santos Filho (UFMG), que nos disponibilizou uma versão do Observatório da Dengue e sempre nos deu o auxílio necessário para que concluíssemos este trabalho da melhor maneira possível.

Agradecemos ao Thiago Araújo (PUC-RJ) por nos disponibilizar a Ferramenta de Inspenção desenvolvida em seu trabalho de Doutorado que foi essencial para o monitoramento da versão do Observatório da Dengue e pela atenção que sempre esteve disposto a dar quando era aberto um canal de comunicação.

Agradecemos ao Túlio Porto, pelas grandes discussões sobre o Observatório da Web e sobre outros diversos assuntos gerais, como políticos, econômicos, sociológicos e filosóficos, aumentando o nível intelectual e produtivo do nosso ambiente de trabalho.

Agradecemos à todos os que nos ajudaram e apoiaram a concluir esse trabalho nos desejando bons sentimentos.

Resumo

Este trabalho foca na correção do sistema Observatório da Dengue em tempo de execução através da aplicação de técnicas de dependabilidade e do uso de ferramentas de inspeção e monitoramento, com a finalidade de registrar erros e obter possíveis respostas às falhas antes que possam causar defeitos. O Observatório da Dengue é um sistema que apresenta um portal com assuntos específicos comentados nas mais diversas fontes da internet, como redes sociais, blogs e sites, sendo os dados apresentados de forma sintetizada através de metáforas visuais e indicadores disponíveis para os próprios usuários da internet. Neste sistema, há uma coleta massiva dos dados da internet, sendo processados em uma sequência (*pipeline*) que filtra e extrai a informação necessária para sua publicação na interface ao usuário final. Como a sequência é composta por vários componentes, é proposto o monitoramento dos principais componentes através da coleta de logs anotados com as informações de contexto. Será realizada a instrumentação de cada componente e, através de uma interface gráfica simples, serão mostradas todas as informações pontuais coletadas de cada componente da sequência do Observatório da Dengue. Também será realizado, paralelamente, o monitoramento infraestrutural em que o Observatório da Dengue está rodando, aumentando a cobertura.

Palavras-chave: Rastreabilidade, Falha, Erro, Defeito, Inspeção, Manutenção corretiva, Manutenção preventiva, Observatório da Dengue

Abstract

This work focus on the correction of the *Observatório da Dengue* system in time of execution through the aplication of techniques of dependability and the use of inspection tools and monitoring, with the point to register errors and obtain possible answers to the faults before they can cause defects. The *Observatório da Dengue* is a system that shows one portal with specifics subjects comments on the most various sources of the internet, like social network, blogs and sites, being the data present in the synthetical form by the visuals metaphors and indicators availables to the own users of internet. In this system there's a massive gathering of internet's data, being processed in a sequency (*pipeline*) which filters and extract the necessary information to the publication in the final user interface. As the sequency is composed by several components, it is proposed the monitoring of the main components by the gathering of the noted logs with the context information. It will be held the instrumentation of each component and, by the simple interface graphic, it will be showed all the punctual information gathered of each component of the sequency of *Observatório da Dengue*. Also will be held, at the same time, the infrastructural monitoring that the *Observatório da Dengue* is running, increasing the monitoring coverage.

Keywords: Tracebility, Fault, Error, Failure, Inspection, Corrective Maintenance, Preventive Maintenence, Observatório da Dengue

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contexto

O Observatório da Web é um projeto integrado ao InWeb (Instituto Nacional de Ciência e Tecnologia para a Web), formado por especialistas integrantes de quatro instituições federais de ensino: Universidade Federal de Minas Gerais (UFMG), Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Universidade Federal do Amazonas (UFAM) e Universidade Federal do Rio Grande do Sul (UFRGS) [5].

Com aproximadamente 65 milhões de brasileiros conectados à Internet – o correspondente a 36% da população, segundo dados do Comitê Gestor da Internet no Brasil – eventos como a campanha presidencial de 2010 começaram a refletir um fenômeno já observado em países desenvolvidos [4]. Cada vez mais, a Internet exerce um papel relevante na formação da opinião pública [4]. Com a proposta de acompanhar esta nova realidade, o Observatório da Web é uma ferramenta gratuita dedicada ao monitoramento de importantes fatos, eventos e entidades na rede mundial de computadores em tempo real [4].

Para a realização deste trabalho foi disponibilizado parte do Observatório da Dengue [3], que é um sistema de vigilância epidemiológica ativa a partir de dados coletados na internet. No Observatório, as informações sobre dengue são coletadas, analisadas e apresentadas em tempo real a partir de muitas fontes de dados da internet, incluindo redes sociais e blogs [3]. O Observatório oferece ao público em geral informações sobre vídeos, notícias, sites populares e conteúdo do Twitter sobre a dengue [3].

A Organização Mundial da Saúde estima que entre 50 e 100 milhões de pessoas são infectadas anualmente, em mais de 100 países, de diferentes continentes no mundo, com exceção da Europa [3]. Ao todo cerca de 500 mil pessoas são internadas, e mais de 20 mil pessoas morrem como consequência de suas complicações [3].

Segundo dados do Ministério da Saúde, a Dengue é considerada um dos grandes desafios em saúde publica e atinge todas as regiões brasileiras [3]. Entre 2002 e 2010 o número de casos prováveis de dengue no Brasil foi de aproximadamente quatro milhões, com destaque especial nas epidemias ocorridas nos anos de 2002, 2008 e 2010, sendo que no ano de 2010 foram registrados mais de um milhão de casos prováveis da doença, e destes, 63%, foram registrados nas regiões Centro-Oeste e Sudeste [3]. Estes dados revelam a importância do monitoramento contínuo da epidemiologia da dengue no país [3].

Em muitos sistemas de software, os dados são coletados continuamente durante a execução do programa, sendo armazenados em arquivos de log e analisados no caso de defeitos do sis-

tema e mal funcionamento, com a finalidade de identificar as causas e localizações dos problemas [13], ou seja, realizando uma manutenção corretiva no sistema. Porém, no Observatório da Dengue deseja-se monitorar, inspecionar e corrigir o sistema em tempo real, ou seja, realizar uma manutenção preventiva. Com isso, analisar arquivos de log gerados para cada componente do sistema e relacioná-los entre si não é a melhor escolha, considerando ainda a grande quantidade de logs coletados e distribuídos em diferentes arquivos durante a execução do sistema.

Buscando uma manutenção preventiva, faz-se necessário monitorar cada componente do Observatório da Dengue para que as falhas sejam eliminadas antes que elas possam ativar um erro que pode, alcançando a interface de serviço do componente, propagar-se em um defeito.

Para ser capaz de corrigir um sistema em tempo de execução, é possível aplicar algumas ferramentas no ambiente do Observatório da Dengue, como mecanismo de monitoramento de ambiente, que é capaz de monitorar diversos aspectos do sistemas alvo.

Na busca por ferramentas que pudessem auxiliar neste trabalho, duas ferramentas com abordagens diferentes e complementares foram selecionadas, a Ferramenta de Inspeção [6] surgiu como uma proposta para cobertura funcional e a ferramenta de monitoramento Zabbix [1] foi selecionada para a cobertura infraestrutural. Usando a ferramenta de monitoramento é possível analisar toda parte de infraestrutural e criar alertas, sinalizações enviadas por e-mail ou celular e analisar relatórios de eventos tornando rápido o conhecimento de erros. Já com a Ferramenta de Inspeção [6] busca-se verificar, por exemplo, padrões de falhas sem a necessidade de buscá-las manualmente em um arquivo de logs, detectar erros em determinados componentes e rastrear falhas através da interface gráfica simples, localizando dados, funcionalidades e falhas a partir da coleta de eventos realizada com a instrumentação de cada componente armazenados em um banco de dados NoSQL [16].

Para que seja possível uma manutenção preventiva, será utilizada, além da ferramenta de monitoramento [1] e da Ferramenta de Inspeção [6], uma sólida base teórica sobre dependabilidade [8] para classificação das falhas, dos erros e dos defeitos nos diferentes componentes que compõem a sequência do Observatório da Dengue.

1.2 Motivação

Dada a importância de manter um sistema como o Observatório da Dengue no ar, torna-se necessária a criação de um painel de controle que monitore seus componentes em tempo real, possibilitando aumentar a qualidade do serviço.

1.3 Problema

Dado um sistema complexo, como o Observatório da Dengue com diversos componentes e módulos interagindo entre si, considera-se que é um sistema onde falhas ocorrem com frequência e requer um esforço considerável para recuperar o estado correto do sistema, pois os logs de falhas estão espalhados em diferentes arquivos tornando dispendiosa sua manutenção.

1.4 Hipótese

É possível aplicar técnicas e ferramentas de inspeção e monitoramento no Observatório da Dengue, atuando em níveis distintos para detecção de falhas, erros e defeitos, obtendo formas eficientes de capturar e monitorar suas falhas.

1.5 Objetivo geral

Auxiliar na construção de um ambiente do Observatório da Dengue que aumente o nível de dependabilidade do sistema.

1.6 Objetivos específicos

- Entender o funcionamento do Observatório da Dengue;
- Conseguir ferramentas que permitam monitorar as falhas do Observatório da Dengue;
- Conseguir classificar os defeitos e suas causas de forma eficiente.

1.7 Descrição dos capítulos

O trabalho está organizado da seguinte forma:

- **Capitulo 2:** Reúne a descrição do Observatório da Dengue, conceitos e referencial teórico sobre dependabilidade, ferramenta de inspeção e ferramenta de monitoramento.
- **Capitulo 3:** Apresenta-se a metodologia adotada, o projeto do ambiente e como foram desenvolvidos e aplicados os mecanismos no ambiente.
- **Capitulo 4:** Mostra os resultados obtidos e a análise feita com base nos objetivos definidos.
- **Capitulo 5:** Conclui-se verificando se os objetivos foram atingidos, apresentando os trabalhos futuros e as dificuldades encontradas na realização do trabalho.
- **Capitulo 6:** Apêndice com diversos dados coletados e classificações realizadas ao longo do trabalho.

Capítulo 2

Conceitos Básicos

Com o objetivo de expor ao leitor conceitos que servirão como base para a compreensão plena deste trabalho, este capítulo aborda os temas sobre sistemas, dependabilidade e segurança, Observatório da Web, Ferramenta de Inspeção e Ferramenta de Monitoramento.

2.1 Sistemas

É necessário fazer uma cobertura geral sobre função do sistema, comportamento, estrutura e serviços, pois neste trabalho será realizada a instrumentação de um sistema e a inspeção do mesmo.

São observadas as seguintes definições que auxiliam nesta cobertura [8]:

- **Sistema:** é uma entidade que interage com outras entidades. Exemplo disso seriam outros sistemas, incluindo hardware, software, humanos, e o mundo físico com seus fenômenos naturais;
- **Ambiente:** são estes outros sistemas que interagem com o dado sistema;
- **Fronteira do sistema:** é a fronteira comum entre o sistema e seu ambiente;
- **Função do sistema:** qual a finalidade do sistema que é descrita pela especificação funcional nos termos de funcionalidade e performance;
- **Comportamento do sistema:** o que o sistema faz para implementar sua função e é descrito por uma sequência de estados;
- **Estado total (*total state*) de um sistema:** é o conjunto dos seguintes estados: computação, comunicação, informação armazenada, interconexão, e condição física;
- **Estrutura de um sistema:** é o que habilita a geração do comportamento.

De uma visão estrutural, um sistema é composto por um conjunto de componentes ligados entre si, a fim de interagir, onde cada componente é um outro sistema. A recursão para quando o componente é considerado atômico. Qualquer outra estrutura interna não pode ser discernida, ou não é interessante e pode ser ignorada. Consequentemente, os estados totais de um sistema é o conjunto de estados externos de seus componentes atômicos [8].

- **Serviço provido pelo sistema:** é o comportamento percebido pelo usuário;

- **Usuário:** é outro sistema que recebe o serviço do provedor;
- **Interface de serviço:** é a parte do provedor *fronteira do sistema* onde o serviço entregue é percebido;
- **Estado externo:** é a parte do provedor *estado total* que é percebida na interface de serviço;
- **Estado interno:** é a parte restante do *estado externo*;
- **Interface do usuário:** é onde o usuário recebe o serviço.

O serviço entregue é uma sequência do provedor *estados externos*. Percebe-se que um sistema pode ser sequencialmente e simultaneamente um provedor e um usuário em relação à outro sistema, por exemplo, *entregar serviço para* e *receber serviço de* outro determinado sistema [8].

Um sistema geralmente implementa mais de uma função e entrega mais de um serviço. Função e serviço podem assim ser vistos como um composto de itens de função e de itens de serviço.

2.2 Dependabilidade

Desenvolver um sistema com alta dependabilidade requer um grau de conhecimento elevado sobre características de falhas. Segundo Laprie [12], dependabilidade é a propriedade que define a capacidade dos sistemas computacionais de prestar um serviço que se pode justificadamente confiar, segundo Avizienis et al [7] é a habilidade de entregar um serviço que se pode justificadamente confiar, e segundo Avizienis et al [8], dependabilidade é a habilidade de evitar defeitos que são mais frequentes e mais severos que o aceitável.

Os termos *fault*, *error* e *failure* foram traduzidos para falha, erro e defeito [9] [20], respectivamente.

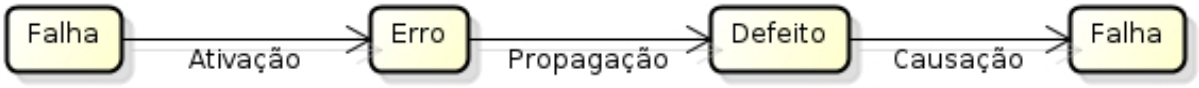


Figura 2.1: Encadeamento das ameaças [8]

2.2.1 Conceitos básicos

Ameaças ao sistema

Para que se possa fazer uma análise de dependabilidade, é necessário estabelecer como são classificadas as ameaças ao sistema. São elas [8]:

- **Falha:** é a causa julgada ou hipótese para o erro;
- **Erro:** é a parte do *estado total* do sistema que pode levar, subsequentemente, para o defeito no serviço;

- **Defeito:** é um evento que ocorre quando um serviço entregue se desvia do serviço correto.

O *serviço correto* é entregue quando implementa a função do sistema. O defeito é a transição de um serviço correto para um serviço incorreto, ocorrendo quando um erro alcança a interface de serviço e altera o serviço. A transição de serviço incorreto para serviço correto é uma *restauração do serviço* (*service restoration*). O intervalo de tempo onde um serviço incorreto é entregue é chamado de *interrupção de serviço* (*service outage*). A falha pode ser interna ou externa e é ativa quando produz um erro, do contrário, é dormente. Um erro é *detectado* se a sua presença é indicada por uma *mensagem de erro* ou *sinal de erro*. Caso haja erros presentes mas não detectados, então são erros *latentes*. O desvio do serviço correto pode assumir diferentes formas chamadas de *modos de defeitos de serviço* e são classificados de acordo com as *severidades do defeito* [8].

Um sistema consiste em um conjunto de componentes que interagem, então o estado do sistema é o conjunto de estados do componente. Uma falha originalmente causa um erro no interior do estado de um ou mais componentes, mas o defeito no sistema não irá ocorrer enquanto o erro não alcançar a interface de serviço do sistema [7]. A presença anterior de uma *vulnerabilidade*, ou seja, uma falha interna que habilita uma falha externa para danificar o sistema, é necessária para uma falha externa causar um erro e, possivelmente, defeito(s) subsequente(s). Na maior parte dos casos, uma falha primeiramente causa um erro em um *estado de serviço de um componente* que é parte de um *estado interno* do sistema e o *estado externo* não é afetado de imediato [8].

Quando a *especificação funcional* do sistema inclui um conjunto de muitas funções, o defeito de um ou mais desses serviços que implementam as funções podem deixar o sistema em um *modo degradado* que ainda oferece um subconjunto de serviços necessários para o usuário. A especificação pode identificar muitos desses modos, como serviço limitado, serviço emergencial, entre outros. Podemos dizer que o sistema sofreu um *defeito parcial* de suas funcionalidades e performance [8].

Relação entre falhas, erros e defeitos

A criação e a manifestação dos mecanismos de falhas, erros, e defeitos é ilustrada na figura 2.2 e são sumarizadas a seguir [8]:

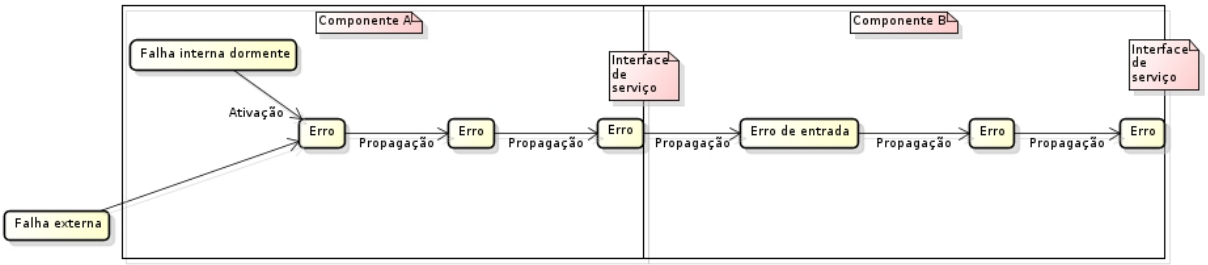


Figura 2.2: Diagrama de propagação de erros [8]

- **Falha:**
  - uma falha é ativa quando produz um erro, caso contrário, é dormente. Uma falha ativa também é:

1. uma falha interna que estava anteriormente dormente e que foi ativada por um processo computacional ou condições ambientais, ou
2. uma falha externa.

– *ativação de falha* é a aplicação de uma entrada (o padrão de ativação) para um componente fazendo com que a falha dormente se torne ativa. Ciclo de falhas mais interna entre seus estados dormentes e ativo.

- **Erro:**

  – propagação de erro de um dado componente (*propagação interna*) é causado por um processo computacional:

    1. um erro é sucessivamente transformado em outros erros;
    2. propagação de erro de um componente A para um componente B que recebe um serviço de A (*propagação externa*) ocorre quando, através de propagação interna, um erro alcança a *interface de serviço* do componente A. Neste momento, o serviço entregue por A para B se torna incorreto, e o *defeito de serviço* subsequente de A aparece como uma falha externa de B e propaga o erro para B pela sua *interface de uso*.

- **Defeito:**

  – um *defeito de serviço* ocorre quando um erro é propagado para a *interface de serviço* e faz com que o serviço entregue pelo sistema se desvie do serviço correto.

    1. o defeito de um componente causa uma falha permanente ou transitória no sistema que contêm o componente.
    2. o defeito de serviço de um sistema causa uma falha externa permanente ou transitória para o sistema que recebe o serviço do dado sistema.

Esses mecanismos habilitam um encadeamento de ameaças completo, mostrado na Figura 2.1.

A habilidade de identificar um padrão de ativação de uma falha que causa um ou mais erros é a *reprodutibilidade de ativação de falhas* (*fault activation reproducibility*). Falhas podem ser categorizadas de acordo com sua reprodutibilidade de ativação: Falhas cuja sua ativação é reproduzível, são chamadas *falhas sólidas* (*solid*), ou *difíceis* (*hard*), e onde falhas cuja ativação não é sistematicamente reproduzíveis são *falhas elusivas* (*elusive*), ou *leves* (*soft*). A maior parte das falhas de desenvolvimento em software complexo e de grande porte, são falhas elusivas [8].

A similaridade da manifestação de falhas de desenvolvimento elusivas e de falhas físicas transitórias, leva ambas as classes a serem agrupadas juntas como *falhas intermitentes* (*intermittent faults*). Erros produzidos por falhas intermitentes são normalmente nomeadas como *erros leves* (*soft errors*) [8].

Dado um sistema com fronteiras definidas, uma *falha isolada* (*single fault*) é uma falha causada por um evento físico adverso ou uma ação humana danosa. *Falhas múltiplas* (*multiple faults*) são duas ou mais falhas sequenciais concorrentes, sobrepostas, ou isoladas cujas consequências, ou seja, os erros advindos dessas falhas são presentes no sistema de forma concorrente. Análise de falhas múltiplas nos leva a distinguir [8]:

1. *falhas independentes:* que são atribuídas para causas diferentes;

2. *falhas relacionadas:* que são atribuídas para uma causa comum.

Falhas relacionadas geralmente causam *erros similares*, por exemplo, erros que não podem ser distinguidos por qualquer que sejam os mecanismos que são empregados, visto que falhas independentes normalmente causam *erros distintos*. Os defeitos causados por erros similares são *modo comum de defeitos* (*common-mode failures*).

Atributos de dependabilidade e segurança

Dependabilidade é uma integração dos conceitos de: disponibilidade, confiabilidade, segurança (*safety*), integridade e manutenibilidade [8].

- **Disponibilidade:** capacidade do sistema entregar um serviço correto;
- **Confiabilidade:** serviço correto de forma contínua;
- **Segurança (*safety*):** é a ausência de consequências catastróficas nos usuários e no ambiente;
- **Integridade:** ausência de alterações impróprias no sistema;
- **Manutenibilidade:** habilidade de se submeter à modificações e reparos.

Quando falamos de segurança (*security*), um outro atributo ganha importância, que é a confidencialidade:

- **Confidencialidade:** é a ausência de divulgação não autorizada de informação.

Segurança (*security*) é a composição dos atributos de confidencialidade, integridade e disponibilidade, que requer a existência de:

1. *disponibilidade* apenas para ações autorizadas;
2. *confidencialidade*;
3. *integridade* com significado impróprio não autorizado.

Integridade é um pré requisito para disponibilidade, confiabilidade e segurança (*safety*), mas pode não ser para confidencialidade. A definição de *integridade* pode ser extendida para [7]:

1. quando o sistema implementa um policiamento autorizado, impróprio engloba não autorizado;
2. alterações impróprias englobam ações que resultam em atualizações preventivas de informação;
3. estado do sistema engloba modificações e danos no hardware.

Mesmo com todos esses atributos citados quando nos referimos à dependabilidade e segurança, não necessariamente um sistema precisa que todos os atributos sejam satisfeitos, pois pode não haver necessidade de existir determinados atributos para determinados sistemas.

Pode-se inferir que a dependabilidade e segurança é alta quando seus atributos que compõem o sistema possuem uma alta avaliação.

Como alcançar a dependabilidade e segurança de um sistema?

Durante os últimos 50 anos, muitos meios foram desenvolvidos para atingir os vários atributos de dependabilidade e segurança [8]. Esses meios podem ser agrupados em quatro principais categorias:

- **Prevenção de falhas:** meios para prevenir a ocorrência ou a introdução de falhas.
- **Tolerância a falhas:** meios para evitar falhas de serviço na presença de falhas.
- **Remoção de falhas:** meios para reduzir o número e a severidade de falhas.
- **Previsão de falhas:** meios de estimar o número presente, a incidência futura, e as prováveis consequências das falhas.

Prevenção de falhas e tolerância a falhas tem o objetivo de proporcionar a habilidade de entregar um serviço que é confiável e remoção de falhas e previsão de falhas tem o objetivo de alcançar a confiança da habilidade por meio da justificativa de que as especificações funcionais e da dependabilidade e segurança são adequadas e que o sistema é susceptível de alcançá-los [8].

2.2.2 Ameaças no ciclo de vida do sistema

Ciclo de vida de um sistema

O ciclo de vida de um sistema é dividido em duas fases: *fase de desenvolvimento* e *fase de uso*.

Na **fase de desenvolvimento** o sistema interage com o ambiente de desenvolvimento e as *falhas de desenvolvimento* devem ser introduzidas no sistema pelo ambiente. O *ambiente de desenvolvimento* é composto pelos elementos: o mundo físico, os desenvolvedores, as ferramentas de desenvolvimento e as instalações de produção e testes [8].

A **fase de uso** inicia-se quando o sistema é aceito e ele começa a ser disponibilizado para os usuários. O *uso* consiste na alternância dos períodos de uma *correta prestação de serviços*, *interrupção de serviço* e *desligamento do serviço* (uma parada proposital da prestação do serviço por uma entidade autorizada). A manutenção tem lugar nos três períodos da *fase de uso*. O ambiente de uso é composto pelos elementos: o mundo físico, administradores, usuários, fornecedores, infraestrutura e intrusos [8].

A manutenção não inclui somente reparos físicos, como também modificações do sistema que possuem lugar durante a *fase de uso*. É possível notar que reparo e *tolerância a falhas* são conceitos relacionados. As várias formas de manutenção são sumarizadas na Figura 2.3 [8].

A diferença entre a *tolerância a falhas* e manutenção é que a manutenção envolve a participação de um agente externo, por exemplo, um reparador, equipamento de teste, controle remoto de recarga de software. Além disso, a diferença é que reparo é parte de *remoção de falhas* (durante a *fase de uso*), e *previsão de falhas* normalmente considera situações de reparo. De fato, reparo pode ser visto como uma atividade de *tolerância a falhas* dentro de um sistema maior que inclui o sistema a ser reparado e as pessoas e outros sistemas que executam tais reparos [8].

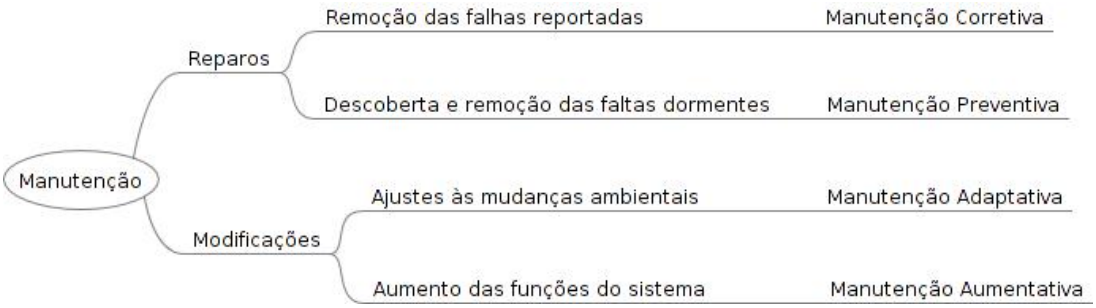


Figura 2.3: As várias formas de manutenção [8]

Classificação das falhas

As classes de falha combinada da Figura 2.5a pertencem à três grupos parcialmente sobrepostos:

- *falhas de desenvolvimento* que incluem todas as classes de falha que ocorrem durante o desenvolvimento,
- *falhas físicas* que incluem todas as classes de falha que afetam hardware,
- *falhas de interação* que incluem todas as falhas externas.

As caixas na parte de baixo da Figura 2.5 identificam os nomes de algumas classes de falha ilustrativa.

Todas as falhas que podem afetar o sistema durante sua vida são classificadas em oito visões básicas, que levam à uma classe de falha elementar, como observado na Figura 2.4. Caso seja possível todas as combinações das classes elementares de falhas, significa que 256 combinações de classes de falhas diferentes são possíveis. No entanto, 31 combinações diferentes são possíveis. As oito visões básicas são [8]:

- **Fase de criação:**
  - De desenvolvimento (*Development*): ocorre durante a fase de desenvolvimento do sistema.
  - Operacional (*Operacional*): ocorre durante a entrega do serviço proposto.
- **Localização:**
  - Interna (*Internal*): ocorre dentro do sistema.
  - Externa (*External*): ocorre fora dos limites do sistema e provoca danos durante a interação com o mesmo.
- **Causa:**
  - Natural (*Natural*): causada por fenômenos da natureza e sem intervenção humana.
  - Humana (*Human-Made*): é resultado de intervenções impróprias do homem.
- **Dimensão:**

Figura 2.4: Categoria de falhas - Adaptadado de Avizienis et al [8]

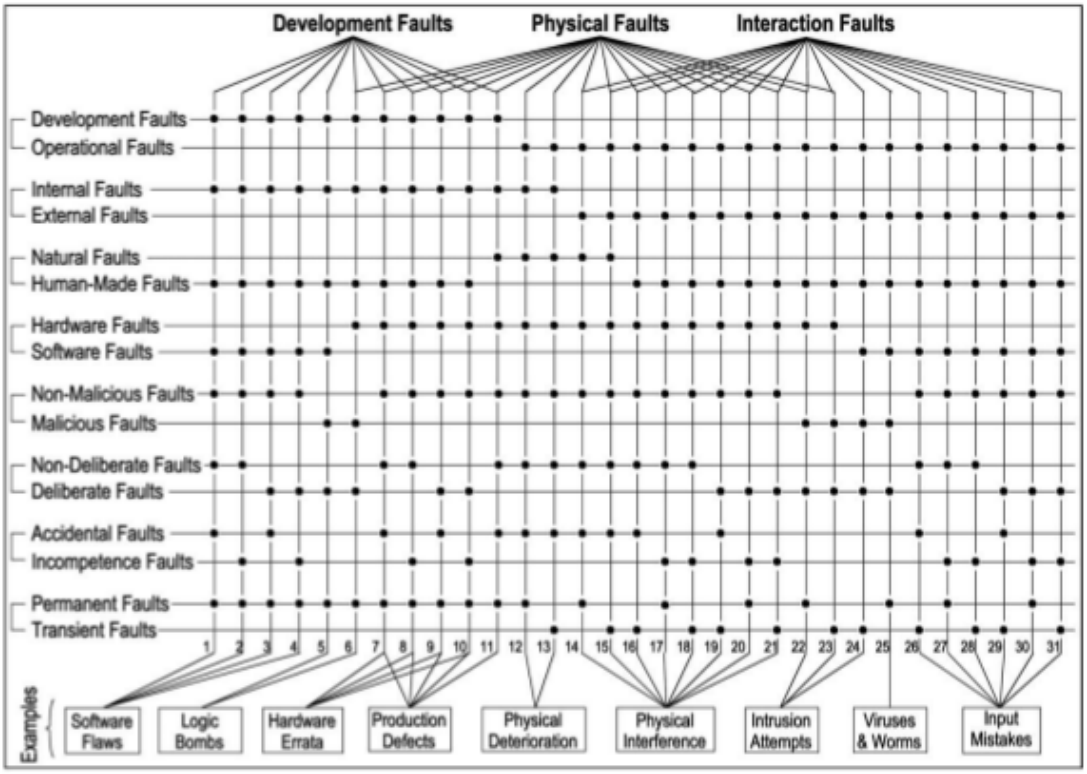
  - Hardware: se origina no hardware ou afeta o mesmo.
  - Software: de alguma forma afeta o software do sistema.

- **Objetivo:**
  - Maliciosa (*Malicious*): introduzida por um humano com o objetivo malicioso de causar danos ao sistema. Pode ser introduzida durante o desenvolvimento do sistema com o objetivo de causar dano durante seu uso ou diretamente durante o uso.
  - Não Maliciosa (*Non-Malicious*): introduzida sem um objetivo malicioso.

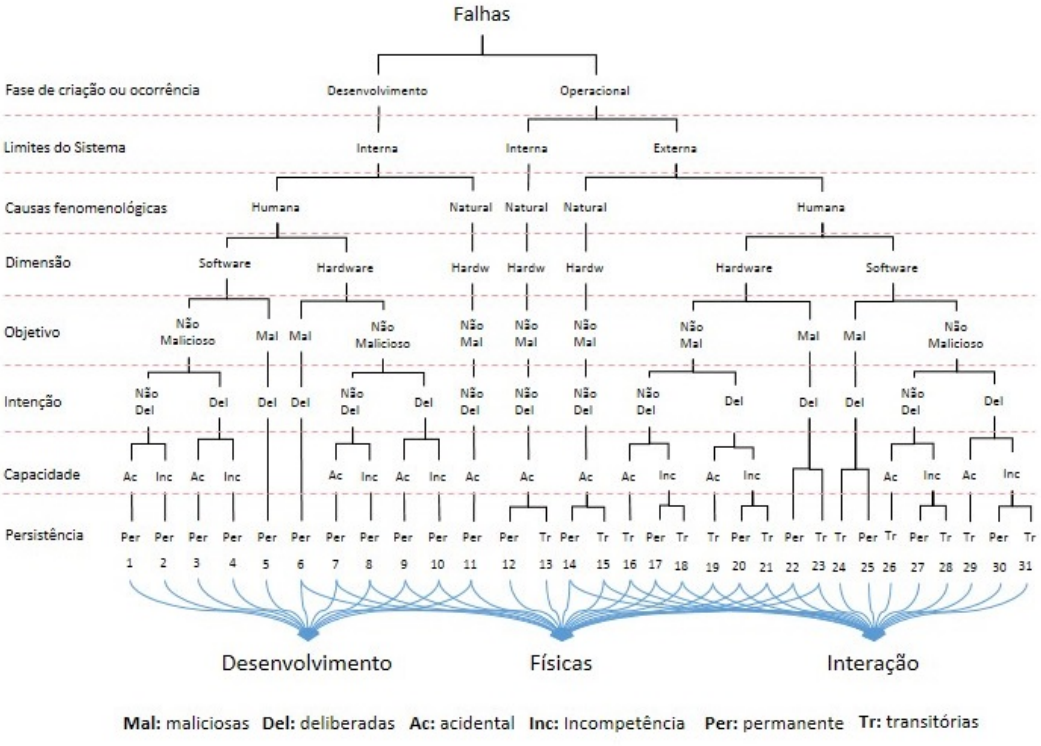
- **Intenção:**
  - Deliberada (*Deliberate*): resultado de uma má decisão, ou seja, uma ação intencional que é errada e causa falhas.
  - Não Deliberada (*Non-Deliberate*): introduzida sem consciência, ou seja, equivocadamente por uma ação não intencional.

- **Capacidade:**
  - Acidental (*Accidental*): introduzida inadvertidamente.
  - Incompetência (*Incompetence*): resultado da falta de competência profissional pelo ser humano autorizado ou da inadequação da organização de desenvolvimento.

- **Persistência:**
  - Persistente (*Persistent*): persiste durante o tempo a partir da sua ativação.
  - Transitória (*Transient*): ocorre durante um período limitado de tempo.



(a) Representação em Matriz [8]



(b) Representação em árvore de Leal e Azevedo [11] adaptado de [8]

Figura 2.5: As classes das falhas combinadas [8]

Classificação dos defeitos

Em relação aos defeitos, é de conhecimento que todo defeito é resultado de pelo menos uma propagação de falha, mas nem toda falha deixa o sistema em estado errôneo que por consequência propaga-se em defeito. Observando a Figura 2.6, os defeitos são classificados [8]:

Figura 2.6: Categoria de defeitos [8]

- Quanto ao **domínio**:
  - De conteúdo: quando o conteúdo entregue difere do especificado.
  - De tempo: quando o tempo de entrega do serviço difere do especificado. Pode ser tanto quando o serviço é entregue antes do previsto (*early timing failure*) ou quando ocorre depois do previsto (*late timing lailure*).
- Quando ocorre tanto defeito de **conteúdo** quanto de **tempo**:
  - Defeito de parada (*halt failure*): quando o sistema não apresenta nenhuma atividade. Um caso especial desse tipo de defeito é chamado de Defeito Silencioso quando nenhum serviço é entregue, nem mesmo mensagens de defeito.
  - Defeito instável (*erratic failure*): quando o conteúdo é entregue, porém com defeito, ou seja, diferente do especificado.
- Quanto a **detectabilidade**:
  - Sinalizadas: quando a perda é detectada.
  - Não sinalizadas: quando não ocorre detecção da perda.
- Quanto a **consistência do defeito**:
  - Consistente: quando o defeito é percebido igualmente por todo o sistema.
  - Inconsistente: quando o defeito é percebido de forma diferente por cada usuário do sistema, sendo que não necessariamente todos precisem perceber o defeito.
- Quanto a **consequência do defeito**:

- Defeito pequeno: são de custo semelhante ao do benefício da entrega do serviço correto.
- Defeito médio: são de um custo maior ao do benefício da entrega do serviço correto.
- Defeito grave: custos podem ser incomensuravelmente maiores ao do benefício da entrega do serviço correto.

Sistemas que são projetados e implementados de forma que eles fiquem defeituosos apenas em modos específicos de defeitos descritos na especificação de dependabilidade e segurança e apenas em uma extensão aceitável, são chamados de *sistemas de defeitos controlados* (*fail-controlled systems*). Um sistema cujos defeitos são de uma extensão aceitável de *parada de defeitos* (*halting failures*) apenas, é um *sistema de parada de defeito* (*fail-halt systems*); as situações de *sistema preso* (*stuck service*) e de *sistema silencioso* (*silence lead*), respectivamente, para *sistemas passivos de defeitos* (*fail-passive systems*) e *sistemas de silencioso defeito* (*fail-silent systems*). Um sistema cujos defeitos são, de uma extensão aceitável, todos de *defeito pequeno* é um *sistema contra defeito* (*fail-safe system*) [8].

A *interrupção do serviço* pode variar significativamente, dependendo das ações envolvidas na *restauração do serviço* depois que o defeito já ocorreu: automática ou recuperação por operador assistido, recomeçar, ou resetar; manutenção corretiva. Correção de falhas de desenvolvimento são normlamente realizadas *fora do ar* (*offline*), depois da restauração do sistema, e os componentes atualizados resultantes da correção da falha são então introduzidos em algum momento apropriado com ou sem interrupção da operação do sistema. Uma interrupção intencional da operação do sistema para uma atualização ou *manutenção preventiva* é um *desligamento do serviço*, também chamada de *interrupção planejada* [8].

É esperado que vários tipos de falhas vão afetar o sistema durante a fase de uso. As falhas podem causar uma inaceitável degradação de desempenho ou um defeito total para entregar o serviço especificado. Por essa razão, a especificação de dependabilidade e segurança está de acordo com as metas para cada atributo [8].

A especificação identifica explicitamente as classes de falhas que são esperadas e o *ambiente de uso* onde o sistema irá operar. A especificação pode requerer também salvaguardas contra certas condições indesejadas ou perigosas. Além disso, a inclusão de especificações de técnicas de prevenção de falhas ou tolerância a falhas podem ser requeridas pelo usuário. A especificação também pode conter falhas [8].

A dependabilidade ou segurança de defeitos ocorre quando um dado sistema sofre defeitos de serviço mais frequentemente ou mais serveramente do que o aceitável [8].

Classificação dos erros

Uma classificação conveniente de erros seria descrevê-los em termos dos elementares defeitos de serviço que eles causam. Algumas falhas podem causar simultaneamente erros em mais de um componente. Esses erros são chamados de *vários erros relacionados* (*multiple related errors*). *Erros individuais* (*single errors*) são erros que afetam apenas um único componente [8].

São dois fatores para que um erro se propague ou não a um defeito no serviço [8]:

1. A estrutura do sistema e a natureza de qualquer redundância que existe:

- redundância de proteção, introduzida para prover tolerância a falhas, que é explicitamente pretendida para previnir um erro de gerar um defeito de serviço.
- redundância não intencional, que pode ter o mesmo resultado da redundância intencional.

2. O comportamento do sistema:

   - a parte do estado que contêm um erro pode nunca ser necessária para o serviço, ou um erro pode ser eliminado antes de gerar um defeito.

2.2.3 Como alcançar sistemas sem falhas?

Prevenção de falha

É a parte da engenharia geral. Prevenir falhas de desenvolvimento é uma visão óbvia para desenvolver metodologias, tanto para software quanto para hardware. Melhorias no processo de desenvolvimento para reduzir o número de falhas introduzidas nos sistemas produzidos é um degrau a frente naquilo em que é baseado na gravação de falhas nos produtos e a eliminação das causas das falhas via modificação de processos [8].

Remoção de falhas

Remoção de falhas *durante o uso* de um sistema é uma manutenção corretiva ou preventiva. Manutenção corretiva visa eliminar falhas que produziram um ou mais erros e têm sido relatados, enquanto que a manutenção preventiva tem como objetivo descobrir e eliminar falhas antes que elas possam causar erros durante a operação normal. As últimas falhas incluem [8]:

1. as falhas físicas que ocorreram desde as últimas ações de manutenção preventiva e
2. as falhas de desenvolvimento que levaram a erros em outros sistemas semelhantes.

Manutenção corretiva de falhas de desenvolvimento é geralmente realizado em etapas: A falha pode ser isolada pela primeira vez antes da remoção real estar concluída. Estas formas de manutenção se aplicam a sistemas não tolerantes a falhas, bem como para sistemas tolerantes a falhas, que podem ser de fácil manutenção online (sem a interrupção da entrega do serviço) ou offline (durante a interrupção do serviço) [8].

Relações entre os meios de dependabilidade e segurança

As imperfeições das análises humanas trazidas nas relações explicam o por que de ser apenas a combinação das atividades (prevenção, tolerância, remoção e previsão de falhas, preferivelmente em cada passo do processo de desenho e implementação) que podem melhor levar à um sistema computacional confiável e seguro (*dependable and secure*). Essas relações podem ser esboçadas como segue [8]:

- **Prevenção de falhas**: apesar de ser realizada por meio de metodologias de desenvolvimento e regras de construção, falhas podem ocorrer. Por conta disso, existe a necessidade de remoção de falhas.

- **Remoção de falhas**: é imperfeita, pois nem todas as falhas podem ser encontradas e outras falhas podem ser introduzidas quando ocorrer a remoção de uma falha e, também, componentes terceirizados (*off-the-shelf components*) do sistema podem (e normalmente possuem) conter falhas. Por isso a importância da previsão de falhas.
- Nossa crescente dependência dos sistemas de computação traz a exigência de tolerância a falhas, que por sua vez é baseada em regras de construção. Por isso há novamente a necessidade da aplicação de remoção de falhas e previsão de falhas para os próprios mecanismos de tolerância a falhas.

Deve-se notar que o processo é ainda mais recursivo do que parece: sistemas de computação atuais são tão complexos que sua concepção e execução precisam de ferramentas de software e hardware com a finalidade do custo ser viável (em sentido mais amplo, incluindo a capacidade de ter sucesso dentro de um prazo aceitável). Essas ferramentas devem ser confiáveis e seguras também. E assim continua a recursividade [8].

O raciocínio anterior ilustra as interações estreitas entre a remoção de falhas e a previsão de falhas, motivando sua reunião na análise de dependabilidade e segurança, visando alcançar a confiança na capacidade de oferecer um serviço confiável, enquanto que o grupo de prevenção de falhas e tolerância a falhas constituem a provisão de dependabilidade e segurança, que visa proporcionar a capacidade de oferecer um serviço confiável. Outro agrupamento dos meios é a associação de [8]:

1. a prevenção de falhas e remoção de falhas na anulação das falhas, ou seja, como alcançar sistemas sem falhas e
2. tolerância a falhas e previsão de falhas em aceitação de falhas, ou seja, como viver com sistemas sujeitos a falhas.

Além de destacar a necessidade de avaliar os procedimentos e mecanismos de tolerância a falhas, a consideração de remoção de falhas e previsão de falhas como dois componentes de uma mesma atividade (*análise de dependabilidade*) leva a uma melhor compreensão da noção de cobertura e, portanto, de um importante problema introduzido pela recursão: a avaliação da avaliação, ou como chegar à confiança nos métodos e ferramentas utilizadas na construção de confiança do sistema. *Cobertura* é referida como uma medida sobre a representatividade das situações em que o sistema está sujeito durante a sua análise comparando-se com as situações reais em que o sistema será confrontado durante a sua vida operacional. A noção de cobertura é muito geral, podendo ser realizada mais precisamente por indicação do seu campo de aplicação, por exemplo, a cobertura de um teste de software que respeita o texto do software, gráfico de controle, etc., a cobertura de um teste de integração de circuito com relação um modelo de falha, a cobertura de tolerância a falhas com respeito à uma classe de falhas, a cobertura de uma hipótese de desenvolvimento com relação à realidade [8].

A avaliação sobre se um sistema é realmente seguro, ou seja, oferece um serviço justificadamente confiável, vai além das técnicas de análise de como elas foram abordadas anteriormente por, pelo menos, as três seguintes razões e limitações [8]:

- Verificação precisa da cobertura do desenho ou validação de hipóteses com relação à realidade implicaria um conhecimento e um domínio da tecnologia utilizada, da utilização pretendida no sistema, etc., que excede em muito o que é geralmente realizável.

- A avaliação de um sistema para alguns atributos de dependabilidade e especialmente de segurança com relação a certas classes de falhas são atualmente consideradas como inviável ou como produzindo resultados não significativos, pois as bases de teoria propabilística não existem e não são amplamente aceitas. Exemplos são a segurança (*safety*) no que diz respeito à falhas acidentais de desenvolvimento e segurança (*security*) no que dis respeito à falhas intencionais.
- As especificações no que diz respeito à qual análise é executada são sucetíveis de conter falhas como qualquer sistema.

Entre essas inúmeras consequências, há também [8]:

- A ênfase colocada no processo de desenvolvimento quando há avaliação de um sistema, ou seja, sobre os métodos e técnicas utilizadas no desenvolvimento e como eles são empregados. Em alguns casos, uma nota é atribuída e emitida para o sistema de acordo com:
  1. a natureza dos métodos e as técnicas empregadas no desenvolvimento e
  2. uma avaliação de sua utilização.
- A presença, nas especificações de alguns sistemas tolerantes a falhas, de uma lista de tipos e números de erros que devem ser toleradas. Tal especificação não seria necessária se as limitações mencionadas acima pudessem ser superadas.

2.3 Observatório da Web

2.3.1 Conceitos básicos

O sistema apresenta um portal com assuntos específicos comentados nas mais diversas fontes da internet, como redes sociais, blogs e sites, sendo os dados apresentados de forma sintetizada através de metáforas visuais e indicadores disponíveis para os próprios usuários da internet. A instância analisada no trabalho é apenas uma possível configuração do Observatório, neste caso, o da Dengue [3]. Há várias outras, mas para fins de estudo inicial foi esta a escolhida.

O Observatório da Dengue [3] é um sistema de vigilância epidemiológica ativa a partir de dados coletados na internet. No Observatório, as informações sobre dengue são coletadas, analisadas e apresentadas em tempo real a partir de muitas fontes de dados da internet, incluindo redes sociais e blogs [3]. O Observatório 2.7 oferece ao público em geral informações sobre vídeos, notícias, sites populares e conteúdo do Twitter sobre a dengue [3].

A Organização Mundial da Saúde estima que entre 50 e 100 milhões de pessoas são infectadas anualmente, em mais de 100 países, de diferentes continentes no mundo, com exceção da Europa [3]. Ao todo cerca de 500 mil pessoas são internadas, e mais de 20 mil pessoas morrem como consequência de suas complicações [3].

Segundo dados do Ministério da Saúde, a Dengue é considerada um dos grandes desafios em saúde publica e atinge todas as regiões brasileiras [3]. Entre 2002 e 2010 o número de casos prováveis de dengue no Brasil foi de aproximadamente quatro milhões, com destaque especial nas epidemias ocorridas nos anos de 2002, 2008 e 2010, sendo que no ano de 2010 foram registrados mais de um milhão de casos prováveis da doença, e destes, 63%, foram registrados

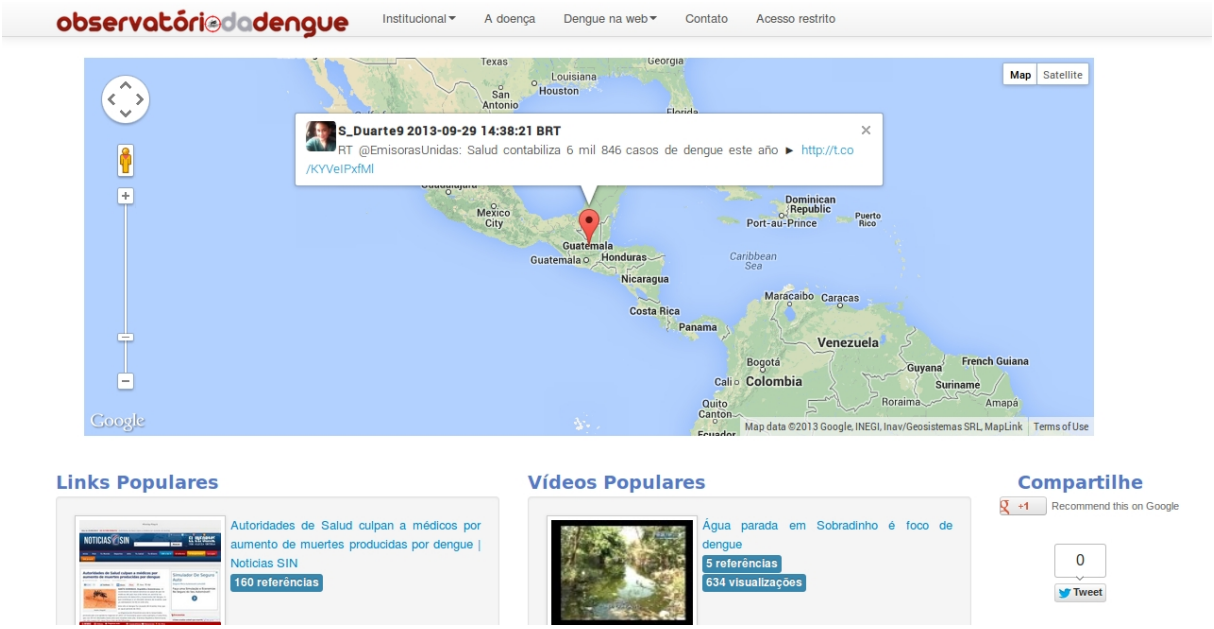


Figura 2.7: Página principal do Observatório da Dengue [3]

nas regiões Centro-Oeste e Sudeste [3]. Estes dados revelam a importância do monitoramento contínuo da epidemiologia da dengue no país [3].

O sistema divide-se em: modelo conceitual, arquitetura do sistema e arcabouço de software [5], como pode ser visualizado na Figura 2.8. Nosso foco será na arquitetura do sistema.

O modelo conceitual é composto por sete definições básicas [5]:

- Contexto - Assunto geral observado na Web. Exemplo: eleições presidenciais.
- Entidades - Algo que está sendo observado. Exemplo: pré-candidatos.
- Fontes - Local de onde são retiradas as informações observadas. Exemplo: Twitter e Facebook.
- Temas - Assuntos específicos percencentes ao contexto. Exemplo: questões sendo discutidas como saúde, educação e economia.
- Grupos - Conjuntos de entidades. Exemplo: partidos políticos.
- Autores - Responsável pelo conteúdo disponível na fonte. Exemplo: proprietários do site.
- Eventos - Acontecimentos importantes no contexto observado. Exemplo: debates.

A partir de instâncias no modelo conceitual contendo as especificações de cada uma das definições listadas acima, são criados os parâmetros de entrada para o arcabouço do sistema. O arcabouço do sistema executa sobre uma arquitetura de software complexa [9], que compõe as fases de execução como mostrado na Figura 2.9.

2.3.2 Arquitura do sistema

A arquitetura do sistema divide-se em cinco fases: Coleta de Dados, Extração e Pré-processamento, Análises, Pós-processamento e Publicação [18], sendo que apenas parte do

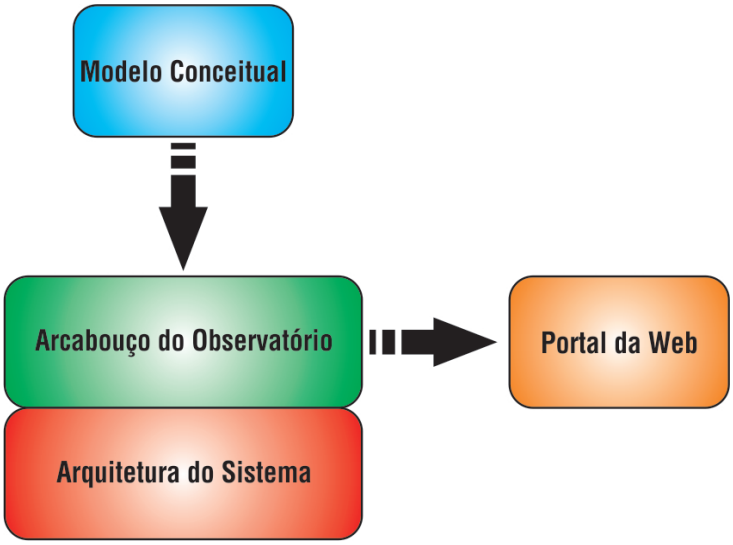


Figura 2.8: Dependências Gerais do Observatório da Web [5]

Figura 2.9: Fases de execução da arquitetura [5]

Pós-processamento e a Publicação não foram avaliadas diretamente neste trabalho, como podemos observar na Figura 2.10.

Na Coleta de Dados, é feito um armazenamento massivo dos dados das diversas fontes da internet (isso no Observatório da Dengue completo. Nesta versão do Observatório da Dengue, a coleta dos dados é feita unicamente do twitter), sendo armazenados ou no banco de dados (MongoDB) ou diretamente na fila inicial criada no RabbitMQ. Na Extração e Pré-processamento, ocorre a extração das notícias e referências; organização e padronização dos dados (como stemming e remoção de stopwords), identificação de idioma e expansão de URLs [18]. Nas Análises, ocorre o agrupamento de notícias, personalidades e fontes; classificação do conteúdo e mineração de padrões frequentes [18].

O pipeline (fase de Extração e Pré-processamento, Análises e parte do Pós-processamento) possui vários filtros criados através do componente *Reader* pelas chamadas ao middleware RabbitMQ [2]. Este middleware cria as filas e as gerencia, recebendo as mensagens passadas de uma fila para outra ou de componentes externos (como um banco de dados) para uma fila. Cada fila criada possui um componente que manipula as mensagens que se encontram no filtro de nome semelhante. Ao final do pipeline, os dados são atualizados no banco de dados MongoDB e os dados referentes às publicações serão armazenados no banco de dados MySQL (ver diagrama de atividades 2.11).

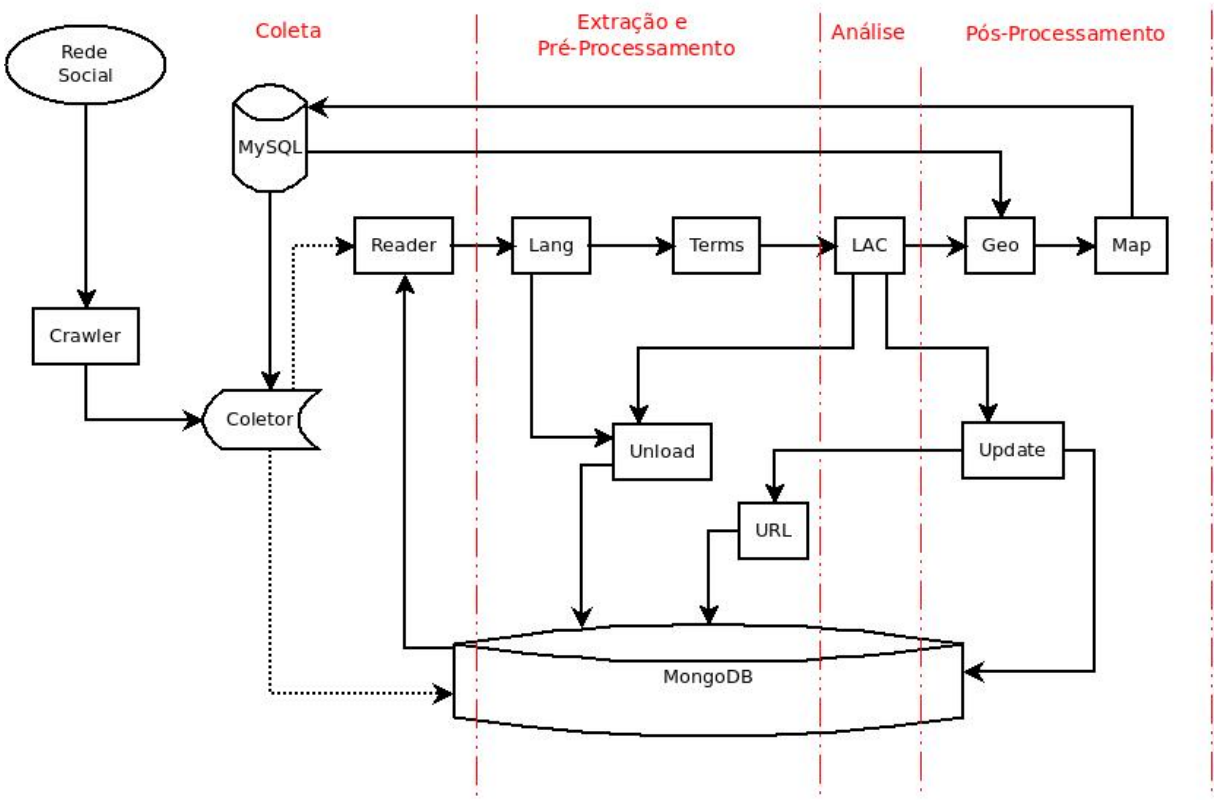


Figura 2.10: Pipeline do Observatório da Dengue

2.3.3 Caracterização dos componentes

O Observatório da Web compreende os seguintes componentes identificados, primeiramente, por Silva e Souza [9]:

- Fontes de Dados - componente externo: RSO. Exemplo: Twitter;
- RabbitMQ - organiza as mensagens em filas, onde cada fila possui um nome semelhante ao componente responsável pelo consumo do conteúdo;
- Banco de Dados MongoDB - faz a armazenagem dos dados processados, podendo fazer a armazenagem da coleta dos dados caso o RabbitMQ fique fora do ar. Esse componente é replicado em virtude da importância que exerce no arcabouço do sistema;
- Componente de filtragem - normaliza os dados coletados e os prepara para os algoritmos de extração;
- Componente de extração de entidades - fazem as operações necessárias para que as técnicas computacionais possam trabalhar;
- Técnicas computacionais - cria indicadores e classificações dos dados a partir de algoritmos;
- Metáforas Visuais - Códigos que permitem a visualização dos resultados obtidos em gráficos e outros recursos. Exemplo: Html, javascript, Flash;

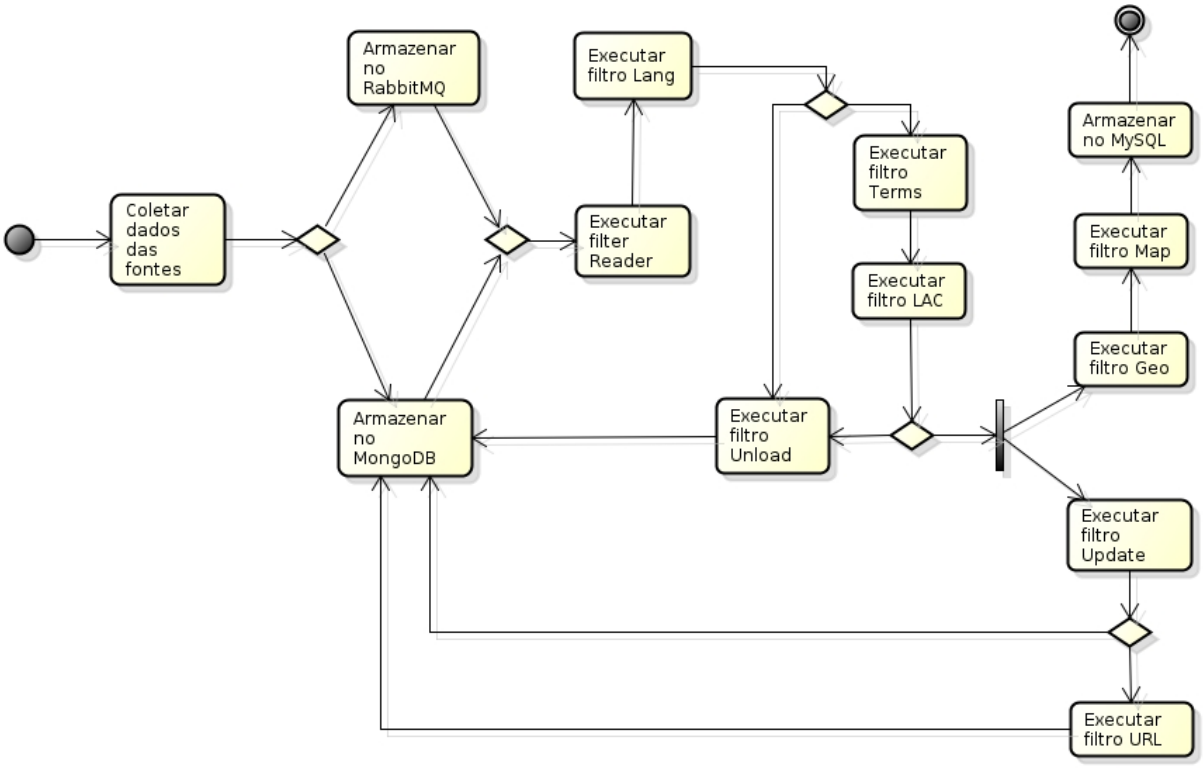


Figura 2.11: Diagrama de atividades do Observatório da Dengue

- Banco de Dados MySQL - faz a armazenagem das informações geradas pelo componente Metáforas Visuais e, se for o caso, possui informações já armazenadas para auxiliar na publicação;
- Servidor Web e Portal. Camada mais externa do Observatório, acessível pelo usuário final.

Componentes identificados na arquitetura do sistema

Após analisar os diagramas de sequência em 6.4.1, foi possível desenhar o diagrama de atividades da Figura 2.11 e o diagrama de componentes da Figura 2.13, cujos componentes são descritos como:

- Crawler - API da fonte de dados que realiza a coleta massiva de dados. Exemplo: Api do Twitter;
- Coletor:

  1. conecta-se ao banco de dados MySQL e pega os dados necessários para a coleta de dados;
  2. verifica o tipo de autenticação das redes sociais (no nosso caso, apenas o Twitter);
  3. verifica o tipo de armazenamento:
     (a) caso o RabbitMQ esteja disponível, o armazenamento será feito diretamente na primeira fila, ou seja, na fila *reader*;

(b) caso o RabbitMQ não esteja disponível, o armazenamento será feito no banco de dados MongoDB.

4. possui a chamada para o método que realiza a coleta massiva de dados pelo *Crawler*.

- Componentes responsáveis pelo processamento das mensagens nos filtros gerenciados pelo RabbitMQ:

1. Reader - componente que cria o workflow no RabbitMQ;
2. Lang - componente que filtra as mensagens a partir da língua que elas representam:
   (a) se ou *pt* ou *es*, então a mensagem será passada para o filtro *terms*;
   (b) caso contrário, a mensagem será encaminhada para o filtro *unload*;
   (c) No período que vai de janeiro à abril, o componente Lang privilegia outras línguas, pois é intensa a circulação de pessoas, como pode ser observado no gráfico da Figura 2.12.
3. Unload - componente que armazena a mensagem no banco de dados MongoDB:
   (a) se a mensagem existe na coleção padrão, ou seja, a coleção que o componente *Reader* acessa, a mensagem é deletada da coleção padrão e armazenada na coleção *unload*;
   (b) caso contrário, a mensagem é armazenada na coleção *unload*.
4. Terms - componente utilizado para marcar textos importantes pré-definidos em um arquivo *.json* usados para a classificação, ou seja, pelo classificador *LAC* [17];
5. Lac - é um componente que faz uso do classificador LAC [17] que foca nas características de uma dada instância de teste, aumentando as chances de gerar mais regras que são úteis para a classificação da instância de teste. Este componente, no Observatório da Dengue:
   (a) remove as stopwords, ou seja, remove palavras irrelevantes para a mineração de dados.
   (b) realiza os treinos.
   (c) se for spam, enviará a mensagem para o filtro *unload*,
   (d) caso contrário, enviará para os filtros *geo* e *update*.
6. Update - componente verifica se as url's já foram expandidas:
   (a) se sim, atualiza o banco de dados MongoDB;
   (b) se não, é uma nova url e a mensagem será enviada para o filtro *url*.
7. URL - componente que realiza a expansão das url's e armazena no bando de dados MongoDB;
8. Geo - é um componente que verifica se existe dados suficientes para identificar a localização de onde a mensagem foi enviada. Se existir:
   (a) acessa o banco de dados MySQL e verifica se existe a localização pré-definida no mesmo;
   (b) caso exista, a mensagem será encaminhada para o filtro *map*.
9. Map - componente que salva as mensagens no banco de dados MySQL;

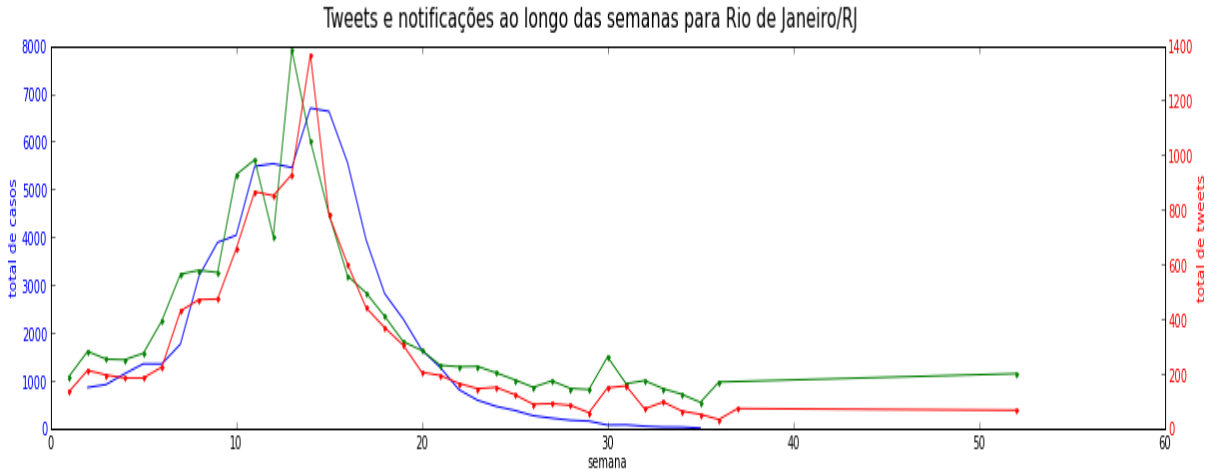


Figura 2.12: Gráfico sobre os casos de dengue no Rio de Janeiro [10]

Gráfico para o Rio de janeiro, onde temos em azul o total de casos, vermelho o total de tweets e em verde a estimativa baseada em dados de 2012.

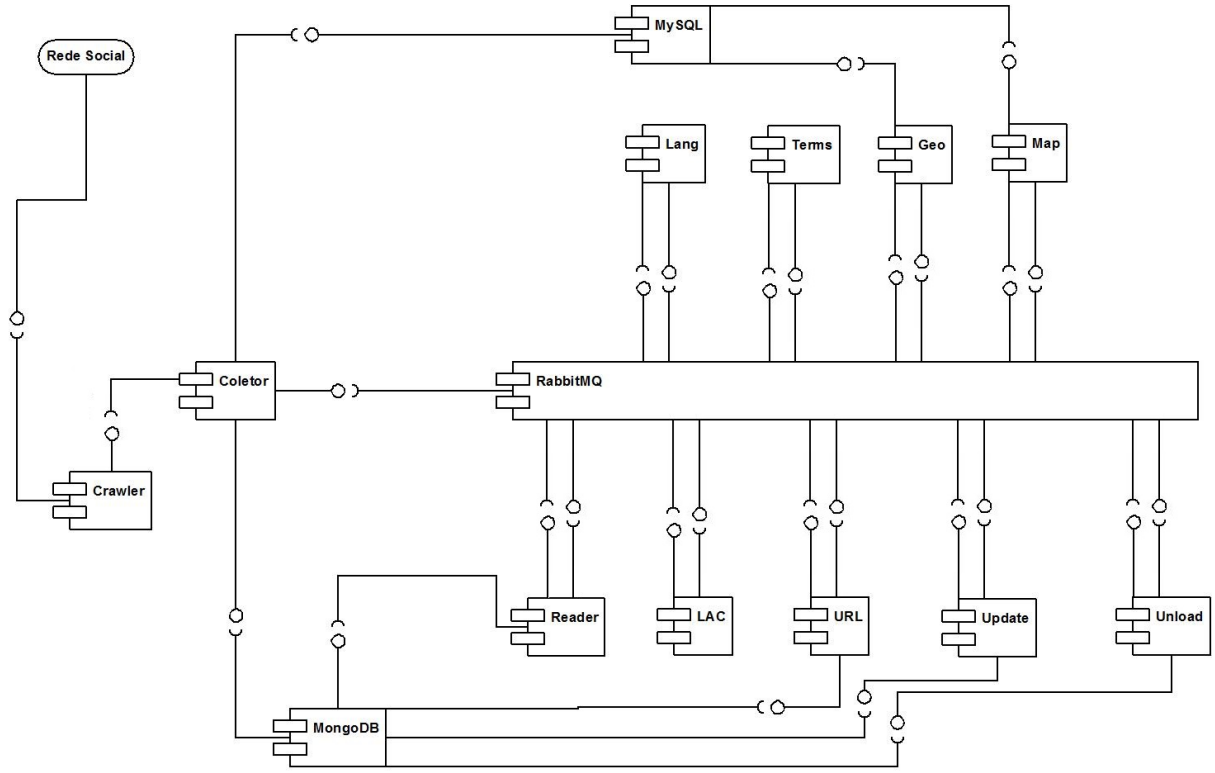


Figura 2.13: Diagrama de componentes do Observatório da Dengue

2.4 Ferramenta de Inspeção

Por que escolhemos esta ferramenta?

Para ser capaz de corrigir um sistema em tempo de execução, um mecanismo que não pode faltar na ferramenta é a introspecção, definida como a capacidade do sistema de conhecer a si mesmo, ou seja, de disponibilizar para consulta informações sobre sua arquitetura, suas funcionalidades e seu estado em cada ponto de uma execução [19]. A Ferramenta de Inspeção [6] possui este mecanismo que procura resolver problemas encontrados *na análise de falhas em logs produzidos com mensagens que descrevem o comportamento do sistema, onde existem muitos logs, informações de diferentes contextos misturados em uma mesma dimensão e logs distribuídos em diferentes máquinas* através de um log em que os eventos são anotados com as informações do contexto em que foram criados, onde cada entidade geradora de log (processo, thread, máquina) notifica seus eventos para um repositório central, que os armazena de forma estruturada. Através de uma interface de consulta, é possível para um usuário estudar o comportamento do sistema filtrando os eventos a serem exibidos segundo um contexto de interesse, contexto este formado por um conjunto de tags que enriquecem a informação do evento e podem ser utilizadas para indicar o que se deseja visualizar [6].

A Ferramenta de Inspeção [6] é um software que permite o desenvolvedor escrever sua própria informação de introspecção, com isso, terá maior eficácia na depuração se a semântica do sistema for incorporada à geração dos seus eventos. A Ferramenta oferece uma interface de consulta 2.14, possuindo filtros seguindo o contexto de interesse do usuário, resolvendo o problema do volume de logs. Com isso, há um grau de flexibilidade alto na consulta, viabilizado pelas propriedades inseridas na etapa de instrumentação (ver 2.4.1) [6].

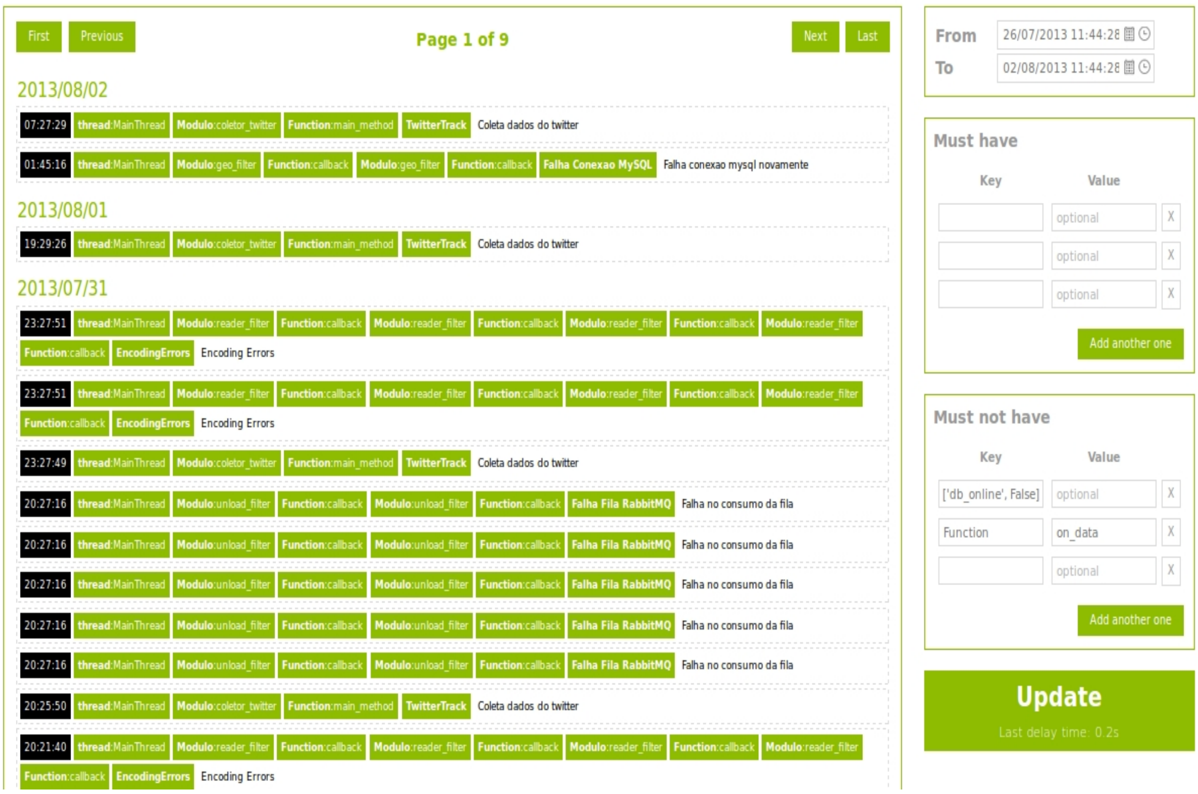


Figura 2.14: Interface de Consulta [6]

De acordo com Sommerville [15], *inspeções de programa* são revisões cujo objetivo é a

detecção de defeitos de programa, podendo ser erros de lógica, anomalias no código que possam indicar uma condição errônea ou não-conformidade aos padrões do projeto ou organizacionais. Inspeções são uma forma de *análise estática*. Porém, com a Ferramenta de Inspeção [6], a coleta dos eventos no código é realizada em *tempo de execução*, permitindo uma *análise dinâmica*.

Na interface de consulta (Figura 2.14) podemos observar no canto superior direito o limite inicial (From) e final (To) de eventos coletados no período delimitado. Logo abaixo da delimitação temporal, existem os filtros classificados como "Must have"e "Must not have"para eventos que queremos analisar e eventos que não queremos analisar, respectivamente.

Na Figura 2.15 abaixo, é obervado os eventos ampliados da Interface de Consulta 2.14, onde no espaço com fundo negro é observado o horário da coleta do evento, *Modulo* significa o componente, *Function* o método e *EncodingErros, TwitterTrack* e *Falha Fila RabbitMQ* são as *tags* escolhidas para informar o significado do evento.

Figura 2.15: Eventos da Interface de Consulta

2.4.1 Instrumentação para coleta de eventos

Um exemplo de como pode ser realizada a instrumentação de um método em um componente do sistema Observatório da Dengue é mostrada a seguir:

Listing 2.1: Inicialização da instrumentação em um método python

```
@scoped_tag('Function','on_data')
def on_data(self, data):
```

E a coleta dos eventos é concretizada inserindo-se o código a seguir na parte onde se deseja coletar eventos do método:

Listing 2.2: Realização para a coleta de eventos

```
logger_instance.message('Erro na Conexao',['Erro','MongoDB','
    Conexao'])
```

Dados os exemplos, é possível observar que a etapa de escrita da instrumentação não oferece muitas dificuldades, porém, qual parte deve ser instrumentada, pode ter um gasto maior se não for realizada durante a fase de desenvolvimento, adicionando que o gasto pode vir a ser ainda maior se for realizada por alguém que não conheça a arquitetura do software.

2.4.2 Arquitetura

A Ferramenta de Inspeção envolve [6]:

- evento representado com uma mensagem, um identificador temporal e um conjunto de tags;
- cada tag é representada por um par nome-valor, representando as propriedades contextuais do sistema no instante em que o evento foi criado;
- valor de cada tag é opcional, dependendo de sua natureza;
- arquitetura da solução com explicação dividida em três partes:
    1. escrita das informações de introspecção;
    2. fluxo dos eventos;
    3. ferramenta para inspecionar o comportamento do sistema.

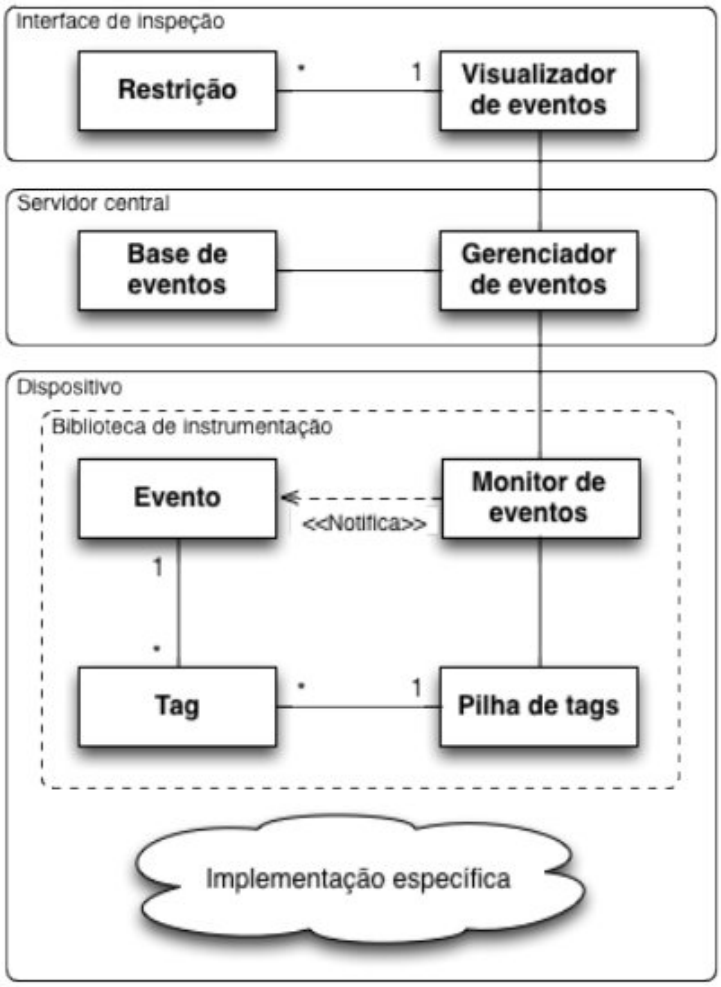


Figura 2.16: Arquitetura da Solução [6]

2.5 Ferramenta de Monitoramento

Existem sistemas de monitoramento que atuam em diversos níveis da computação, desde simples algoritmos até monitoramento de grandes servidores de redes. Mas focaremos o uso do monitoramento no nível infra-estrutural que o Observatório da Dengue está executando.

Monitorar um sistema complexo com diversos componentes é um método eficaz para que o administrador possa examinar relatórios e gráficos gerados pela ferramenta de monitoramento. A ferramenta de monitoramento coleta informações de uso de hardware(processamento, uso de memórias e etc), fluxos de dados(interface de rede) e eventos através de um *daemon* instalado no ambiente alvo. Esses dados coletados são armazenados e sintetizados no servidor da ferramenta. Os administradores então analisam as informações para agir sobre os erros e defeitos, também é possível de receber notificações em tempo real com o uso de *triggers* configuráveis, possibilitando que os administradores possam atuar rapidamente e diretamente sobre os erros e defeitos.

Considerando falhas operacionais, isto é, falhas que tornam o sistema inoperante, utilizamos a ferramenta de monitoramentos *Zabbix* [1] para monitorar a arquitetura física em que o Observatório da Dengue está sendo executado. Essas falhas podem ser, por exemplo, falhas de interferência e inconsistência na rede de comunicação e falhas na operação correta do hardware no servidor.

Para melhor entender o termos do que usaremos para descrever o *Zabbix*, define-se:

1. ***Triggers***: Uma função que é disparada quando a expressão sobre um determinado componente é alcançada.

   Exemplo: Caso o espaço na partição / seja menor que 200mb, envie um aviso.

2. **Evento:** No *Zabbix*, evento é uma descrição de um acontecimento previamente configurado pela *trigger*.

O ambiente de monitoramento usado, o *Zabbix* [1], é composto dos seguintes componentes:

Zabbix Server: é a central onde são armazenadas todas as informações enviadas pelos agentes do zabbix. Também armazena todas as estatísticas, relatórios, configurações e dados.

Zabbix Agent: são os coletores instalados em cada computador alvo, que enviam os dados solicitados.

Zabbix Frontend: é a interface gráfica para facilitar o uso e leitura do servidor do Zabbix, normalmente fica instalado no mesmo computador.

Existem muitas características comuns para monitorar e configurar no *Zabbix* [1], entre elas:

1. Extrair dados para monitoramento usando o agente do *Zabbix* remotamente no ambiente que faz conexão com o servidor usando protocolo TCP/IP pela porta 27050;

2. Monitorar fluxo de redes, disponibilidade de um ip ou porta que é usada para a transmissão de dados em modo geral, como por exemplo, a transferência de dados do RSO, comunicação entre componentes;

3. Monitorar classes de serviços de rede, usando uma arvore para visualizar quais eventos são para monitorar, podemos monitorar e classificar a gravidade de falhas;

4. Monitorar uso do armazenamento local, podendo indicar alertas quando o disco está cheio, e também criar *triggers* para remediar situações de risco;
5. Monitorar o desempenho da CPU, podendo alterar a forma de visualização gráfica, criar *triggers* quando certos valores forem atingidos e alertar estados críticos;
6. Classificar eventos em hora do evento, localização, descrição estado, gravidade e duração.

Como o foco do *Zabbix* é a geração de eventos, um evento deve conter valores sobre hora de criação, descrição, gravidade e campo de estado.

Os campos básicos de eventos são:

1. hora do evento;
2. Localização;
3. Descrição;
4. Estado do evento;
5. Gravidade;
6. Duração;
7. id (ID único do evento).

Sendo que cada campo de um evento é descrito com base em uma tabela de definição.

Tabela 2.1: Campo do estado do evento

Número	**Nome**
OK	Indica que o problema não existe mais ou foi resolvido
PROBLEM	Problema ainda está ocorrendo
UNKNOWN	Não foi possível obter o estado do evento*

*Os eventos com estado desconhecidos querem dizer que o *Zabbix* não conseguiu avaliar a expressão contida nos *triggers* e pode ter sido causada por diversas razões:

1. Servidor está fora de alcance;
2. A expressão do *trigger* não pode ser avaliada;
3. A expressão do *trigger* teve uma mudança recente.

Tabela 2.2: Campo de Gravidade do evento

Cor	**Descrição ou Gravidade**
Azul claro	Informação
Verde	Problema está resolvido
Amarelo	Aviso
Vermelho	Gravidade média

Usando o módulo de visualização e gerenciamento de eventos do *Zabbix*, temos o painel de ferramentas na Figura 4.11), podemos fazer consultas e selecionar apenas eventos interessantes para coletar.

Time	Description	Status	Severity	Duration
Sep 2nd, 2013 03:22:12 PM	Disk I/O is overloaded on ObsWeb	OK	Warning	14d 9h 57m
Sep 2nd, 2013 03:20:12 PM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	2m
Aug 30th, 2013 08:03:07 PM	Processor load is too high on ObsWeb	OK	Warning	23d 15h 45m
Aug 30th, 2013 08:02:07 PM	Processor load is too high on ObsWeb	PROBLEM	Warning	1m
Aug 29th, 2013 05:20:17 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	4d 20h 33m
Aug 29th, 2013 05:18:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	PROBLEM	Average	2m 17s
Aug 29th, 2013 03:51:58 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	1h 26m 2s
Aug 29th, 2013 03:47:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	PROBLEM	Average	4m 58s
Aug 26th, 2013 08:17:12 PM	Disk I/O is overloaded on ObsWeb	OK	Warning	6d 19h 3m
Aug 26th, 2013 08:16:12 PM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Aug 26th, 2013 04:09:24 PM	ObsWeb has just been restarted	OK	Information	27d 19h 39m

Figura 2.17: Painel de módulo de visualização de eventos do *Zabbix*.

Temos as seguintes funcionalidade do *framework* de eventos do *Zabbix* [1]:

- Selecionar e filtrar eventos;
- Trabalhar com pesquisa em tempo real;
- Atualizar a visualização;
- Visualizar detalhes do evento;
- Conhecer eventos;
- Retornar eventos para um estado novo;
- Classificar eventos;
- Exportar em formato csv os evento selecionados;
- Criar eventos.

Com base neste referencial teórico, e conhecimentos diversos de Ciência da Computação, foi construída a metodologia que será abordada no próximo capítulo.

Capítulo 3

Metodologia

Foi considerado apenas o período de coleta de eventos do dia 17/08/2013 até 02/09/2013, começando e encerrando ao meio dia, respectivamente.

3.1 Funcional

A instrumentação com a Ferramenta de Inspeção [6] não ofereceu muitas dificuldades, pois possui uma semântica de fácil entendimento, como mostrado nos conceitos básicos deste trabalho. Porém, a instrumentação do módulo disponiblizado para estudo do Observatório da Dengue ocorreu após seu desenvolvimento e por uma pessoa desacoplada da equipe que o desenvolveu. Pode-se observar na Figura 3.1 os passos seguidos para manutenção preventiva no Observatório da Dengue.

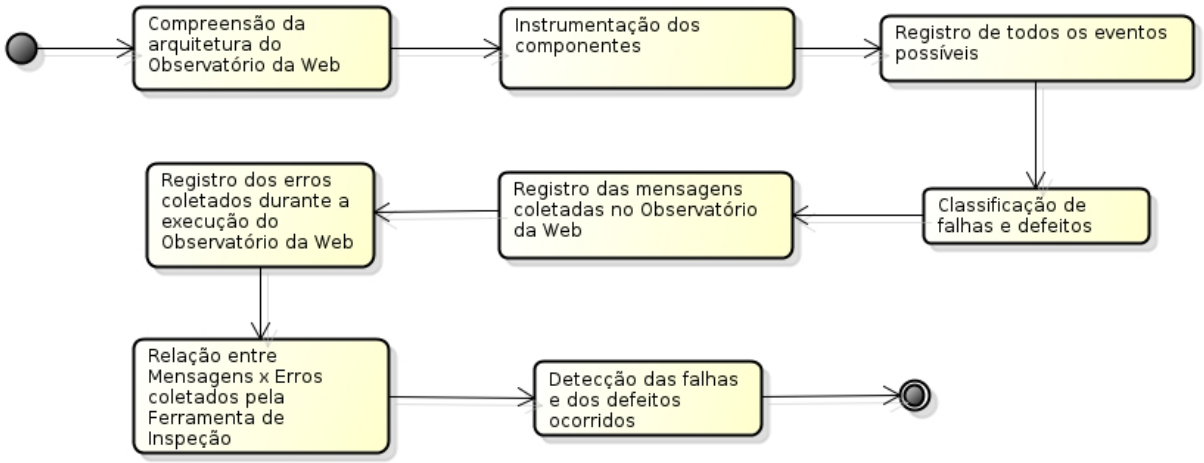


Figura 3.1: Metodologia para manutenção preventiva no Observatório da Dengue

3.1.1 Compreensão e instrumentação do Observatório da Dengue

Foi realizada uma especificação das funcionalidades de cada componente da versão parcial do Observatório da Dengue. A partir desta especificação, foram modelados diagramas de

sequência 6.4.1 para descrever o funcionamento de entrada e saída das mensagens dos componentes.

É possível observar no diagrama da Figura 6.4 o componente *Lang* conectando-se ao middleware RabbitMQ. Após a conexão, o componente solicita a mensagem do filtro *lang* para processá-la e enviá-la ao próximo filtro. Nesse componente, é verificado se a língua é portuguesa, espanhola ou outra qualquer. Caso seja portuguesa ou espanhola, a mensagem será direcionada para o filtro *terms*, caso contrário, será direcionada ao filtro *unload*. Todos os filtros possuem o mesmo objetivo que é o de armazenar as mensagens recebidas por filtros anteriores e serem direcionadas para o filtro seguinte depois de processadas pelo componente de nome semelhante, salvo as situações que a mensagem segue para o banco de dados ou vem do mesmo. Desse mesmo diagrama é possível identificar pontos para instrumentação, como no método que faz a conexão com o middleware. Análises semelhantes foram realizadas com o coletor e com os outros componentes envolvidos diretamente com pelo menos um dos filtros.

Após essa instrumentação foi modelado, a partir dos diagramas de sequência, um diagrama (ver Figura 2.10) que se aproxima do funcionamento do módulo disponibilizado do Observatório da Dengue que auxiliou na abstração de seu funcionamento. A partir do diagrama, uma revisão de cada componente deste sistema foi realizada, porém sendo criado, de forma sistemática, um documento com os registros de quais arquivos e métodos foram instrumentados e que veio a ser, posteriormente, utilizado para a classificação das falhas.

Dado o potencial e a flexibilidade de instrumentação proporcionada pela Ferramenta de Inspeção [6], observa-se que é possível monitorar os componentes do Observatório em vários níveis de granularidade, porém há uma limitação de tempo. Foi realizado o monitoramento do tempo de uma mensagem para outra, a quantidade de mensagens que passam para determinados filtros e o tipo da mesma. Estes objetivos foram alcançados com instrumentações em todos os componentes do pipeline, como mostrado no diagrama de componentes da Figura 2.13.

3.1.2 Registro dos eventos e classificação das falhas e dos defeitos

Após definidas todas as instrumentações 6.5.1, foram classificadas algumas falhas 6.5.3 e alguns defeitos 6.5.4 possíveis de ocorrerem no sistema a partir dos erros 6.5.2.

3.1.3 Registro e relação dos eventos coletados

Dentro do período observado, foi realizada a contagem de eventos em relação às mensagens que entravam e saiam de cada componente e os erros ativados em cada um deles 6.6 e realizada uma comparação entre elas 6.6.1.

3.1.4 Avaliação de dependabilidade

Durante a instrumentação do Observatório da Dengue, era verificado, paralelamente, qual seria o melhor método para análise de falhas e defeitos que poderiam ocorrer no sistema. Então, tendo como base [8] e [7], foi decidido utilizar as classificações de falhas da Figura 2.4 e de defeitos da Figura 2.6. Analisando a arquitetura do Observatório mostrada na Figura 2.10, foram classificadas as possíveis falhas que poderiam ocorrer.

Durante a classificação das falhas, foi observado que era necessário monitorar a quantidade de mensagens que passavam de um filtro para outro, pois a coleta é realizada sem distinções

de dados e os mesmos são filtrados apenas no pipeline, ou seja, a partir do processo *Reader*. Surgiram algumas dúvidas se realmente era viável, pois haveriam muitos eventos, até então considerados desnecessários para a análise de falhas, na interface de consulta da Ferramenta de Inspeção [6].

Para a solução deste problema, padronizamos todas as instrumentações que seriam responsáveis pela quantidade de mensagens que seriam processadas por cada um dos componentes através da *tag* **counter**. Com isso, quando for preciso eliminar todas as mensagens que estão sendo contadas, será necessário apenas digitar no campo *Must not have* a palavra *counter* e não aparecerá nenhum evento com esta *tag*. Para os erros foram realizadas instrumentações semelhantes, sendo assim, foi colocada a *tag* **erro** em cada instrumentação que sinaliza um aviso de erro nos componentes.

Após esta instrumentação começamos a analisar cada componente individualmente, pois cada um seria um *sistema independente* e todos os outros sistemas seriam o *ambiente* deste sistema. Os *defeitos* em cada componente seriam originados em um *estado interno* do *sistema provedor*, que teria o *estado externo* deste sistema sendo percebido na *interface de usuário* do *ambiente*. Para a análise de falhas esse olhar é fundamental, pois a *falha* tem origem em um *estado interno* do *provedor do serviço*, *ativando* um *erro* que se *propaga* internamente até atingir um *estado externo*, alcançando a *interface de serviço* deste sistema e sendo o *defeito* percebido na *interface do usuário*, ou seja, na interface do sistema que recebe o serviço. Dado esse esquema observa-se que, se este *defeito* alcançar a *interface de serviço* do sistema que recebeu o *serviço defeituoso*, a mesma análise pode ser feita para o componente seguinte. Neste caso, observamos três sistemas: o **A**, onde ser originou a *falha*; o **B** que recebeu um *serviço defeituoso* de **A**; e o **C**, que recebeu o *serviço defeituoso* de **B** que teve uma *falha ativada* pelo serviço de **A**, ou seja, o *serviço defeituoso* de **A** é a *causação* da *falha* em **B** que *ativa* um *erro* e se *propaga* para a *interface do usuário* **C**.

3.2 Operacional

Para a aplicação da ferramenta, foi usado a metodologia 3.2, as etapas de instalação e configuração de todo ambiente de monitoramento **Zabbix** encontram-se no apêndice deste trabalho.

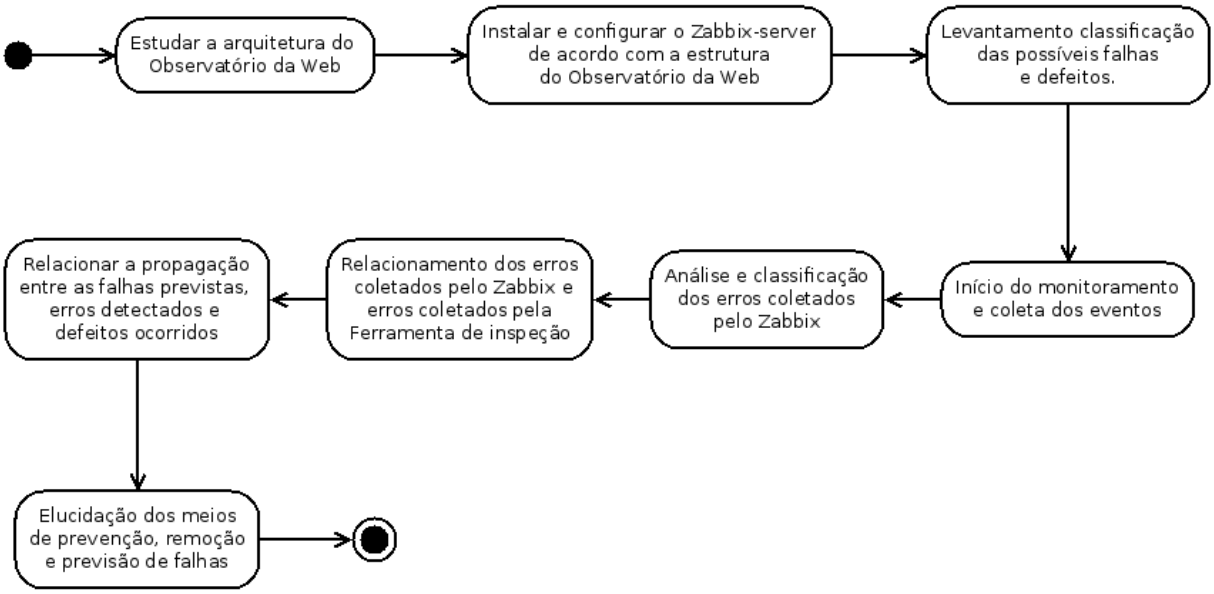


Figura 3.2: Metodologia para análise operacional do Observatório da Dengue.

O servidor do *zabbix* encontra-se na UnB e comunica com o *zabbix-agent* instalado no mesmo ambiente do Observatório da Dengue que está na UFMG 3.3.

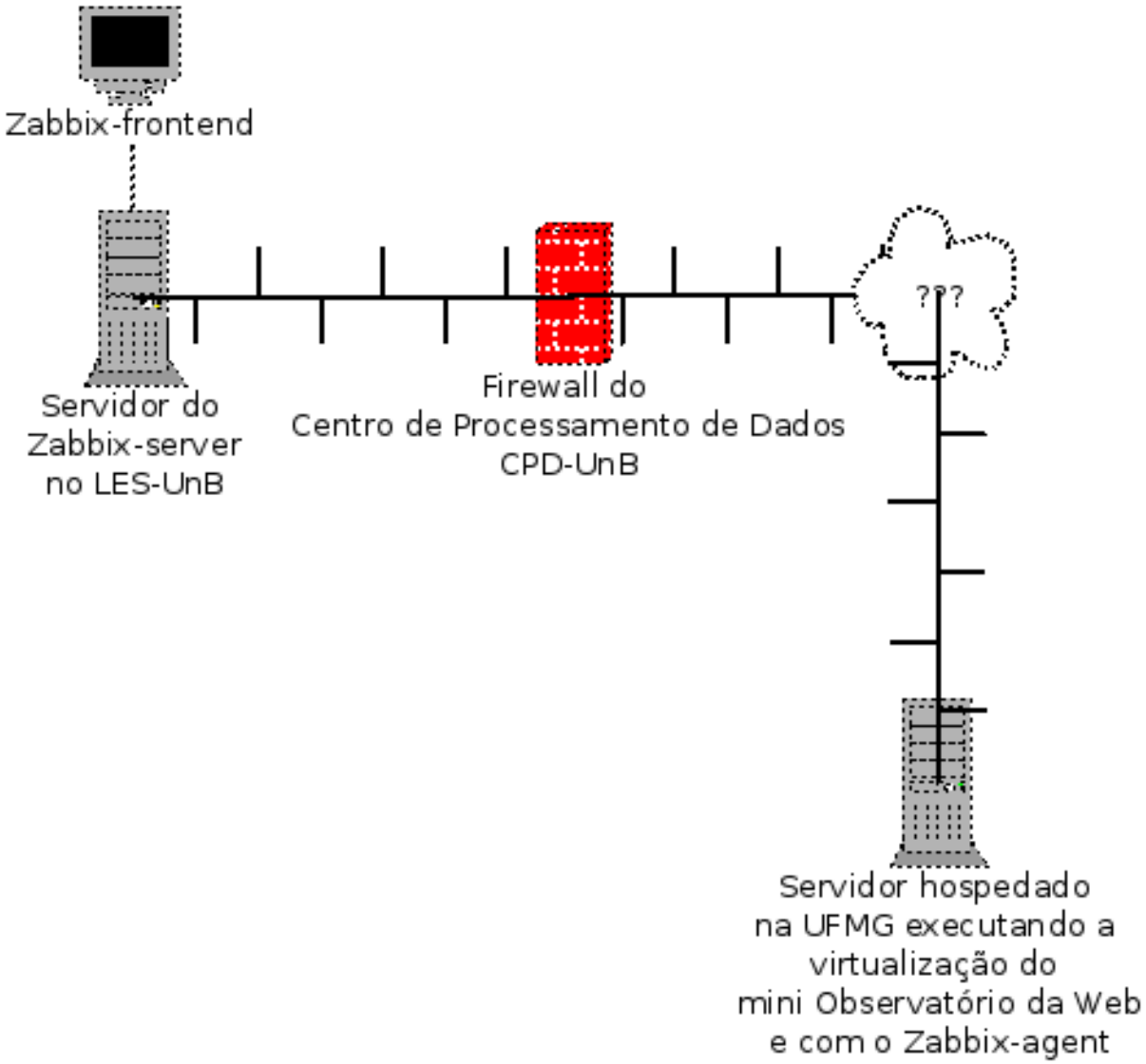


Figura 3.3: Topologia da rede de comunicação entre os servidores.

3.2.1 Passos usados para o entendimento de falhas de hardware

Para alcançar um sistema com os atributos de dependabilidade desejados, podemos usar técnicas de monitoramento de sistemas para detecção, validação e análise do comportamento do sistema para prevenção de falhas no nível de infra-estrutura em que o Observatório da Dengue está instalado. Com o intuito de alcançarmos a maior parte de possíveis defeitos, temos que verificar falhas operacionais que ocorrem na infra-estrutura, tanto em Hardware quanto no Sistema Operacional. Para isso, precisamos definir e classificar quais são os defeitos de Hardware e de Sistema Operacional que podem ter originado da infra-estrutura do Observatório da Dengue.

A distinção de falhas e defeitos, descritos em [8] ajuda a conseguir mais facilmente relacionar às faltas que levaram ao estado do defeito. Aqui abordaremos uma visão *Top Down*. Primeiramente são classificados os defeitos, as falhas e as possíveis relação entre elas com os erros detectáveis pelo *Zabbix* [1].

Defeitos mapeados

Tabela 3.1: Defeitos mapeados

	Domínio	Consistência	Consequência
Defeito na conexão com o servidor remotamente	Defeito de parada	Consistente/Inconsistente	Pequena
Defeito de processo não respondente	Defeito de parada	Consistente	Média
Defeito de sistema não respondente	Defeito de parada	Consistente	Grave
Defeito na conexão com o *Zabbix server*	Defeito de parada	Inconsistente	Pequena
Defeito de conexão na interface de rede	Defeito de parada	Consistente/Inconsistente	Média
Impossibilidade de gravar novos arquivos	Defeito de parada	Consistente	Média
Impossibilidade de adicionar dados ao banco	Defeito de parada	Consistente	Média

Falhas mapeadas

A identificação das classes das falhas, ajuda a abstrair o problema, identificar os padrões de reações e também organizar as novas falhas para facilitar a identificar e tratar as falhas.

Tabela 3.2: Falhas mapeadas

	Fase de criação	Localização	Causa	Dimensão	Objetivos	Intenção	Capacidade	Persistência
Excesso de programas em execução	de desenvolvimento	Interna	Humana	Software	Não maliciosa	Não-deliberada	Incompetência	Transitória
Buffer de programa excessivamente grande	de desenvolvimento	Interna	Humana	Software	Não maliciosa	Não-deliberada	Incompetência	Transitória
Cabo de conexão de interface de rede desconectado ou rompido	Operacional	Externa	Humana ou natural	Hardware	Não maliciosa	Deliberada	Incompetência ou acidental	Persistente
Problema no carregamento de driver para a interface de rede	Operacional	Interna	Humana	Software	Não maliciosa	Não-deliberada	Incompetência ou acidental	Persistente
Bloqueio de portas ou protocolos por firewalls	Operacional	Externa	Humana	Software	Não maliciosa	Deliberada	Incompetência	Persistente
Defeito físicos no disco rígido	Operacional	Externa	Humana	Hardware	Não maliciosa	Não-deliberada	Acidental	Persistente
Elevado número de requisições de acesso à memória por processos	Operacional	Interna	Humana	Software	Não maliciosa	Não-deliberada ou deliberada	Incompetência ou acidental	Transitória
Um ou mais processos executam por muito tempo	Operacional	Interna	Humana	Software	Não maliciosa	Não-deliberada ou deliberada	Incompetência ou acidental	Transitória
Queda de energia do servidor no Observatório da Dengue	Operacional	Externa	Humana ou natural	Hardware	Não maliciosa	Não-deliberada	Acidental	Transitória
Administrador do sistema reinicia para manutenções	Operacional	Externa	Humana	Software	Não maliciosa	Não-deliberada	Acidental	Transitória
Excesso de dados no banco de dados	Operacional	Interna	Humana	Software	Não maliciosa	Não-deliberada		Persistente
Excesso de arquivos grandes em disco	Operacional	Interna	Humana	Software	Não maliciosa	Não-deliberada		Persistente

Em ponto de vista operacional, o *Zabbix* [1] monitora elementos físicos e alguns serviços em nível da arquitetura do sistema que podem ocorrer erros. Os erros são obtidos de acordo com a execução das expressões descritas nos *triggers* 3.4. As possíveis relações de propagação de falhas em erros e de erros em defeitos está descrito a seguir:

Severity	Status	Info	Last change	Age	Duration	Acknowledged	Host	Name
Average	OK		Sep 21st, 2013 02:50:00 PM	1d 20h 14m		Acknowledge (23)	ObsWeb	Zabbix agent on ObsWeb is unreachable for 5 minutes {ObsWeb:agent.ping.**nodata**(5m)}=1
Warning	OK		Sep 17th, 2013 06:44:12 AM	6d 4h 20m		Acknowledge (9)	ObsWeb	Disk I/O is overloaded on ObsWeb {ObsWeb:system.cpu.util[,iowait].**last**(0)}>20
Warning	OK		Aug 30th, 2013 08:03:07 PM	23d 15h 1m		Acknowledge (1)	ObsWeb	Processor load is too high on ObsWeb {ObsWeb:system.cpu.load[percpu,avg1].**last**(0)}>5
Information	OK		Aug 26th, 2013 04:09:24 PM	27d 18h 55m		Acknowledge (5)	ObsWeb	ObsWeb has just been restarted {ObsWeb:system.uptime.**change**(0)}<0
Information	OK		Aug 8th, 2013 03:29:24 PM	1m 15d 19h		Acknowledge (1)	ObsWeb	Host information was changed on ObsWeb {ObsWeb:system.uname.**diff**(0)}>0
Warning	OK		Jul 6th, 2013 04:29:26 PM	2m 18d 18h		Acknowledge (1)	ObsWeb	/etc/passwd has been changed on ObsWeb {ObsWeb:vfs.file.cksum[/etc/passwd].**diff**(0)}>0
Information	OK		Jul 6th, 2013 04:29:18 PM	2m 18d 18h		Acknowledge (1)	ObsWeb	Hostname was changed on ObsWeb {ObsWeb:system.hostname.**diff**(0)}>0
Average	OK		Never	43y 9m 6d		No events	ObsWeb	Lack of available memory on server ObsWeb {ObsWeb:vm.memory.size[available].**last**(0)}<20M
Information	OK		Never	43y 9m 6d		No events	ObsWeb	Configured max number of processes is too low on ObsWeb {ObsWeb:kernel.maxproc.**last**(0)}<256

Figura 3.4: Relatório para a análise das ocorrências de determinados *triggers* detalhadamente.

Tabela 3.3: Relação entre falhas, erros(escritos em inglês) e defeitos

Falhas	**Defeitos**					
	Defeito na conexão com o servidor remotamente	**Defeito de processo não respondente**	**Defeito de sistema não respondente**	**Defeito na conexão com o *Zabbix server***	**Defeito de conexão na interface de rede**	**Impossibilidade de gravar novos dados**
Excesso de programas em execução		Lack of available memory on server	Lack of available memory on server			
		Lack of free swap space	Lack of free swap space			
Buffer de programa excessivamente grande		Lack of available memory on server	Lack of available memory on server			
Cabo de conexão de interface de rede desconectado ou rompido	Zabbix agent is unreachable for 5 minutes			Zabbix agent is unreachable for 5 minutes	Zabbix agent is unreachable for 5 minutes	
Problema no carregamento de driver para a interface de rede	Zabbix agent is unreachable for 5 minutes			Zabbix agent is unreachable for 5 minutes	Zabbix agent is unreachable for 5 minutes	
Bloqueio de portas ou protocolos por firewalls	Zabbix agent is unreachable for 5 minutes			Zabbix agent is unreachable for 5 minutes	Zabbix agent is unreachable for 5 minutes	
Defeito físicos no disco rígido		Disk I/O is overloaded	Disk I/O is overloaded			
Elevado número de requisições de acesso à memória por processos		Disk I/O is overloaded	Disk I/O is overloaded			
Um ou mais processos executam por muito tempo		Processor load is too high	Processor load is too high			
Queda de energia do servidor no Observatório da Dengue						
Excesso de dados no banco de dados		Free disk space is less than 20 % on partition /				Free disk space is less than 20 % on partition /
Excesso de arquivos grandes em disco		Free disk space is less than 20 % on partition /				Free disk space is less than 20 % on partition /

Capítulo 4

Resultados e Discussão

4.1 Resultados

4.1.1 Funcional

Tempo de coleta

Uma mensagem era coletada a cada 00:01:03 (um minuto e três segundos). Isso começou a ser observado depois que ocorreu uma mudança no Observatório da Dengue na estrutura do processo Coletor_twitter. Porém, o tempo entre uma mensagem e outra chega a diminuir entre 1 e 20 segundos. Ou seja, não foi identificado que uma nova mensagem só entra no pipeline após o término do processamento da mensagem que chegou antes, pois eram esperadas 21942 mensagens para 63 segundos, porém foram registradas 23875 mensagens. Podemos observar na Tabela 4.1 a quantidade de mensagens recebidas e enviadas pelos componentes através dos filtros gerenciados pelo *middleware* RabbitMQ [2]. Na Tabela 4.2 podemos visualizar a quantidade de defeitos diferentes que ocorreram em cada componente sem contabilizar a quantidade de vezes que cada um ocorreu e sem contabilizar a quantidade de propagações de erros.

- Mensagens para publicação:
  - lang para term: 19,52%.
  - lac para map: 58,81%.
  - lac para geo: 98,73%.
  - geo para map: 59,57%.
  - mensagens totais para map: 8,50%.
  - recebidas pelo RabbitMQ para map: 11,48%.
- Perda de mensagens no componente Geo:
  - Localização vazia: 30,87%.
  - Localização existente no MySQL: 59,57%.
  - Localização inexistente no MySQL: 9,57%.
  - Perda do RabbitMQ para map: 1,84%

Tabela 4.1: Total de mensagens por módulo

Filtro	Mensagens Recebidas	Destino	Mensagens Enviadas
coletor	23875	RabbitMQ MongoDB	17680 6195
reader	17681	lang	17681
lang	17681	terms unload	3452 14229
terms	3452	lac	3452
lac	3452	geo e update unload	3405 47
geo	3428	map	2030
update	3408	url MongoDB	754 2654
map	2030	MySQL	2030
uRL	218791	MongoDB	755
unload	14283	MongoDB	14283

Tabela 4.2: Totais de Mensagens, Erros e Defeitos

Componente	Total de Mensagens	Total de Erros	Defeitos
Coletor	23875	234	4
Reader	17681	450	3
Geo	3428	284	1
URL	755	13	3
Unload	14283	7	1

- Com as localizações no MySQL:
  - RabbitMQ para map: 13,32%.

Resultados identificados e obtidos

Nos gráficos, **Identificadas** significa que representa as falhas que foram classificadas e **Não identificadas** significa a união das falhas que foram classificadas mais a quantidade de erros dos quais não foram possíveis a identificação de falhas. Considerando que o componente **Url_filter** produziu apenas uma mensagem de erro (*erros url limitado* na legenda do gráfico das Figuras 4.5 e 4.8) para *RabbitMQ Fila Consumo*

O gráfico da Figura 4.1 apresenta todos os componentes que apresentaram erros dentro do período delimitado e a quantidade total de erros em cada um deles, indicando a porcentagem de mensagens geradas de cada erro.

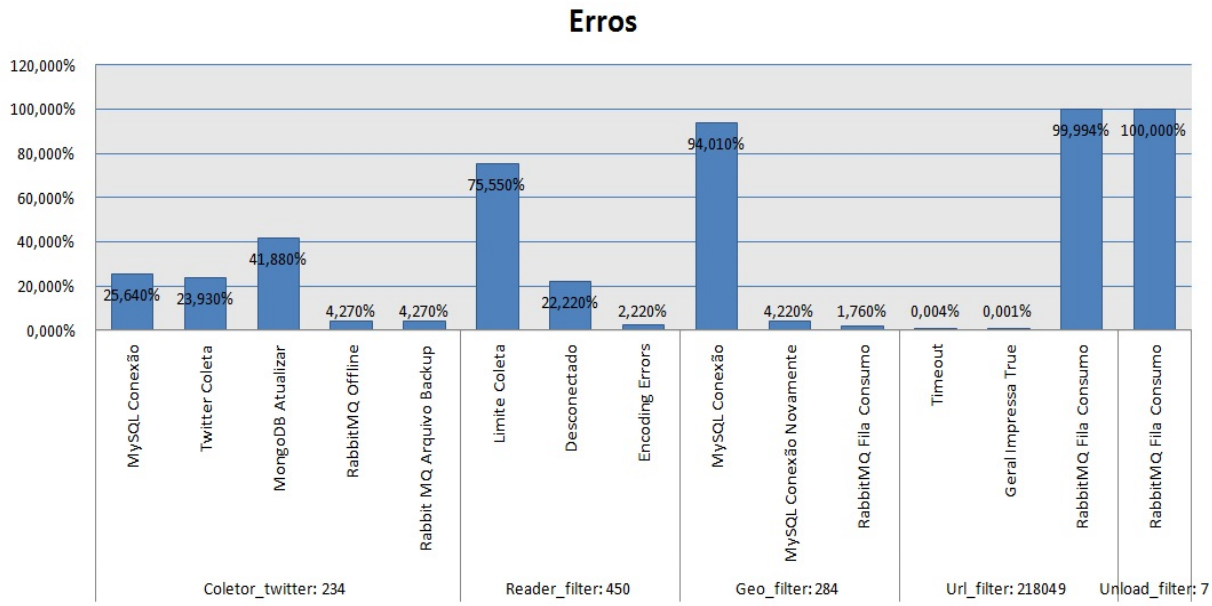


Figura 4.1: Erros encontrados nos componentes

Os gráficos das Figuras 4.2 e 4.3 possuem a porcentagem de cada atributo em relação à quantidade total de falhas identificadas, ou seja, que podem ocorrer no sistema. O gráfico da Figura 4.2 considera apenas as falhas que poderiam gerar erros diferentes em um mesmo componente, enquanto o gráfico da Figura 4.3 considera todas as falhas que poderiam gerar o mesmo erro em um mesmo componente. Foram classificadas mais falhas Operacional, Externa, Humana, Software/Hardware e Transitória, como observado no gráfico da Figura 4.3.

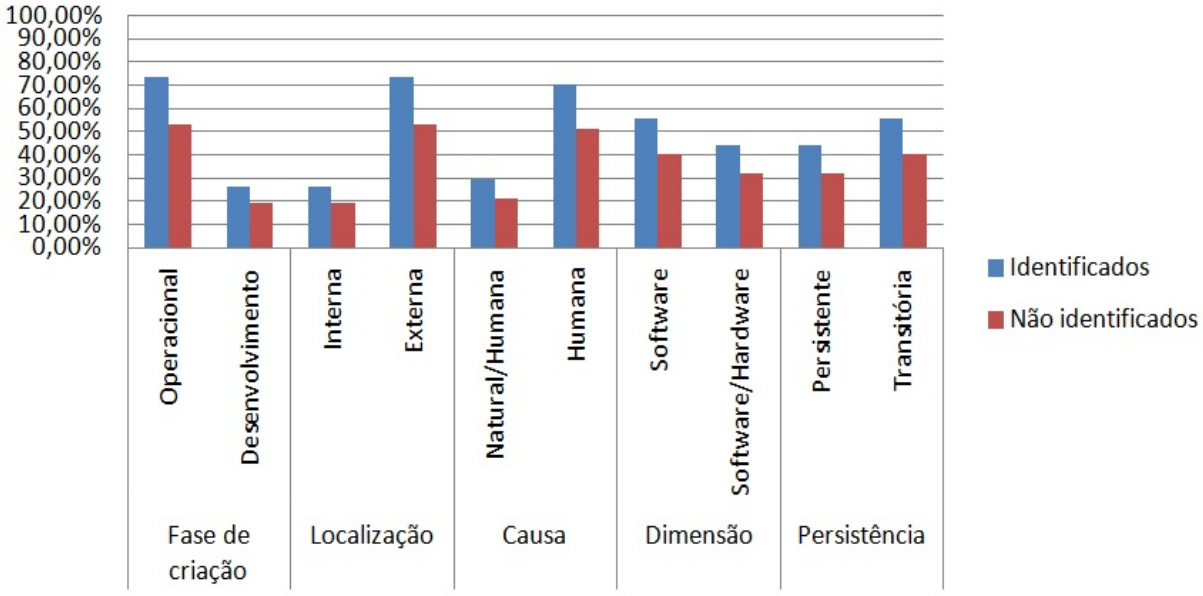


Figura 4.2: Classificação das falhas sem repetição

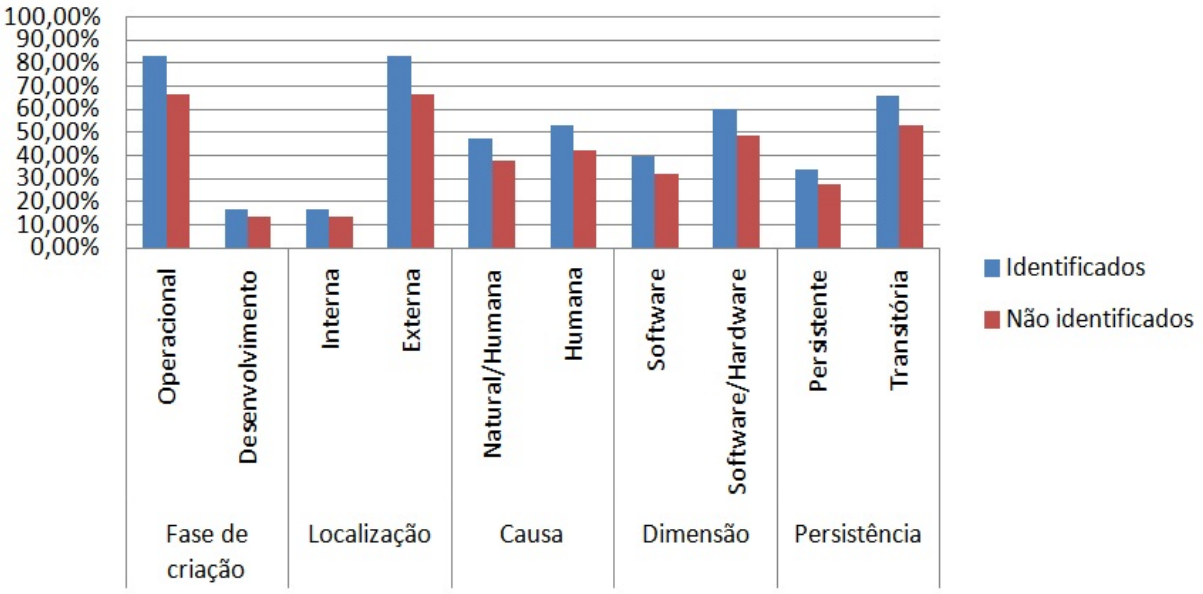


Figura 4.3: Classificação das falhas com repetição

O gráfico da Figura 4.4 mostra a porcentagem da classificação das falhas ocorridas de forma unitária, ou seja, sem considerar a quantidade de vezes que podem ter ocorrido. Podemos observar que houve uma maior ocorrência de falhas Operacional, Externa, Natural/Humana, Software/Hardware e Persistente, diferenciando-se do gráfico da Figura 4.3 pelo aumento do número de falhas de Desenvolvimento, fazendo aumentar o número de falhas Persistente.

Já o gráfico da Figura 4.5 mostra a porcentagem da classificação das falhas considerando a quantidade de vezes que elas podem ter ocorrido durante a execução da versão parcial do Obser-

vatório da Dengue no período delimitado. Podemos observar que houve uma maior ocorrência de falhas de Desenvolvimento, Interna, Humana, Software e Persistente, mostrando o impacto que tem o fato das falhas ocorrerem no Desenvolvimento para as outras classificações.

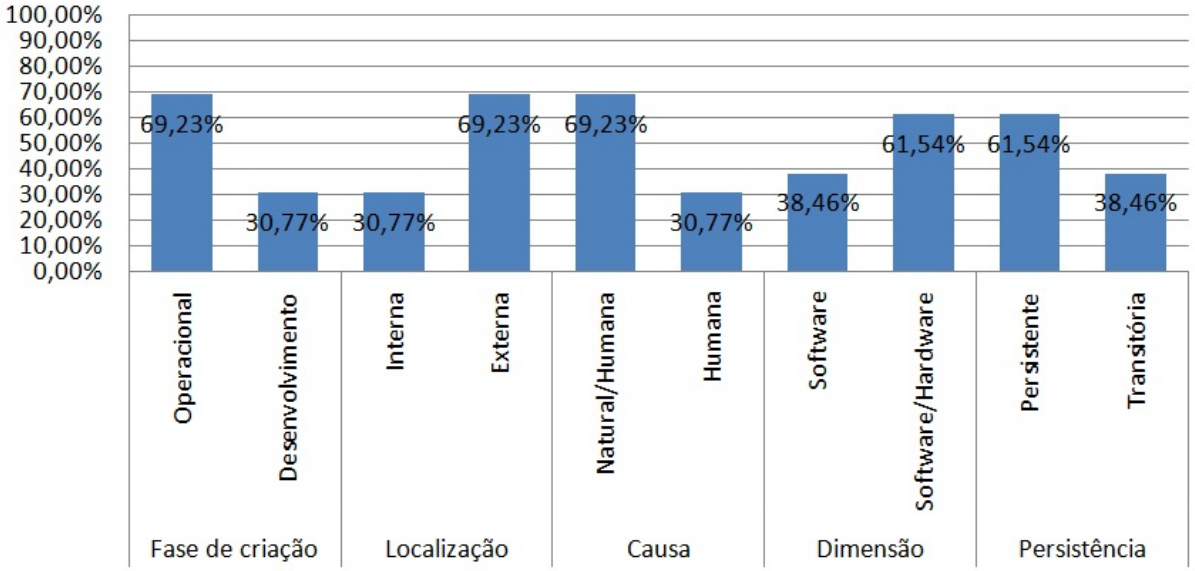


Figura 4.4: Classificação unitária das falhas ocorridas

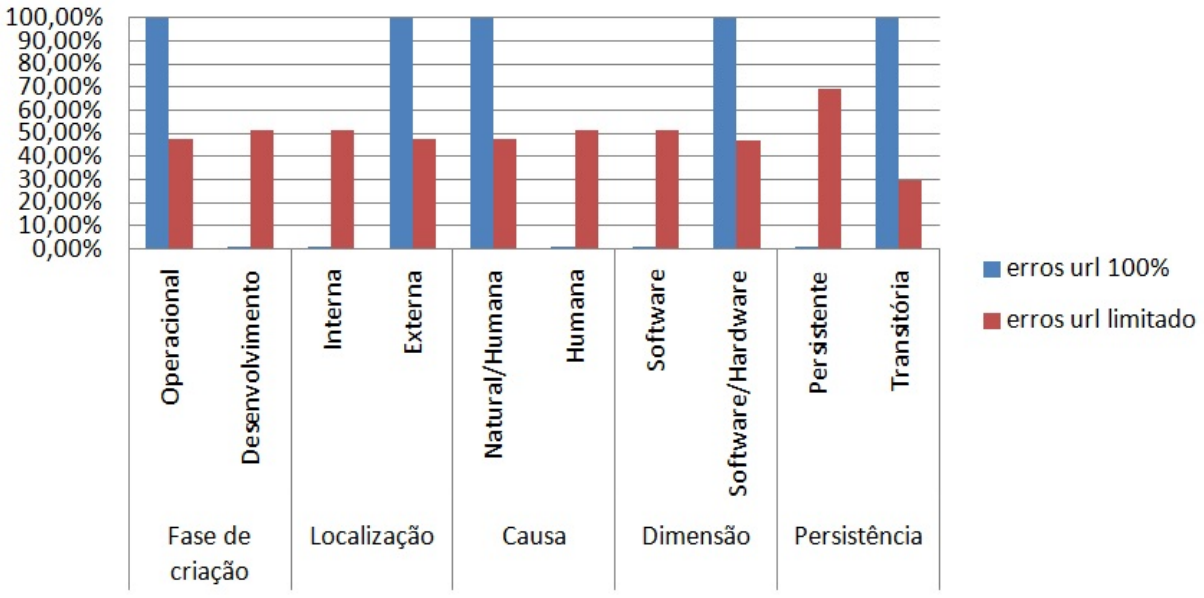


Figura 4.5: Classificação quantitativa das falhas ocorridas

O gráfico da Figura 4.6 mostra a porcentagem de cada atributo em relação à quantidade total de defeitos identificados, ou seja, que podem ocorrer no sistema. Observa-se que foram identificados mais defeitos de Parada, Inconsistente e Pequeno. Logo abaixo da Figura 4.6 encontra-se a legenda para cada sigla.

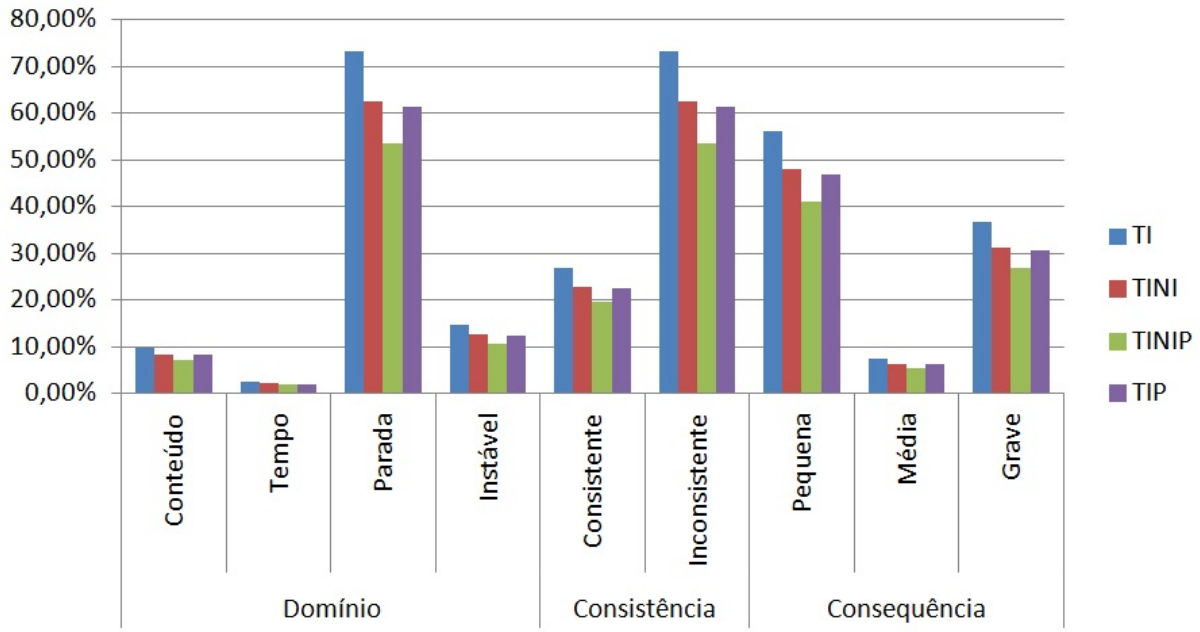


Figura 4.6: Classificação dos defeitos identificados

- TI = Total identificados.
- TINI = Total identificados com Não identificados.
- TINIP = Total identificados com Não identificados e com Propagados.
- TIP = Total identificados com Propagados.

O gráfico da Figura 4.7 mostra a porcentagem da classificação dos defeitos ocorridos de forma unitária, ou seja, sem considerar a quantidade de vezes que podem ter ocorrido. Observa-se que ocorreram mais defeitos de Parada, Inconsistente e Pequeno, porém, ocorreu um aumento de defeitos Médio em relação ao gráfico da Figura 4.6. Isto indica que há alguns componentes que param de funcionar, porém, os outros componentes do Observatório da Dengue continuam em execução, indicando que ainda podem haver mensagens sendo encaminhadas ao banco de dados MySQL.

Já o gráfico da Figura 4.8 mostra a porcentagem da classificação dos defeitos considerando a quantidade de vezes que eles podem ter ocorrido durante a execução do Observatório da Dengue no período delimitado. Observa-se que estão em destaque os atributos Instável, Consistente e Médio, indicando que há mensagens que estão corrompidas e não são aproveitadas pela sequência (*pipeline*) e a perda da mensagem é percebida por todos os componentes.

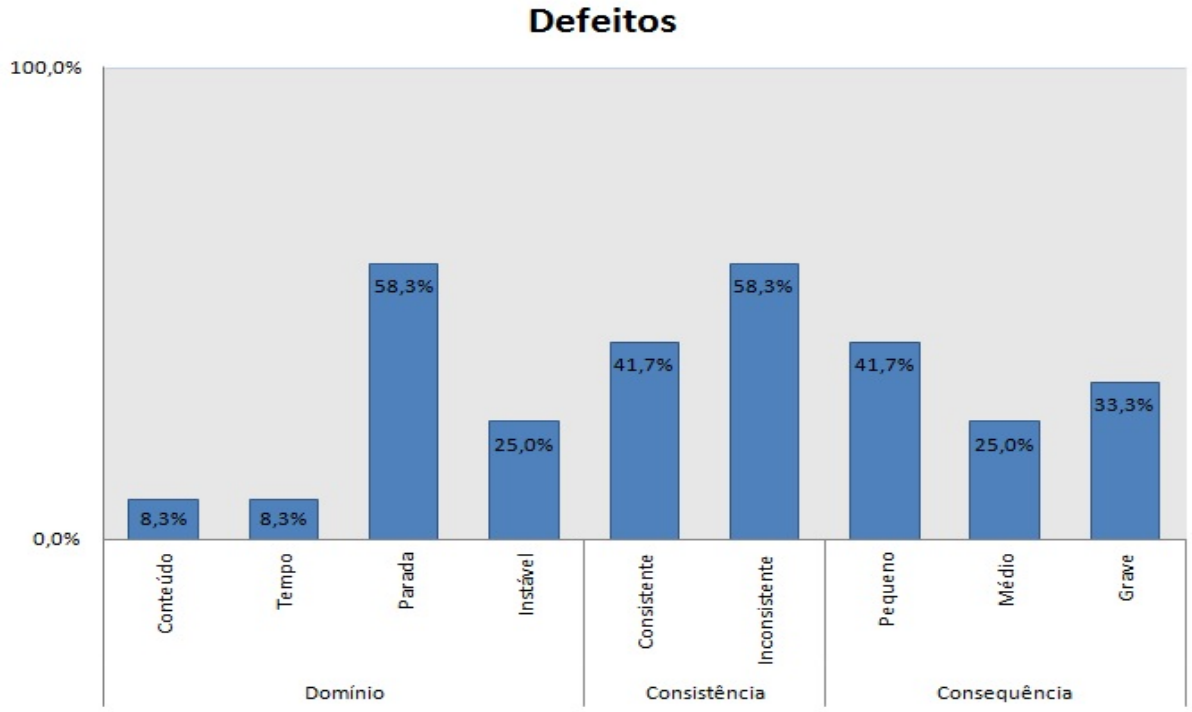


Figura 4.7: Classificação unitária dos defeitos ocorridos

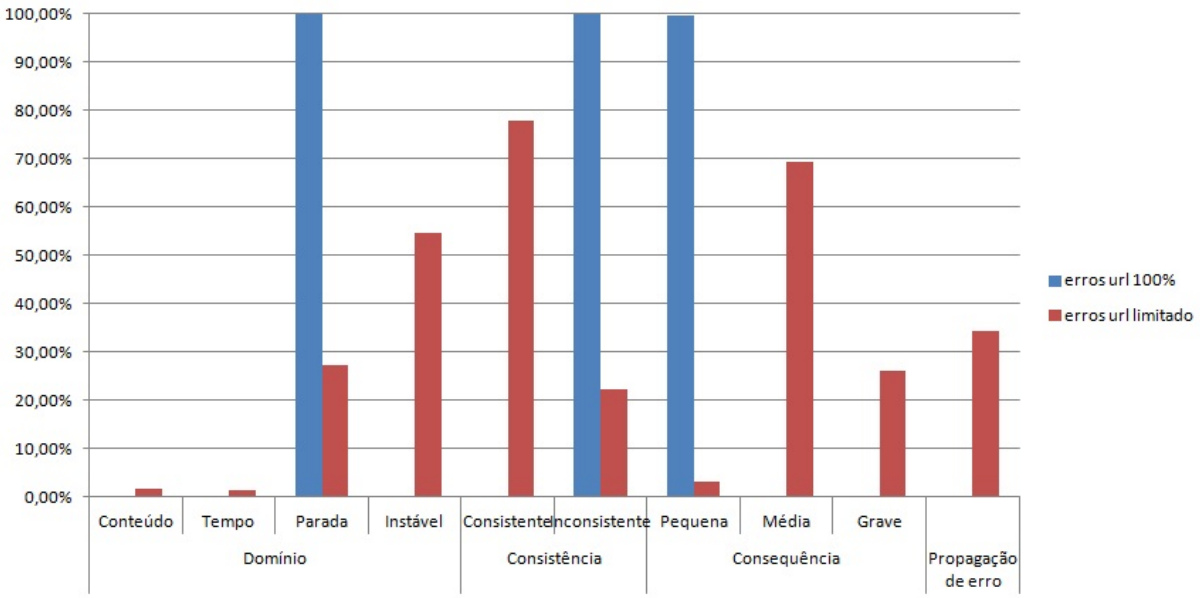


Figura 4.8: Classificação quantitativa dos defeitos ocorridos

4.1.2 Operacional

O servidor do Observatório da Dengue foi monitorado no período de 04/07/2013 até 02/09/2013 com exceção de algumas interrupções causadas pela instabilidade do antigo Sistema

Operacional que o *Zabbix-server* estava instalado ou das manutenções feitas (atualização do sistema operacional). A disponibilidade do ambiente de monitoramento(Zabbix-server) não afeta diretamente o estado de funcionamento correto do ambiente alvo(Observatório da Dengue), mas devemos considerar a cobertura que obtivemos durante o período de coleta de eventos:

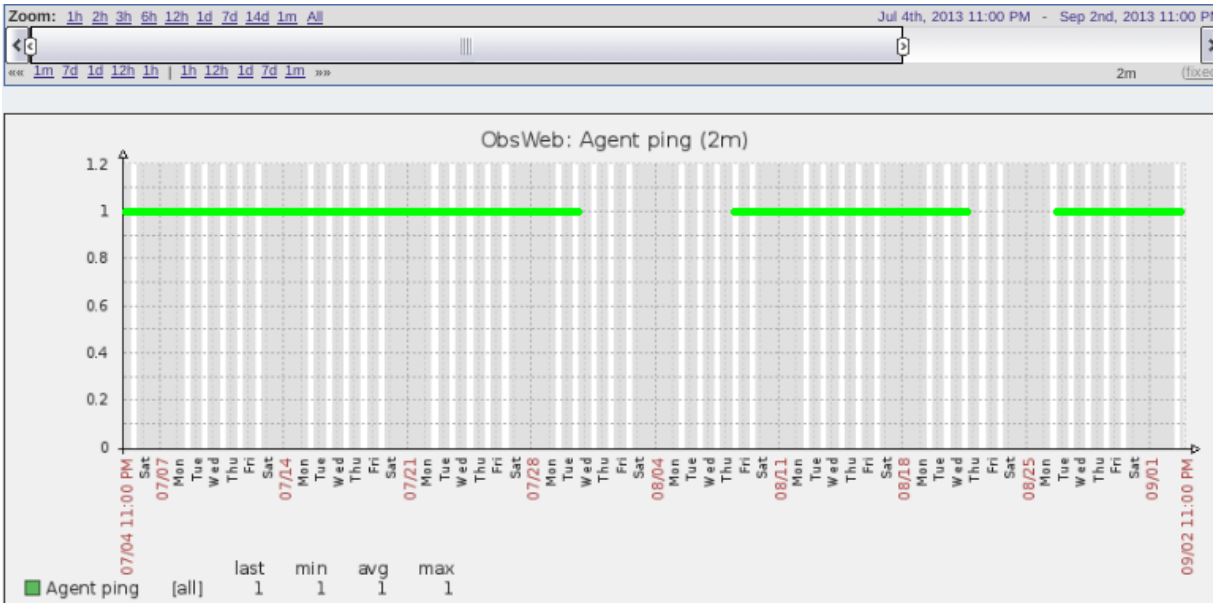


Figura 4.9: Disponibilidade do *Zabbix* durante o ultimo período analisado.

Tempo total analisado = 1 mês, 29 dias e 16h = 1416 horas.

Tempo total de falha na conexão entre o *Zabbix-server* e o servidor do Observatório da Dengue ≈ 334 horas.

Disponibilidade ≈ 75 % [1]

Com o relatório que mostra a disponibilidade e a quantidade da duração de eventos ocorridos 4.10, podemos observar que a quantidade da ocorrência de eventos com estado desconhecidos(coluna *UNKNOW*) é próxima à disponibilidade do **Zabbix**. A quantidade de tempo que ocorreram de erros é bastante baixa como observado na coluna *Problems*.

[1] Observação: A baixa disponibilidade do servidor *Zabbix* ocorreu em um período que a porta de comunicação (27050) estava bloqueada, e em pequenos períodos que o servidor foi desligado acidentalmente.

Name	Problems	Ok	Unknown
/etc/passwd has been changed on ObsWeb	0.0694%	75.5744%	24.3562%
Configured max number of opened files is too low on ObsWeb	0.0000%	69.8314%	30.1686%
Configured max number of processes is too low on ObsWeb	0.0000%	69.8312%	30.1688%
Disk I/O is overloaded on ObsWeb	0.0093%	73.5621%	26.4287%
Free disk space is less than 20% on volume /	0.0000%	76.3437%	23.6563%
Free disk space is less than 20% on volume /scratch	0.0000%	76.3464%	23.6536%
Free inodes is less than 20% on volume /	0.0000%	76.3405%	23.6595%
Free inodes is less than 20% on volume /scratch	0.0000%	76.3461%	23.6539%
Host information was changed on ObsWeb	0.0694%	75.5752%	24.3554%
Host name of zabbix_agentd was changed on ObsWeb	0.0000%	75.8436%	24.1564%
Hostname was changed on ObsWeb	0.0694%	75.5073%	24.4233%
Lack of available memory on server ObsWeb	0.0000%	73.5746%	26.4254%
Lack of free swap space on ObsWeb	0.0000%	73.5700%	26.4300%
ObsWeb has just been restarted	0.0578%	76.0642%	23.8780%
Processor load is too high on ObsWeb	0.0012%	73.5737%	26.4251%
Too many processes on ObsWeb	0.0000%	73.5781%	26.4219%
Too many processes running on ObsWeb	0.0000%	73.5805%	26.4195%

Figura 4.10: Eventos ocorridos com a quantidade de tempo.

Os dados obtidos durante o período analisado são variados, alguns apontam erros e outros são informativos, como podemos observar em 4.11. Caso seja necessário procurar por eventos que o *Zabbix* não conseguiu avaliar a expressão do *Trigger*, como está em 4.12

Time	Description	Status	Severity	Duration
Sep 2nd, 2013 03:22:12 PM	Disk I/O is overloaded on ObsWeb	OK	Warning	14d 9h 57m
Sep 2nd, 2013 03:20:12 PM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	2m
Aug 30th, 2013 08:03:07 PM	Processor load is too high on ObsWeb	OK	Warning	23d 15h 45m
Aug 30th, 2013 08:02:07 PM	Processor load is too high on ObsWeb	PROBLEM	Warning	1m
Aug 29th, 2013 05:20:17 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	4d 20h 33m
Aug 29th, 2013 05:18:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	PROBLEM	Average	2m 17s
Aug 29th, 2013 03:51:58 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	1h 26m 2s
Aug 29th, 2013 03:47:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	PROBLEM	Average	4m 58s
Aug 26th, 2013 08:17:12 PM	Disk I/O is overloaded on ObsWeb	OK	Warning	6d 19h 3m
Aug 26th, 2013 08:16:12 PM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Aug 26th, 2013 04:09:24 PM	ObsWeb has just been restarted	OK	Information	27d 19h 39m

Figura 4.11: Tela do *Zabbix* mostrando os eventos com estados que foram estáveis.

Aug 31st, 2013 02:42:12 PM	Disk I/O is overloaded on ObsWeb	OK	Warning	2d 38m
Aug 31st, 2013 02:42:07 PM	Processor load is too high on ObsWeb	OK	Warning	2d 23h 8m
Aug 31st, 2013 02:42:03 PM	Too many processes on ObsWeb	OK	Warning	2d 23h 8m
Aug 31st, 2013 02:42:02 PM	Too many processes running on ObsWeb	OK	Warning	2d 23h 8m
Aug 31st, 2013 02:42:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	2d 23h 12m
Aug 31st, 2013 02:41:27 PM	Lack of available memory on server ObsWeb	UNKNOWN	Average	1m
Aug 31st, 2013 02:41:36 PM	Free disk space is less than 20% on volume /scratch	UNKNOWN	Warning	1m
Aug 31st, 2013 02:41:35 PM	Free disk space is less than 20% on volume /	UNKNOWN	Warning	1m
Aug 31st, 2013 02:41:32 PM	Free inodes is less than 20% on volume /scratch	UNKNOWN	Warning	1m
Aug 31st, 2013 02:41:31 PM	Free inodes is less than 20% on volume /	UNKNOWN	Warning	1m
Aug 31st, 2013 02:41:21 PM	Lack of free swap space on ObsWeb	UNKNOWN	Warning	1m
Aug 31st, 2013 02:41:12 PM	Disk I/O is overloaded on ObsWeb	UNKNOWN	Warning	1m

Figura 4.12: Tela do *Zabbix* mostrando também os eventos com estados desconhecidos.

As **triggers** foram separados em número de ocorrências em 4.13 mostrando o número de ocorrências durante o período analisado.

Host	Trigger	Severity	Number of status changes
ObsWeb	Zabbix agent on ObsWeb is unreachable for 5 minutes	Average	18
ObsWeb	Disk I/O is overloaded on ObsWeb	Warning	12
ObsWeb	ObsWeb has just been restarted	Information	10
ObsWeb	/etc/passwd has been changed on ObsWeb	Warning	2
ObsWeb	Host information was changed on ObsWeb	Information	2
ObsWeb	Hostname was changed on ObsWeb	Information	2
ObsWeb	Processor load is too high on ObsWeb	Warning	2

Figura 4.13: Relatório com número de **triggers** acionados durante o período de análise.

4.1.3 Análise dos resultados operacionais os atributos de dependabilidade

Como observado anteriormente, houve poucos episódios de falhas operacionais ocorridas durante o período analisado, nessa seção serão tratados as formas de alcançar os atributos de dependabilidade desejados. Em uma análise *top-down*, primeiramente é feita a análise dos defeitos, observando os erros detectados no *Zabbix* [1] e finalmente identificar as falhas que se propagaram para gerar o defeito.

1. **Defeito na conexão com o *Zabbix server***

   - **Falha:** Bloqueio de porta pelo *firewall* da UnB

     Prevenção: Manter o pedido de liberação de portas do CPD da UnB sempre atualizado

     Remoção: Enviar novo pedido ao CPD(Centro de Processamento de Dados)

     Previsão: Evento ocorreu apenas uma vez, e a data de bloqueio da porta já estava prevista A grande quantidade de erros captados pelo *Zabbix* do tipo, deve-se aos reiniciamento do sistema operacional do *Zabbix server* para atualização do sistema, ou em quedas de energia.

2. **Defeito de processo não respondente**

   - **Falha: Elevado número de requisições de acesso à memória por processos**

     Prevenção:

     Limitar a prioridade de execução do processo;

     Alterar a forma de agendamento de I/O (ocioso, melhor esforço com baixa prioridade).

     Remoção:

     Reiniciar o(s) processo(s) responsável(eis) pelo elevada taxa de I/O.

     Previsão:

     Ocorreram em horários específicos e com pouca duração. Os horários do acontecimento seguem um padrão possível de agendamento de execução de tarefa.

   - **Falha:** Um ou mais processos executam por muito tempo.

     Prevenção: Limitar a prioridade de execução do processo;

Remoção: Reiniciar processo que ocupa o processador.

Previsão: Ocorreram em horários específicos e com pouca duração. Os horários do acontecimento seguem um padrão possível de agendamento de execução de tarefa.

3. **Defeito de sistema não respondente**

- **Falha: Elevado número de requisições de acesso à memória por processos**

  Prevenção:

  Limitar a prioridade de execução do processo;

  Alterar a forma de agendamento de I/O (ocioso, melhor esforço com baixa prioridade).

  Remoção:

  Reiniciar o(s) processo(s) responsável(eis) pelo elevada taxa de I/O.

  Previsão:

  Ocorreram em horários específicos e com pouca duração. Os horários do acontecimento seguem um padrão possível de agendamento de execução de tarefa como observado em 4.14.

- **Falha:** Um ou mais processos executam por muito tempo.

  Prevenção: Limitar a prioridade de execução do processo;

  Remoção: Reiniciar processo que ocupa o processador.

  Previsão: Ocorreram em horários específicos, mas com pouca duração.

Time	Host	Description	Status	Severity	Duration
Sep 2nd, 2013 03:22:12 PM	ObsWeb	Disk I/O is overloaded on ObsWeb	OK	Warning	6d 21h 26m
Sep 2nd, 2013 03:20:12 PM	ObsWeb	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	2m
Aug 26th, 2013 08:17:12 PM	ObsWeb	Disk I/O is overloaded on ObsWeb	OK	Warning	6d 19h 3m
Aug 26th, 2013 08:16:12 PM	ObsWeb	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Aug 18th, 2013 06:37:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	OK	Warning	8d 13h 39m
Aug 18th, 2013 06:36:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Aug 17th, 2013 03:33:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	OK	Warning	1d 3h 3m
Aug 17th, 2013 03:32:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Jul 12th, 2013 06:27:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	OK	Warning	1m 5d 21h
Jul 12th, 2013 06:26:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Jul 9th, 2013 06:27:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	OK	Warning	2d 23h 59m
Jul 9th, 2013 06:26:12 AM	ObsWeb	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m

Figura 4.14: Padrão dos eventos relacionados à sobrecarga de disco.

4. **Defeito: Impossibilidade de gravar novos arquivos**

- **Falha:** Excesso de dados no banco de dados.

  Prevenção: Verificar locais onde existem arquivos temporários que podem ser descartados periodicamente e criar um mecanismo (usando gatilho no *Zabbix* ou tarefa agendada) que exclui estes arquivos.

Remoção: Remover, ou mover arquivos que não tem mais necessidade de ficar na partição;

Previsão: A coleta de dados, que é a principal responsável pelo aumento do uso do disco, aumenta gradualmente e lentamente como observado em 4.15, tornando fácil a prevenção.

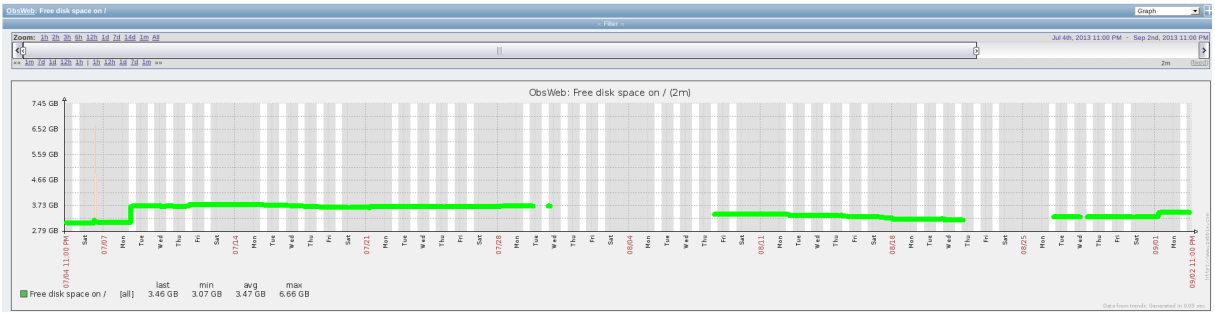


Figura 4.15: Diminuição gradual do espaço livre na partição raiz.

Para correlacionamento entre as duas ferramentas, a ferramenta de inspeção e a ferramenta de monitoramento, seleciona-se o período de monitoramento comum aos dois. Houve eventos que foram possíveis relacionar através do momento que ocorreram. Na ferramenta de monitoramento apresentaram os erros:

Time	Description	Status	Severity	Duration
Aug 30th, 2013 08:03:07 PM	Processor load is too high on ObsWeb	OK	Warning	18d 48m
Aug 30th, 2013 08:02:07 PM	Processor load is too high on ObsWeb	PROBLEM	Warning	1m
Aug 29th, 2013 05:20:17 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	4d 20h 33m
Aug 29th, 2013 05:18:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	PROBLEM	Average	2m 17s
Aug 29th, 2013 03:51:58 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	OK	Average	1h 26m 2s
Aug 29th, 2013 03:47:00 PM	Zabbix agent on ObsWeb is unreachable for 5 minutes	PROBLEM	Average	4m 58s
Aug 26th, 2013 08:17:12 PM	Disk I/O is overloaded on ObsWeb	OK	Warning	6d 19h 3m
Aug 26th, 2013 08:16:12 PM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Aug 26th, 2013 04:09:24 PM	ObsWeb has just been restarted	OK	Information	22d 4h 41m
Aug 26th, 2013 03:59:24 PM	ObsWeb has just been restarted	PROBLEM	Information	10m
Aug 18th, 2013 06:37:12 AM	Disk I/O is overloaded on ObsWeb	OK	Warning	8d 13h 39m
Aug 18th, 2013 06:36:12 AM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m
Aug 17th, 2013 03:33:12 AM	Disk I/O is overloaded on ObsWeb	OK	Warning	1d 3h 3m
Aug 17th, 2013 03:32:12 AM	Disk I/O is overloaded on ObsWeb	PROBLEM	Warning	1m

Figura 4.16: Eventos correlacionados.

Assim como no dia 29 apresentou erro de conexão com Observatório da Dengue, a Ferramenta de Inspeção [6] apresentou o seguinte erro, relacionado à perda de conexão da rede:

- Às 15:42:11 do dia 29/08/2013 até 16:03:20 do dia 30/08/2013, coletou mensagens de erro no MySQL e na coleta de dados no twitter. Não houve coleta de dados

4.2 Discussão

4.2.1 Discussão dos resultados operacionais

Tendo como base a baixa complexidade do servidor do Observatório da Dengue e o pequeno período monitorado, foram obtidos poucos resultados para analisar e propor mudanças em arquitetura física ou de sistema operacional para o Observatório da Dengue. Com base nos erros detectados pelo *Zabbix*, um possível problema, seria problemas de escalabilidade caso os erros em disco e processamento ocorram com maior frequência. Com todo o ambiente criado nesse trabalho mostrou-se potencialmente poderoso para rastrear e remover falhas no Observatório da Dengue.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

Compreender o funcionamento da arquitetura do Observatório da Dengue, sem uma documentação detalhada, teve um gasto considerável de tempo para o projeto. A primeira tentativa de compreensão do funcionamento do Observatório da Dengue foi instrumentar todos os arquivos que possuíam nomes semelhantes aos dos filtros encontrados no browser do RabbitMQ ativos. No entanto, descobrimos logo em seguida que não é possível instrumentar métodos que realizam publicações nos filtros. Porém, é possível verificar as mensagens que são retiradas de um filtro para serem processadas e passadas adiante.

Mesmo com o esforço para compreensão da arquitetura do sistema do Observatório da Dengue, ainda ocorrem instrumentações de partes de alguns componentes, com a finalidade de tornar a busca pelos erros mais precisa.

Com a coleta de eventos, observamos que é possível uma *manutenção preventiva* ao invés de apenas uma *manutenção corretiva*, como parecia estar ocorrendo no Observatório da Dengue. Como os eventos são anotados e especificam com *tags* como localizar cada tipo de evento, torna-se viável identificar os erros que ocorrem com mais frequência em um determinado período de tempo e prever as falhas antes que elas venham a causar algum defeito.

A *cobertura* de cada componente foi realizada com a verificação dos eventos relativos à quantidade de mensagens que chegavam e saíam de cada um dos componentes e através da verificação dos eventos de erros coletados do corpo de cada um deles, evitando a instrumentação dos métodos utilizados pelos componentes da sequência (*pipeline*) que são implementados por outros componentes. Essa cobertura possibilitou verificar a relação entre a quantidade de mensagens processadas na sequência e a quantidade de erros produzidos. Mantivemos a sequência (*o pipeline*) do Observatório da Dengue o mais controlado possível e dentro de nossas limitações de tempo.

5.2 Trabalhos futuros

A análise de dependabilidade foi realizada na monografia do aluno Túlio Porto [14] a partir dos dados coletados neste trabalho, com isso esperamos validar os resultados obtidos e avançar para a próxima etapa, que é a de planejamento do *painel de controle de qualidade* para o Observatório da Dengue.

Uma das idéias é a implementação de gráficos ou indicadores, sendo que os erros seriam lidos diretamente da coleção do MongoDB onde é feita a armazenagem de eventos, associando uma *tag Erro*, por exemplo, ao nome do componente representado pela *tag*, por exemplo, *modulo* e com o valor *reader_filter*. Isso ajudaria o administrador do sistema a observar os componentes sem a necessidade de filtrar as mensagens (algo que já é realizado sem muitas dificuldade) e a comparar, ao olhar para o monitor, quais componentes estão produzindo mais erros em determinados períodos de tempo (também sem a necessidade de filtrar).

Referências

[1] Zabbix documentation. http://www.zabbix.com/documentation/2.0/manual/introduction, 2012 (Acesso em 01 dez. 2012). 2, 27, 29, 34, 36, 48, 57

[2] Rabbitmq. http://www.rabbitmq.com/tutorials/tutorial-one-python.html, 2013 (Acesso em 06 maio. 2013). 19, 38

[3] Observatório da dengue. http://www.observatorio.inweb.org.br/dengue/conteudo/inicial, 2013 (Acesso em 29 set. 2013). vii, 1, 17, 18

[4] Observatório da web. http://observatorio.inweb.org.br/, 2013 (Acesso em 29 set. 2013). 1

[5] Virgílio A. F. Almeida, Walter dos Santos Filho, Gisele L. Pappa, Wagner Meira Jr, Dorgival Guedes, Adriano Veloso, Adriano M. Pereira, Pedro H. C. Guerra, Arlei L. da Silva, Fernando H. J. Mourão, Tiago R. de Magalhães, Felipe M. Machado, Letícia L. Cherchiglia, Lívia S. Simões, Rafael A. Batista, Filipe L. Arcanjo, Gustavo M. Brunoro, Nathan R. B. Mariano, Gabriel Magno, Marco Túlio C. Ribeiro, Leonardo V. Teixeira, Altigran S. da Silva, Bruno Wanderley Reis, and Regina Helena Silva. Observatório da web: Uma plataforma de monitoração, síntese e visualização de eventos massivos em tempo real. pages 110–120, 2010. vii, 1, 18, 19

[6] Thiago Araújo, Carla Wanderley, and Arndt von Staa. Um mecanismo de introspecção para depurar o comportamento de sistemas distribuídos. 2010. vii, 2, 24, 25, 26, 30, 31, 32, 50

[7] A. Avižienis, J.C. Laprie, B. Randell, and University of Newcastle upon Tyne. Computing Science. Fundamental concepts of dependability. 2001. 5, 6, 8, 31

[8] Algirdas Avižienis, Jean Claude Laprie, Brian Randell, and Carl Landwehrn. Basic concepts and taxonomy of dependable and secure computing. *IEEE Computer Society*, 1(1):11–33, January-March 2004. vii, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 31, 34

[9] Bruno de O. S. Silva and Bruno A. P. de S. Souza. *Uma Abordagem Exploratória de Dependabilidade do Twitter no Contexto do Observatório da Web*. UnB, Campus Darcy Ribeiro - Universidade de Brasília, 2011. Monografia apresentada como requisito parcial para conclusão do Curso de Computação – Licenciatura. 5, 18, 20

[10] Walter dos Santos Filho. Tweets e notificações ao longo das semanas para rio de janeiro/rj. Gráfico recomendado para justificação da importância do Observatório da Dengue, setembro 2013. vii, 23

[11] Renato dos Santos Leal and Artur Azevedo. *Estudo sistemático em dependabilidade e métodos ágeis: uma análise de falhas e defeitos*. UnB, Campus Darcy Ribeiro - Universidade de Brasília, 2013. Monografia apresentada como requisito parcial para conclusão do Curso de Ciência da Computação. 12

[12] Jean-Claude Laprie. Dependability of computer systems: from concepts to limits. 1998. 5

[13] Leonardo Mariani and Fabrizio Pastore. Automated identification of failure causes in system logs. *Washington, DC, USA: IEEE Computer Society*, pages 117–126, 2008. 2

[14] Túlio Porto. *Uma Abordagem Exploratória de Análise de Dependabilidade no Contexto do Observatório da Web*. UnB, Campus Darcy Ribeiro - Universidade de Brasília, 2013. Monografia apresentada como requisito parcial para conclusão do Curso de Computação – Licenciatura. 52

[15] Ian Sommerville. *Engenharia de Software*. Pearson, São Paulo, 8ª edition, 2007. 24

[16] C. Strozzi. Nosql. http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql, (August. 2013). 2

[17] Adriano Veloso, Wagner Meira Jr., and Mohammed J. Zaki. Lazy associative classification. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 645–654, Washington, DC, USA, 2006. IEEE Computer Society. 22

[18] Adriano Veloso, Gisele L. Pappa, and Wagner Meira Jr. Mineração de grandes volumes de dados em tempo real - uma visão dos desafios e soluções de mineração de dados no contexto do observatório da web. Slides do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais - DCC-UFMG, 2012. 18, 19

[19] Arndt von Staa. *Programação Modular*. Campus, 2000. 24

[20] Taisy Silva Weber. Tolerância a falhas: conceitos e exemplos. 2001. Programa de Pós-Graduação em Computação - Instituto de Informática UFRGS. 5

Capítulo 6

Apendices

6.1 Instalação do ambiente *Zabbix*

Para monitorar um sistema com o *Zabbix*, é necessário mapear toda infra-estrutura usada pelo ambiente alvo, como o sistema ObsWeb foi disponibilizado em uma única VM hospedada na UFMG, foi necessário instalar apenas o *Zabbix server* em um servidor dedicado na UnB e o *Zabbix Agent* na *Virtual Machine* da UFMG, sem a necessidade de montar uma rede complexa de fluxo entre computadores.

Segue a especificação da arquitetura em que o Observatório da Dengue está rodando, lembrando que é uma maquina virtual:

- A virtualização completa do sistema é feita através do software **QEMU**;
- **Processador:** 2.6 GHz 64 bits;
- **Memórias:**

  RAM: 2Gb RAM;

  Virtual: 4Gb SWAP;

- **Memória física:** 10Gb para sistema raiz / + 20Gb para /scratch;
- **Rede:** Ethernet genérica;
- **Sistema Operacional:**

  Distribuição: GNU/LINUX Ubuntu 12.04.2 LTS;

  Kernel: Linux 3.2.0-51-virtual x86_64.

6.1.1 Lado servidor

Foram feitas 2 instalações do zabbix server durante a coleta e analise de dados, inicialmente, foi instalado o sistema operacional Ubuntu 13.04, mas apresentou problemas graves de disponibilidade, comprometendo o monitoramento ininterrupto do ObsWeb. Após isso, foi instalado o sistema operacional Debian Wheezy, no qual abordaremos a instalação do zabbix com mais detalhes.

Sistema Operacional: Como descrito acima, foi instalado o SO Debian Wheezy simples, apenas com uma interface gráfica XFCE para visualização do Zabbix Front-end.

Zabbix server: Seguindo a documentação disponibilizada no site oficial do zabbix [1] e utilizando os seguintes passos:

1. Adicionar o repositório de teste oficial do Debian para instalar a versão mais nova do zabbix, para isso basta descomentar as seguintes linhas em /etc/apt.sources.list

```
deb http://ftp.fr.debian.org/debian/ testing main
deb-src http://ftp.fr.debian.org/debian/ testing main
```

2. Atualizar a lista de repositórios e instalar o zabbix-server e o zabbix frontend

```
# apt-get update
# apt-get install zabbix-server-mysql zabbix-frontend-php
```

3. Comente novamente as linhasdo repositório teste após a instalação.

4. Para que o Zabbix server inicie junto com o sistema operacional , substitua o valor da variável START=no para START=yes no arquivo /etc/default/zabbix-server

5. Criar a base de dados e dar permissões ao zabbix:

```
create database zabbix;
grant all privileges on zabbix.* to zabbix@localhost identified
by 'chavedeacesso';
exit
```

6. Extrair os modelos de banco de dados para a base de dados do zabbix no mysql:

```
# zcat /usr/share/zabbix-server-mysql/schema.sql.gz | mysql
-uzabbix -pchavedeacesso zabbix
# zcat /usr/share/zabbix-server-mysql/images.sql.gz | mysql
-uzabbix -pchavedeacesso zabbix
# zcat /usr/share/zabbix-server-mysql/data.sql.gz | mysql
-uzabbix -pchavedeacesso zabbix
```

7. Configure a senha do banco de dados que adicionando ao arquivo: /etc/zabbix/zabbix_server.conf

8. Cria-se o arquivo /etc/apache2/conf.d/zabbix-server de configuração no apache para o zabbix com o seguinte conteúdo:

```
# Diretivas para o zabbix-server
# Define /zabbix alias, this is the default
<IfModule mod_alias.c>
Alias /zabbix /usr/share/zabbix
</IfModule>
<Directory /usr/share/zabbix>
```

```
Options Indexes FollowSymLinks MultiViews
AllowOverride None
Order allow,deny
allow from all
    </Directory>
```

9. Reinicie o servidor apache:

```
# apache2ctl restart
```

10. Modificado o arquivo /etc/php5/apache2/php.ini com as seguintes configurações:

```
max_execution_time = 600
max_input_time = 600
memory_limit = 256M
post_max_size = 32M
upload_max_filesize = 16M
date.timezone = "America/Sao_Paulo"
```

11. Reinicie o serviço do zabbix

```
/etc/init.d/zabbix-server restart
```

depois o arquivo de configuração exemplo é copiado, fazendo os ajustes de permissão necessários.

```
    cp /usr/share/zabbix/conf/zabbix.conf.php.example
/etc/zabbix/zabbix.conf.php
    nano  /etc/zabbix/zabbix.conf.php (modifique a chave de acesso)
    chown root:www-data /etc/zabbix/zabbix.conf.php
    chmod 660 /etc/zabbix/zabbix.conf.php
```

12. (Pluggin extra) Para instalar o *pluggin* de correlacionamento de eventos no *Zabbix frontend*, é necessário baixar o arquivo disponibilizado no site (colocar referencia)

```
    wget link
    sudo chmod +x arquivo
    sh arquivo
```

configure o instalador conforme as figuras: –>Add figuras

6.1.2 Framework do Zabbix

Após terminada a instalação e configuração do *Zabbix-server*, o *framework* deve ser configurado para monitorar o ambiente alvo na forma desejada. A imagem seguinte mostra como adicionar e configurar o servidor que hospeda o Observatório da Dengue na UFMG.

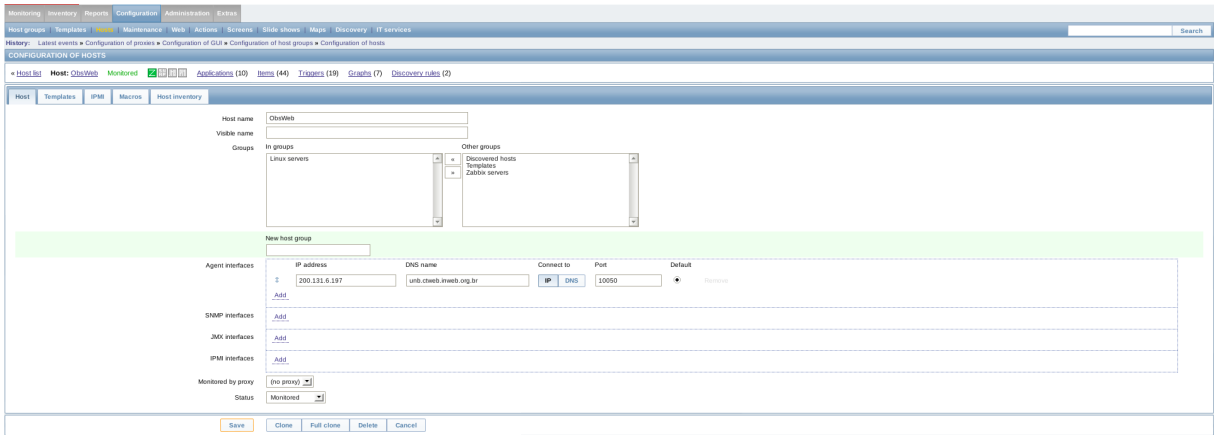


Figura 6.1: Configuração para conexão com o *Zabbix-agent* no Observatório da Dengue.

6.1.3 Lado cliente

O ambiente alvo, o Observatório da Dengue, está instalado em uma *Virtual Machine* com acesso remoto via SSH pela porta 443, mas para a comunicação com o *Zabbix server*, precisamos da porta 27050 liberada nas redes do computador hospedado UFMG e da UnB.

1. O *Zabbix agent* foi instalado enviando, via ssh, o arquivo de instalação do repositório *Zabbix* para Ubuntu.

```
scp -P 443 repositório.deb ubuntu@unb.ctweb.inweb.org.br:/tmp/
sudo dpkg -U repositório.deb
sudo apt-get update
sudo apt-get install zabbix-agent
```

2. Após feita a instalação, é necessário configurar o /etc/zabbix/zabbix-agent.conf para conectar com o *zabbix-server*

```
LogFileSize=20
DebugLevel=3
Server=164.41.209.70
Timeout=3
```

3. Para concluir a instalação e configuração, é necessário reiniciar o serviço do *zabbix-agent*:

```
sudo systemctl restart zabbix-agent
```

6.2 Classificação detalhada de falhas e defeitos operacionais

6.2.1 Defeitos mapeadas

Defeito na conexão com o servidor remotamente
Domínio: Defeito de parada
Consistência: Consistente/Inconsistente
Consequência: Pequena

Defeito de processo não respondente
Domínio: Defeito de parada
Consistência: Consistente
Consequência: Média

Defeito de sistema não respondente
Domínio: Defeito de parada
Consistência: Consistente
Consequência: Grave

Defeito na conexão com o *Zabbix server*
Domínio: Defeito de parada
Consistência: Inconsistente
Consequência: Pequena

Defeito de conexão na interface de rede
Domínio: Defeito de parada
Consistência: Inconsistente / Consistente
Consequência: Média

Impossibilidade de gravar novos arquivos
Domínio: Defeito de parada
Consistência: Consistente
Consequência: Média

Impossibilidade de adicionar dados ao banco de dados
Domínio: Defeito de parada
Consistência: Consistente
Consequência: Média

6.2.2 Falhas mapeadas

Excesso de programas em execução
Fase de criação: de desenvolvimento
Localização: Interna
Causa: Humana

Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade: Incompetência
Persistência: Transitória

Buffer de programa excessivamente grande
Fase de criação: de desenvolvimento
Localização: Interna
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade: Incompetência
Persistência: Transitória

Cabo de conexão de interface de rede desconectado ou rompido
Fase de criação: Operacional
Localização: Externa
Causa: Humana ou Natural
Dimensão: Hardware
Objetivo: Não maliciosa
Intenção: Deliberada
Capacidade: Incompetência / Acidental
Persistência: Persistente

Problema no carregamento de driver para a interface de rede
Fase de criação: Operacional
Localização: Interna
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade: Incompetência / Acidental
Persistência: Persistente

Bloqueio de portas ou protocolos por firewalls
Fase de criação: Operacional
Localização: Externa
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Deliberada
Capacidade: Incompetência
Persistência: Persistente

Defeito físicos no disco rígido
Fase de criação: Operacional
Localização: Externa
Causa: Natural
Dimensão: Hardware
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade: Acidental
Persistência: Persistente

Elevado número de requisições de acesso à memória por processos
Fase de criação: Operacional
Localização: Interna
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada / Deliberada
Capacidade: Incompetência / Acidental
Persistência: Transitória

Um ou mais processos executam por muito tempo
Fase de criação: Operacional
Localização: Interna
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada / Deliberada
Capacidade: Incompetência / Acidental
Persistência: Transitória

Queda de energia do servidor no Observatório da Dengue
Fase de criação: Operacional
Localização: Externa
Causa: Humana / Natural
Dimensão: Hardware
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade: Acidental
Persistência: Transitória

Administrador do sistema reinicia para manutenções
Fase de criação: Operacional
Localização: Externa
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa

Intenção: Não-deliberada
Capacidade:
Persistência: Transitória

Excesso de dados no banco de dados
Fase de criação: Operacional
Localização: Interna
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade:
Persistência: Persistente

Excesso de arquivos grandes em disco
Fase de criação: Operacional
Localização: Interna
Causa: Humana
Dimensão: Software
Objetivo: Não maliciosa
Intenção: Não-deliberada
Capacidade:
Persistência: Persistente

6.3 Interrupções de serviço do Observatório da Dengue

- Às 18:38:51 até às 23:28:41 do dia 25/08/2013 não realizou coleta.
- Às 23:28:42 do dia 25/08/2013 coletou uma mensagem de erro de consumo da fila.
- Às 23:28:43 do dia 25/08/2013 até às 18:10:20 do dia 26/08/2013 coletou apenas três mensagens de erro de consumo da fila, às 23:28:43, às 02:18:43, 07:08:43 e às 13:38:50.
- Às 18:10:21 do dia 26/08/2013 recomeçou a coleta dos dados.
- Às 15:42:11 do dia 29/08/2013 até 16:03:20 do dia 30/08/2013, coletou mensagens de erro no MySQL e na coleta de dados no twitter.
- Às 16:03:21 do dia 30/08/2013 voltou a coletar mais mensagens de erro no MySQL e na coleta de dados no twitter.
- Às 16:26:59 do dia 30/08/2013 voltou a coletar mensagens normalmente.

6.4 Diagramas do Observatório da Dengue

6.4.1 Diagramas de sequência

A partir de análises realizadas sobre a versão parcial do Observatório da dengue, foram construídos diagramas de sequência para cada componente do pipeline (ver 2.3.3), o que possibilitou uma maior compreensão de como ocorre a comunicação entre os componentes.

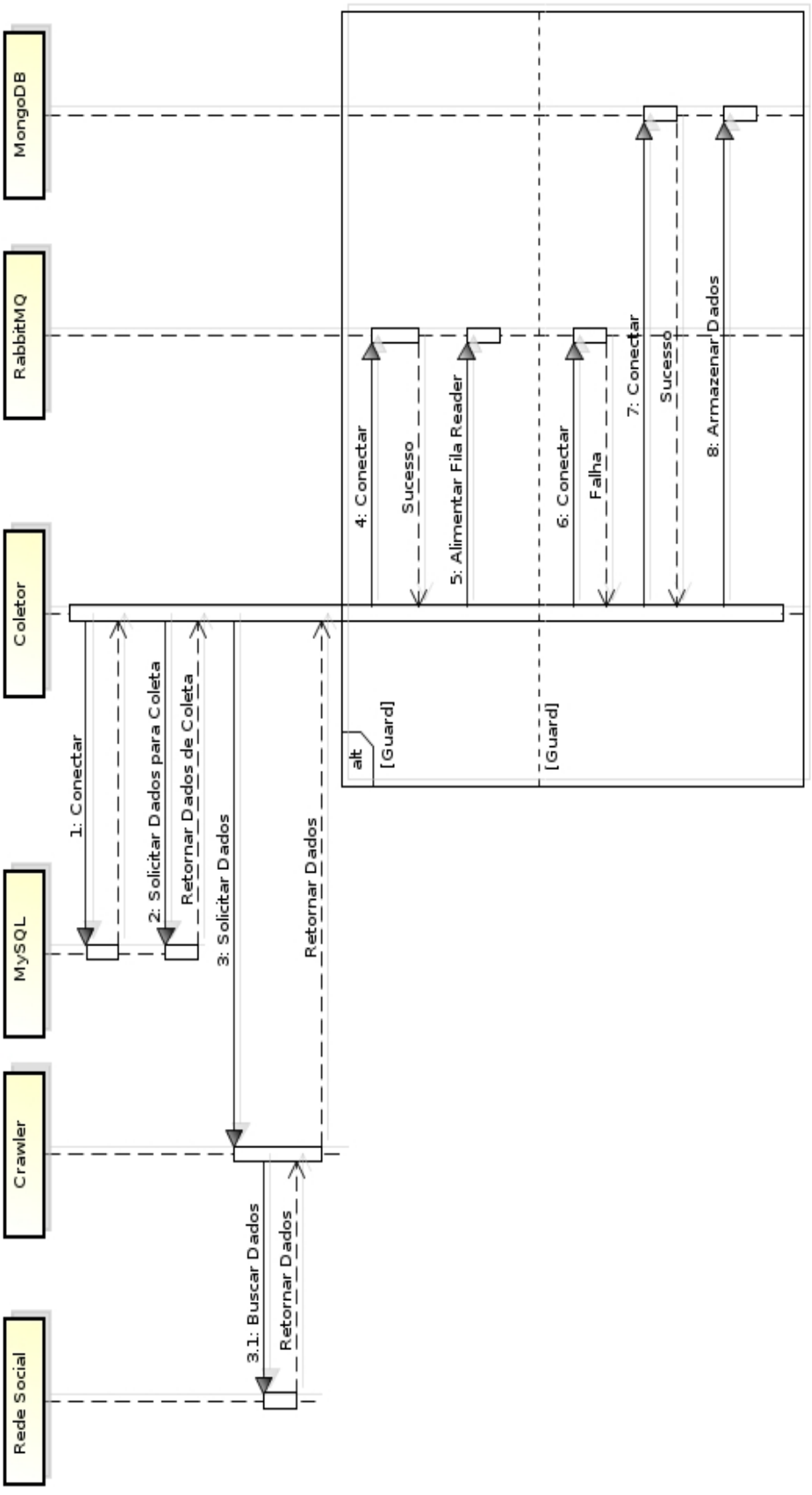


Figura 6.2: Diagrama de sequência da coleta de dados

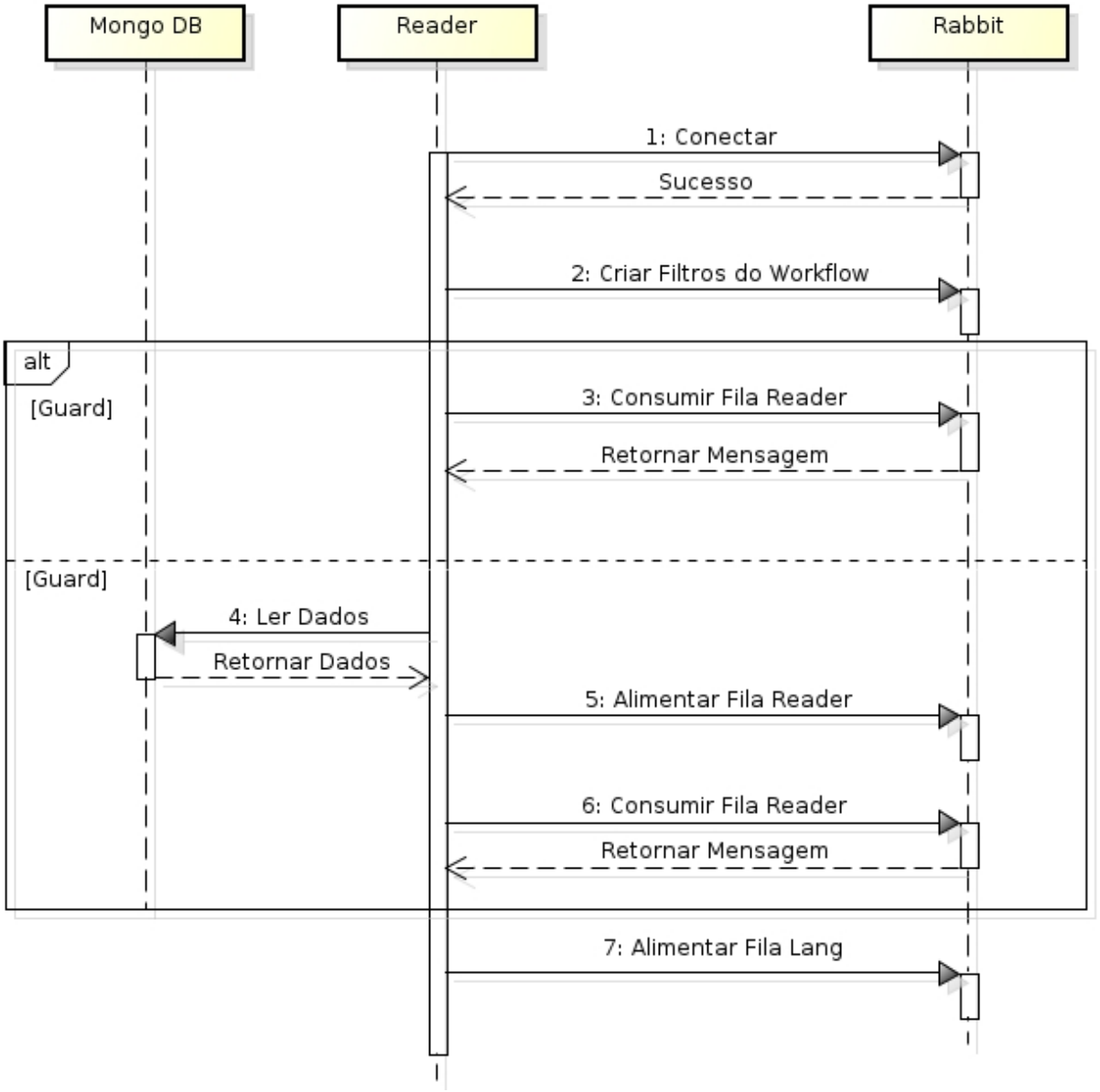


Figura 6.3: Diagrama de sequência do processo Reader

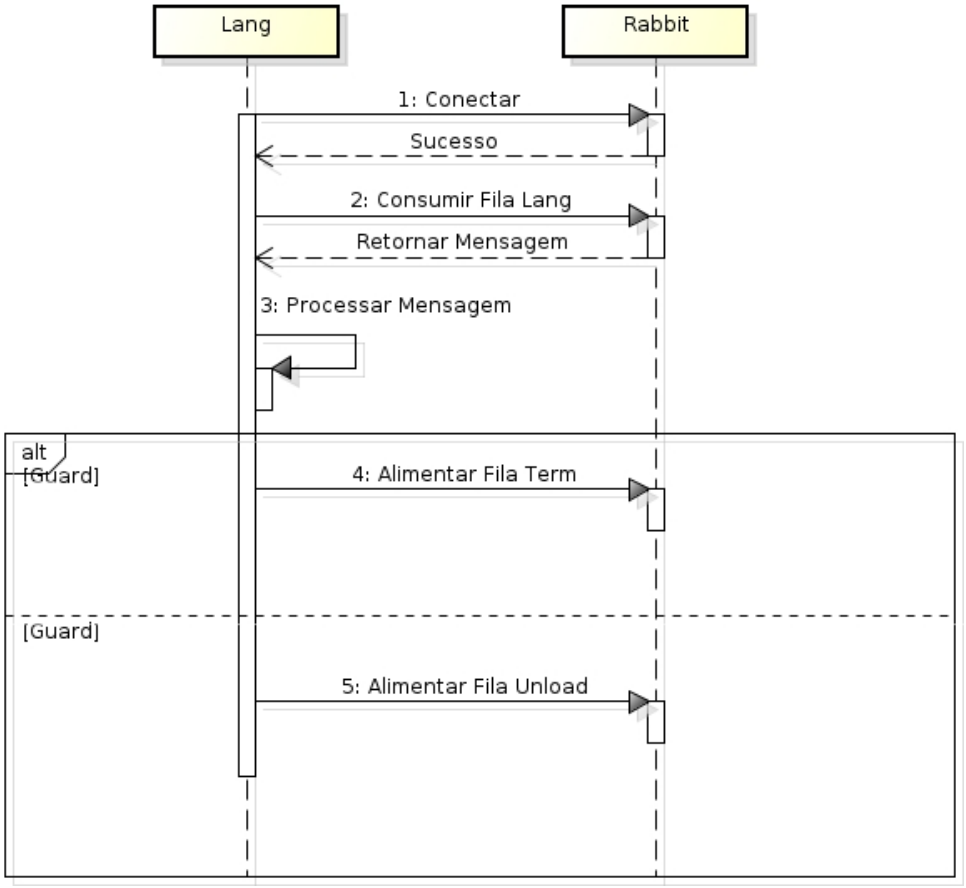


Figura 6.4: Diagrama de sequência do processo Lang

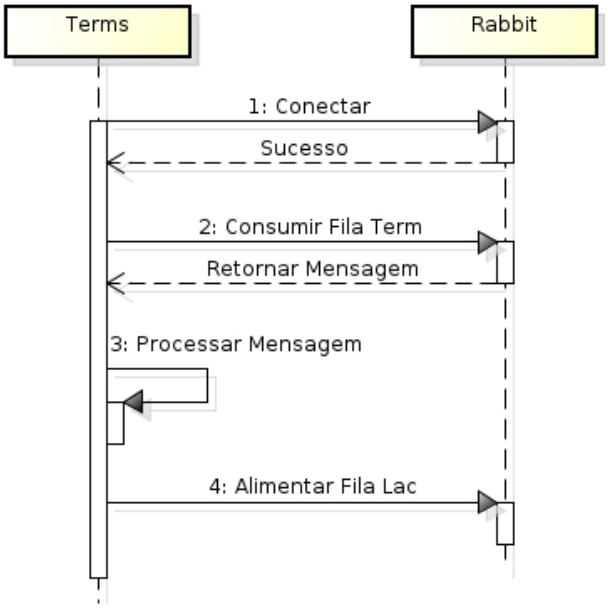


Figura 6.5: Diagrama de sequência do processo Terms

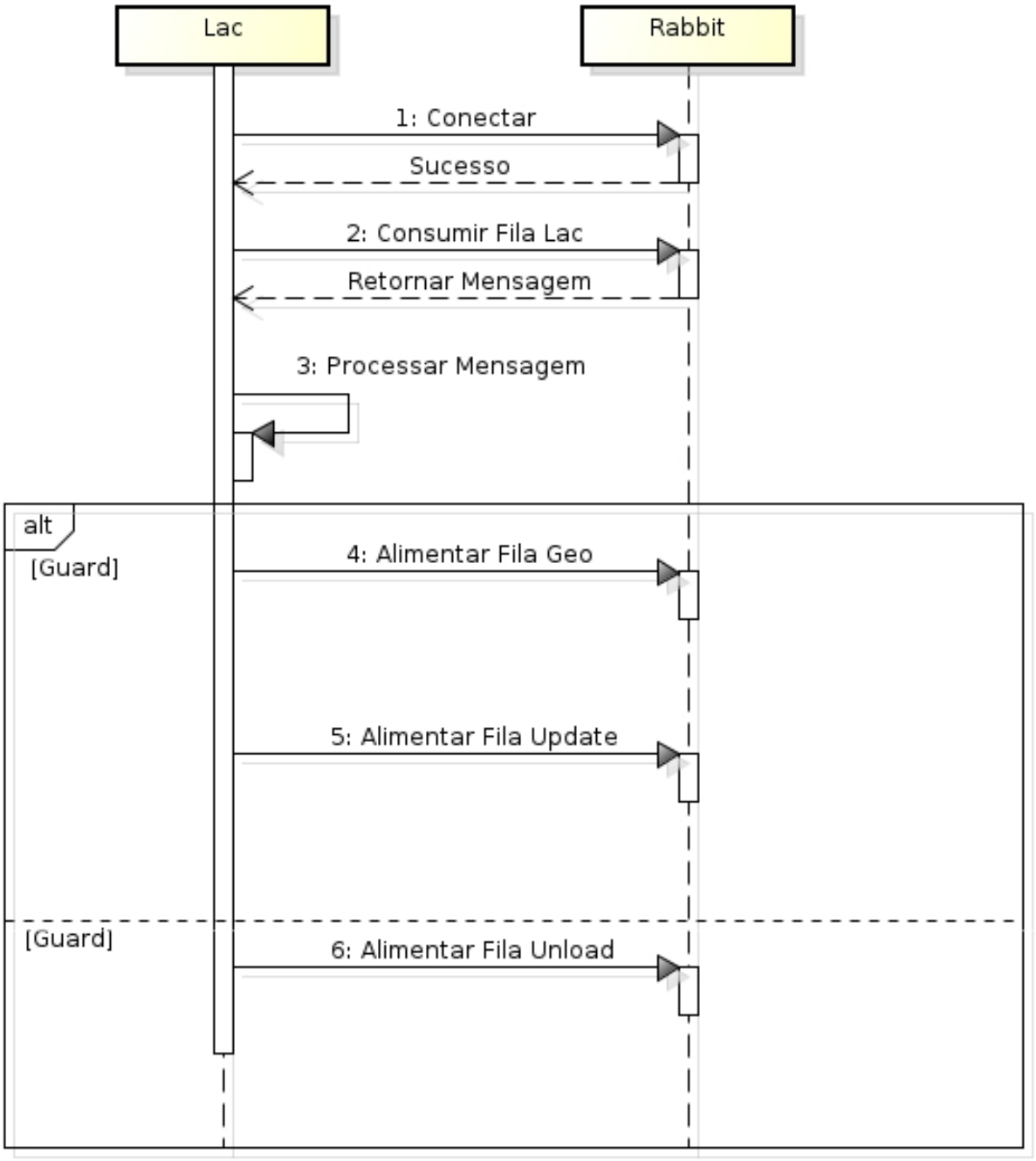


Figura 6.6: Diagrama de sequência do processo Lac

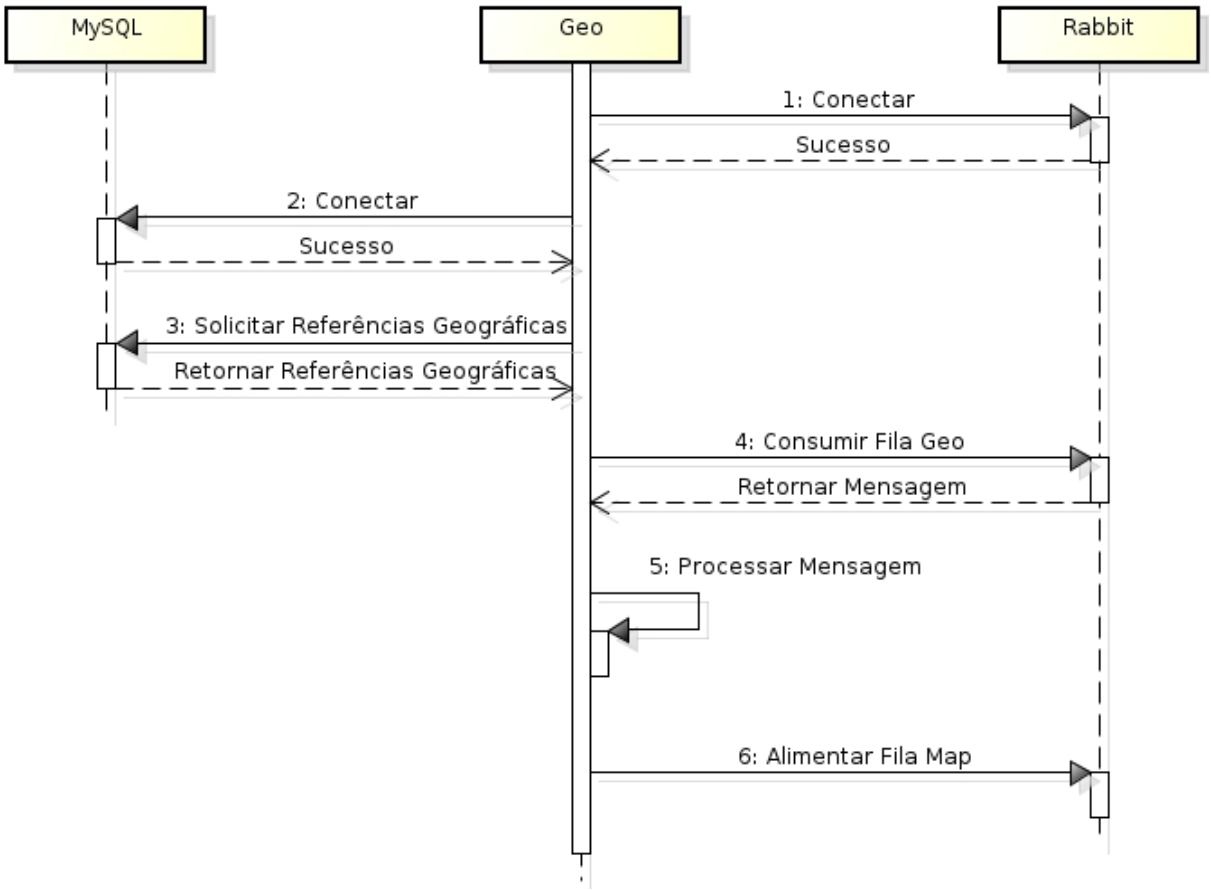


Figura 6.7: Diagrama de sequência do processo Geo

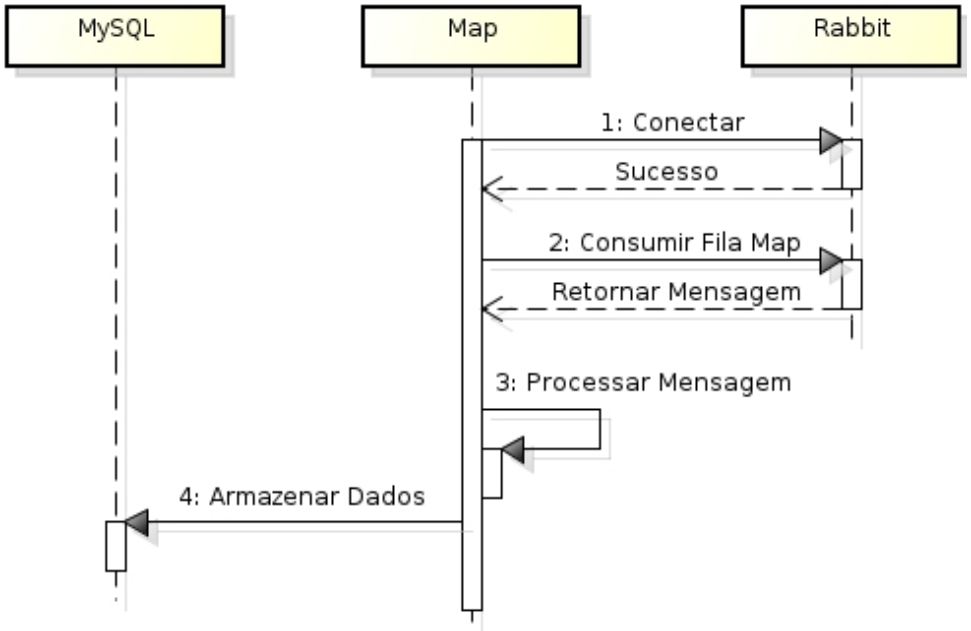


Figura 6.8: Diagrama de sequência do processo Map

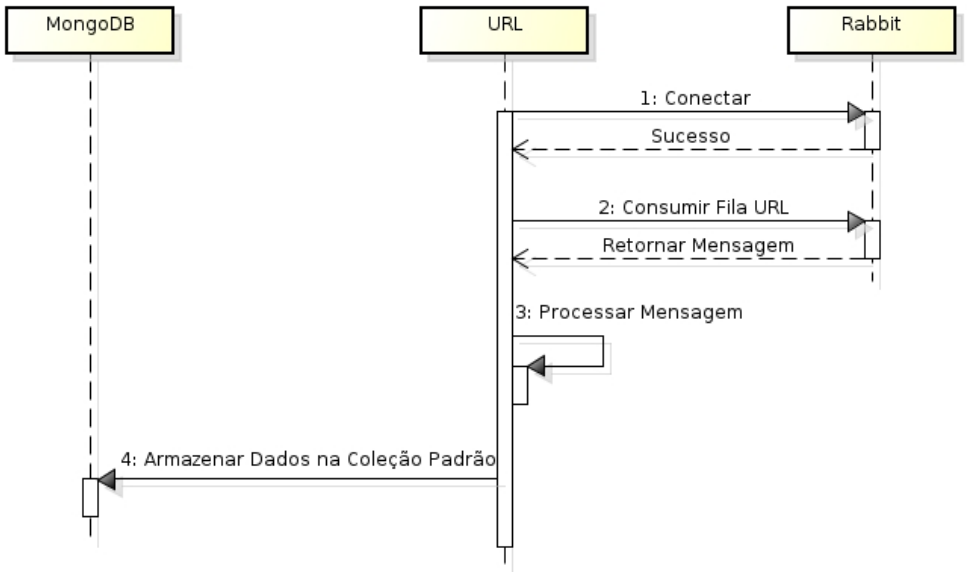


Figura 6.9: Diagrama de sequência do processo URL

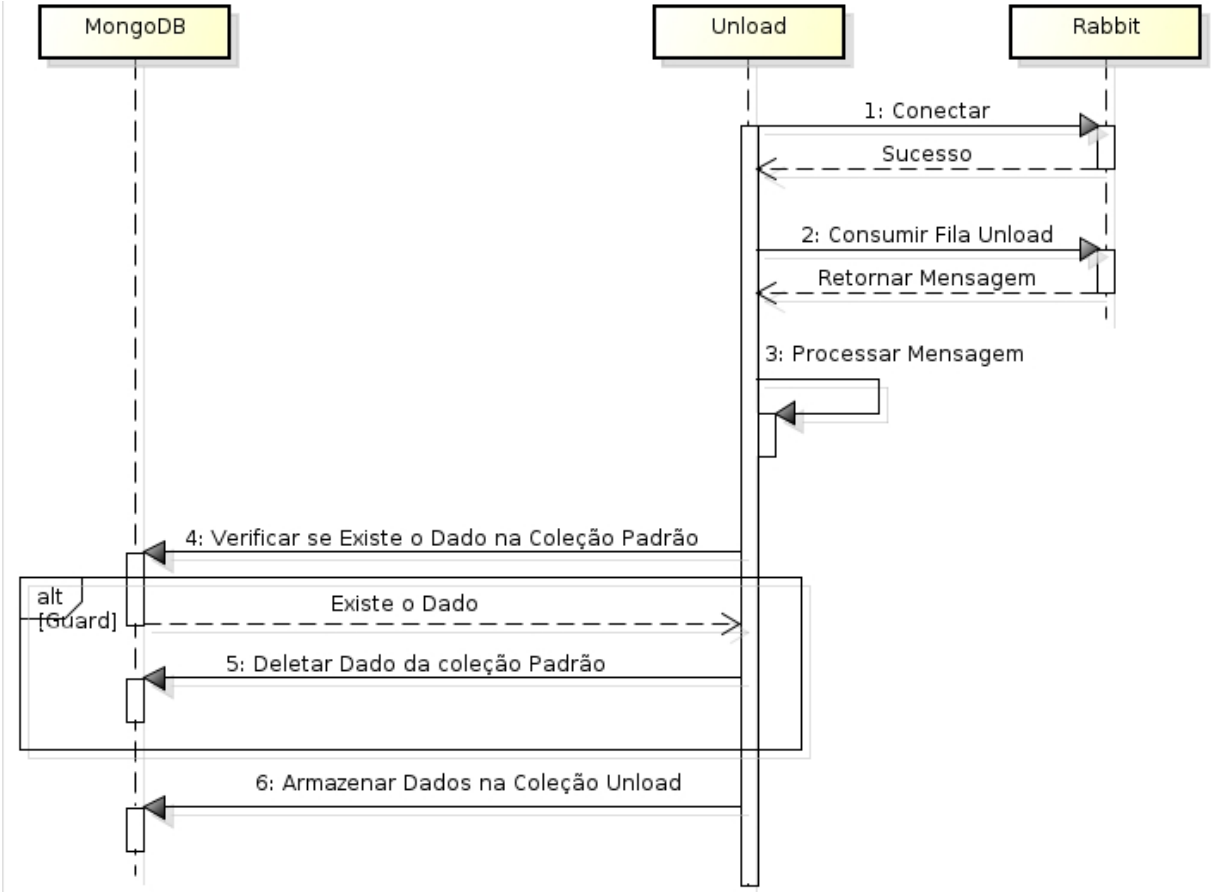


Figura 6.10: Diagrama de sequência do processo Unload

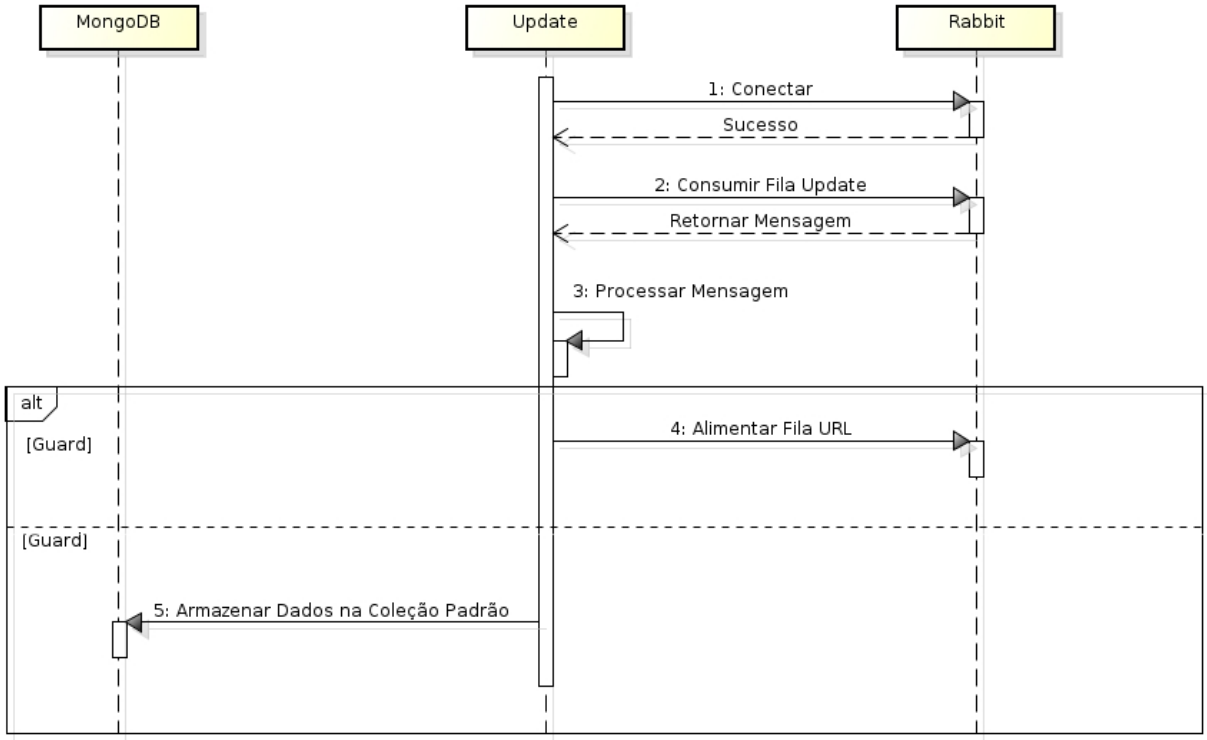


Figura 6.11: Diagrama de sequência do processo Update

6.5 Previsão e classificação funcional

6.5.1 Instrumentações no Observatório da Dengue

Abaixo são expostos os eventos de mensagens nos componentes (identificados com **counter**) e de **erro** que podem ocorrer durante a execução do sistema Observatório da Dengue.

1. Coletor_twitter:

    - MySQL Conexao erro.
    - Twitter Coleta inativa erro.
    - Twitter Track.
    - *Falha ColetaTwitter.**
    - *Falha Conexao MySQL.**
    - *TwitterTrack.**

    - Opcao Verificacao erro.
    - Twitter Coleta desconectado erro.
    - Twitter Coleta naodesconectado erro.
    - Twitter Track parada erro.
    - Twitter Location.
    - Twitter Follow.

- Twitter Follow parada erro.
- Twitter Coleta tipo erro.
- Twitter Coleta configuracao erro.

(a) __init__:

- counter RabbitMQListener.
- counter MongoOutListener.
- MongoDB Offline erro.
- MongoDB atualizar erro.
- RabbitMQ Offline erro.
- RabbitMQ Arquivo backup erro.
- *Falha RabbitMQBackupFile.**
- *Falha RabbitMQOffline.**

- counter FileOutListener.
- Twitter Coleta erro *status*.
- Twitter Coleta timeout erro *status*.
- MongoDB Offline reconectar erro.

2. Reader_filter:

- counter Language Reader *sigla da língua*.
- MongoDB Conexao erro.
- EncodingErrors.
- LimiteColeta limit erro.
- Desconectado desconectado erro.
- *Desconectado disconnect.**
- *LimiteColeta limit.**

- Arquivo Leitura erro.
- Filas Criacao erro.
- RabbitMQ Conexao erro.

3. Lang_filter:

- counter Language LANG *sigla da língua*.

- Contexto Ignora erro.

4. Terms_filter:

- counter Language TERM *sigla da língua*.

- Termo Ignora erro *tipo*.
- Contexto Ignora erro.

5. Lac_filter:

    - counter Language LAC *sigla da língua*.
    - counter SPAM False.
    - counter SPAM True.

    - Limite Minimo erro *status*.
    - Contexto Ignora erro.

6. Geo_filter:

    - counter Language GEO *sigla da língua*.
    - counter Localizacao True.
    - counter Localizacao False.
    - counter Localizacao vazia False.
    - MySQL Conexao erro.
    - RabbitMQ Fila Consumo erro.
    - *Falha Fila RabbitMQ*.*
    - *counter Localization True*.*
    - *counter Localization False*.*
    - *Falha Conexao MySQL*.*
    - *Falha EmptyLocalization False*.*

    - Localizacao Perdida erro.
    - Mensagem descartada erro.

7. Map_filter:

    - counter Language MAP *sigla da língua*.

    - MySQL Conexao erro.
    - MySQL Salvar erro.

8. URL_filter:

    - counter Language URL *sigla da língua*.
    - Timeout erro.
    - Geral impressa True erro.
    - RabbitMQ Fila Consumo erro.
    - *TimeoutUrl*.*
    - *ErroGeralURL*.*
    - *Falha Fila RabbitMQ*.*

- Endereco Incompleto erro.
- Dominio Permissao erro *codigo*.
- Geral impressa False erro.
- Geral gaierror erro.

9. Update_filter:

    - counter Language UPDATE *sigla da língua*.
    - counter NewURL False.
    - counter NewURL True.

    - MongoDB Conexao erro.
    - MongoDB Colecao erro.
    - MongoDB Url Colecao erro.
    - MongoDB Update Colecao True erro.
    - MongoDB Update Colecao False erro.
    - MongoDB Url Insert Colecao erro.
    - Contexto Ignora erro.

10. Unload_filter:

    - counter Language UNLOAD *sigla da língua*.
    - RabbitMQ Fila Consumo erro.
    - *Falha Fila RabbitMQ*.*

    - MongoDB Conexao erro.
    - MongoDB Colecao erro.
    - MongoDB Unload Colecao erro.
    - MongoDB Update Unload Colecao erro.
    - MongoDB Insert Unload Colecao erro.
    - MongoDB Remove Colecao erro.

O * são para as mensagens que foram alteradas no tempo delimitado para avaliação.

6.5.2 Identificação das falhas e dos defeitos

1. Coletor_twitter:

    - **falha:** indisponibilidade do banco de dados MySQL.
    - **erro:** MySQL Conexao ou *Falha Conexao MySQL*.
    - **defeito:** não há defeito, se propaga em outro erro.

- **falha:** indisponibilidade da informação para coleta dos dados na rede social.
- **erro:** Twitter Coleta inativa ou *Falha ColetaTwitter*.
- **defeito:** interrupção de serviço do processo.

- **falha:** indisponibilidade dos dados.
- **erro:** Opcao Verificacao.
- **defeito:** interrupção de serviço do processo.

- **falha:** não identificada.
- **erro:** Twitter Coleta desconectado.
- **defeito:** não identificado.

- **falha:** não identificada.
- **erro:** Twitter Coleta naodesconectado.
- **defeito:** não identificado.

- **falha:** não prevista
- **erro:** Twitter Track parada.
- **defeito:** interrupção de serviço do processo.

- **falha:** não identificada.
- **erro:** Twitter Follow parada.
- **defeito:** interrupção de serviço do processo.

- **falha:** informação corrompida do banco de dados MySQL.
- **erro:** Twitter Coleta tipo.
- **defeito:** interrupção de serviço do processo.

- **falha:** informação corrompida do banco de dados MySQL.
- **erro:** Twitter Coleta configuracao.
- **defeito:** interrupção de serviço do processo.

- __init__:
  - **falha:** indisponibilidade do banco de dados MongoDB.
  - **erro:** MongoDB Offline.
  - **defeito:** não há defeito, se propaga em outro erro.

  - **falha:** indisponibilidade do banco de dados MongoDB.
  - **erro:** MongoDB Offline reconectar.

- **defeito:** não há defeito, se propaga em outro erro.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB atualizar.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do middleware RabbitMQ.
- **erro:** RabbitMQ Offline ou *Falha RabbitMQOffline*.
- **defeito:** não entrega o serviço para o RabbitMQ.

- **falha:** ausência do arquivo de backup.
- **erro:** RabbitMQ Arquivo backup ou *Falha RabbitMQBackupFile*.
- **defeito:** não salvou o dado coletado no arquivo de backup.

- **falha:** não identificada.
- **erro:** Twitter Coleta *status*.
- **defeito:** não identificado.

- **falha:** não identificada.
- **erro:** Twitter Coleta timeout *status*.
- **defeito:** não identificado.

2. Reader_filter:

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Conexao.
- **defeito:** não há defeito, se propaga em outro erro.

- **falha:** dado não codificado da maneira correta.
- **erro:** EncodingErrors.
- **defeito:** não entrega o serviço para o filtro seguinte.

- **falha:** *mensagem* com *limit*, indicando que o twitter limita a quantidade de mensagens.
- **erro:** LimiteColeta limite ou *LimiteColeta limit*.
- **defeito:** não entrega o serviço para o filtro seguinte.

- **falha:** *mensagem* com *disconnect*, indicando que o twitter está desconectado.
- **erro:** Desconectado desconectado ou *Desconectado disconnect*.
- **defeito:** não entrega o serviço para o filtro seguinte.

- **falha:** indisponibilidade do arquivo.
- **erro:** Arquivo Leitura.

- **defeito:** não identificado.

- **falha:** indisponibilidade do middleware RabbitMQ.
- **erro:** RabbitMQ Conexao.
- **defeito:** não há defeito, se propaga em outro erro.

- **falha:** indisponibilidade do middleware RabbitMQ.
- **erro:** Filas Criacao.
- **defeito:** não entrega o serviço para o RabbitMQ.

3. Lang_filter:

   - **falha:** ou não existe a palavra chave ou não existe arquivo com a palavra chave ou RabbitMQ indisponível.
   - **erro:** Contexto Ignora.
   - **defeito:** não entrega o serviço para o filtro seguinte.

4. Terms_filter:

   - **falha:** não existe a palavra chave ou o arquivo com a palavra chave.
   - **erro:** Termo Ignora *tipo*.
   - **defeito:** não entrega o serviço para o filtro seguinte.

   - **falha:** ou não existe a palavra chave ou não existe arquivo com a palavra chave ou RabbitMQ indisponível.
   - **erro:** Contexto Ignora.
   - **defeito:** não entrega o serviço para o filtro seguinte.

5. Lac_filter:

   - **falha:** não identificada.
   - **erro:** Limite Minimo *status*.
   - **defeito:** não identificado.

   - **falha:** ou não existe a palavra chave ou não existe arquivo com a palavra chave ou RabbitMQ indisponível.
   - **erro:** Contexto Ignora.
   - **defeito:** não entrega o serviço para o filtro seguinte.

6. Geo_filter:

   - **falha:** indisponibilidade do banco de dados MySQL.
   - **erro:** MySQL Conexao ou *Falha Conexao MySQL*.

- **defeito:** não há defeito, se propaga em outro erro.

- **falha:** indisponibilidade do banco de dados MySQL.
- **erro:** MySQL Conexao novamente ou *Falha Conexao MySQL.*
- **defeito:** não há defeito, se propaga em outro erro.

- **falha:** não conseguiu verificar, no banco de dados MySQL, se há a localização descrita na mensagem.
- **erro:** RabbitMQ Fila Consumo ou *Falha Fila RabbitMQ* (base_filter: classe virtual).
- **defeito:** não entrega o serviço para o filtro seguinte (no caso, filtro map).

- **falha:** não identificada.
- **erro:** Localizacao Perdida.
- **defeito:** não identificado.

- **falha:** não identificada.
- **erro:** Mensagem descartada.
- **defeito:** não entrega o serviço para o filtro seguinte.

7. Map_filter:

   - **falha:** indisponibilidade do banco de dados MySQL.
   - **erro:** MySQL Conexao.
   - **defeito:** não há defeito, se propaga em outro erro.

   - **falha:** indisponibilidade do banco de dados MySQL.
   - **erro:** MySQL Salvar.
   - **defeito:** não consegue salvar, no banco de dados MySQL, a localização descrita na mensagem.

8. Url_filter:

   - **falha:** não identificada.
   - **erro:** Timeout ou *TimeoutUrl*.
   - **defeito:** não entrega o serviço para o banco de dados MongoDB.

   - **falha:** não identificada.
   - **erro:** Geral impressa True ou *ErroGeralURL.*
   - **defeito:** não entrega o serviço correto para o banco de dados MongoDB.

   - **falha:** indisponibilidade do banco de dados MongoDB.

- **erro:** RabbitMQ Fila Consumo ou *Falha Fila RabbitMQ*.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** leitura incompleta da url.
- **erro:** Endereco Incompleto.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** não há permissão para acesso à essa url.
- **erro:** Dominio Permissao *codigo*.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** não identificada.
- **erro:** Geral impressa False.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** não identificada.
- **erro:** Geral gaierror.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

9. Update_filter:

    - **falha:** indisponibilidade do banco de dados MongoDB.
    - **erro:** MongoDB Conexao.
    - **defeito:** não entrega o serviço para o banco de dados MongoDB.

    - **falha:** indisponibilidade do banco de dados MongoDB.
    - **erro:** MongoDB Colecao.
    - **defeito:** não entrega o serviço para o banco de dados MongoDB.

    - **falha:** indisponibilidade do banco de dados MongoDB.
    - **erro:** MongoDB Url Colecao.
    - **defeito:** não entrega o serviço para o banco de dados MongoDB.

    - **falha:** indisponibilidade do banco de dados MongoDB.
    - **erro:** MongoDB Update Colecao True.
    - **defeito:** não entrega o serviço para o banco de dados MongoDB.

    - **falha:** indisponibilidade do banco de dados MongoDB.
    - **erro:** MongoDB Update Colecao False.
    - **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Url Insert Colecao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** ou não existe a palavra chave ou não existe arquivo com a palavra chave ou RabbitMQ indisponível.
- **erro:** Contexto Ignora.
- **defeito:** não entrega o serviço para o filtro seguinte.

10. Unload_filter:

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** RabbitMQ Fila Consumo ou *Falha Fila RabbitMQ*.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Conexao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Colecao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Unload Colecao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Update Unload Colecao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Insert Unload Colecao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

- **falha:** indisponibilidade do banco de dados MongoDB.
- **erro:** MongoDB Remove Colecao.
- **defeito:** não entrega o serviço para o banco de dados MongoDB.

6.5.3 Classificação das falhas identificadas

Falhas previstas no **Coletor_twitter**:

1. Indisponibilidade do banco de dados MySQL:
    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.
2. Indisponibilidade da informação para coleta dos dados na rede social:
    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.
3. Indisponibilidade dos dados:
    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.
4. Informação corrompida do banco de dados MySQL:
    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.

- **Objetivo:** Não maliciosa.
- **Intenção:** Não deliberada.
- **Capacidade:** Acidental.
- **Persistência:** Persistente.

5. Informação corrompida do banco de dados MySQL:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.

6. Indisponibilidade do banco de dados MongoDB:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.

7. Indisponibilidade do middleware RabbitMQ:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.

8. Ausência do arquivo de backup:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.

- **Causa:** Humana.
- **Dimensão:** Software/Hardware.
- **Objetivo:** Não maliciosa.
- **Intenção:** Não deliberada.
- **Capacidade:** Acidental.
- **Persistência:** Persistente.

Falhas previstas no **Reader_filter**:

1. Indisponibilidade do banco de dados MongoDB.

   - **Fase de criação:** Operacional.
   - **Localização:** Externa.
   - **Causa:** Natural/Humana.
   - **Dimensão:** Software/Hardware.
   - **Objetivo:** Não maliciosa.
   - **Intenção:** Não deliberada.
   - **Capacidade:** Acidental.
   - **Persistência:** Persistente.

2. Dado não codificado da maneira correta.

   - **Fase de criação:** Desenvolvimento.
   - **Localização:** Interna.
   - **Causa:** Humana.
   - **Dimensão:** Software.
   - **Objetivo:** Não maliciosa.
   - **Intenção:** Não deliberada.
   - **Capacidade:** Acidental.
   - **Persistência:** Persistente.

3. *Mensagem* com *limit*, indicando que o twitter limita a quantidade de mensagens:

   - **Fase de criação:** Desenvolvimento.
   - **Localização:** Interna.
   - **Causa:** Humana.
   - **Dimensão:** Software.
   - **Objetivo:** Não maliciosa.
   - **Intenção:** Não deliberada.
   - **Capacidade:** Acidental.
   - **Persistência:** Persistente.

4. *Mensagem* com *disconnect*, indicando que o twitter está desconectado:

    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.

5. Indisponibilidade do arquivo:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

6. Indisponibilidade do middleware RabbitMQ.

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

Falhas previstas no **Lang_filter**:

1. Não existe arquivo com a palavra chave:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.

- **Intenção:** Não deliberada.
- **Capacidade:** Acidental.
- **Persistência:** Transitória.

2. Não existe a palavra chave:

    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.

3. Indisponibilidade do middleware RabbitMQ:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

Falhas previstas no **Terms_filter**:

1. Não existe arquivo com a palavra chave:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

2. Não existe a palavra chave:

    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.

- **Causa:** Humana.
- **Dimensão:** Software.
- **Objetivo:** Não maliciosa.
- **Intenção:** Não deliberada.
- **Capacidade:** Acidental.
- **Persistência:** Persistente.

3. Indisponibilidade do middleware RabbitMQ:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

Falhas previstas no **Lac_filter**:

1. Não existe arquivo com a palavra chave:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

2. Não existe a palavra chave:

    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Persistente.

3. Indisponibilidade do middleware RabbitMQ:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

Falhas previstas no **Geo_filter**:

1. Indisponibilidade do banco de dados MySQL:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

2. Não conseguiu verificar, no banco de dados MySQL, se há a localização descrita na mensagem:

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

Falhas previstas no **Map_filter**:

1. Indisponibilidade do banco de dados MySQL.

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.

- **Dimensão:** Software/Hardware.
- **Objetivo:** Não maliciosa.
- **Intenção:** Não deliberada.
- **Capacidade:** Acidental.
- **Persistência:** Transitória.

Falhas previstas no **Url_filter**:

1. Indisponibilidade do banco de dados MongoDB.

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

2. Leitura incompleta da url.

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

3. Não há permissão para acesso à essa url.

    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

Falhas previstas no **Update_filter**:

1. Indisponibilidade do banco de dados MongoDB.
    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.
2. Não existe arquivo com a palavra chave:
    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.
3. Não existe a palavra chave:
    - **Fase de criação:** Desenvolvimento.
    - **Localização:** Interna.
    - **Causa:** Humana.
    - **Dimensão:** Software.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.
4. Indisponibilidade do middleware RabbitMQ:
    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.

- **Persistência:** Transitória.

Falhas previstas no **Unload_filter**:

1. Indisponibilidade do banco de dados MongoDB.
    - **Fase de criação:** Operacional.
    - **Localização:** Externa.
    - **Causa:** Natural/Humana.
    - **Dimensão:** Software/Hardware.
    - **Objetivo:** Não maliciosa.
    - **Intenção:** Não deliberada.
    - **Capacidade:** Acidental.
    - **Persistência:** Transitória.

6.5.4 Classificação dos defeitos identificados

Defeitos previstos no **Coletor_twitter**:

1. Interrupção de serviço do processo:
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Consistente.
    - **Consequência:** Grave.
2. Não entrega o serviço para o banco de dados MongoDB:
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.
    - **Consequência:** Grave.
3. Não entrega o serviço para o RabbitMQ:
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Consistente.
    - **Consequência:** Grave.
4. Não entrega o serviço para o arquivo de backup:
    - **Domínio:** Instável.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.

- **Consequência:** Pequena.

Defeitos previstos no **Reader_filter**:

1. Não entrega o serviço para o filtro seguinte (*limite*):
    - **Domínio:** Instável.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Consistente.
    - **Consequência:** Média.
2. Não entrega o serviço para o filtro seguinte (*desconectado*):
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Consistente.
    - **Consequência:** Média.
3. Não entrega o serviço para o filtro seguinte (*EncodingErrors*):
    - **Domínio:** Conteúdo.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Consistente.
    - **Consequência:** Média.
4. Não entrega o serviço para o RabbitMQ:
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Consistente.
    - **Consequência:** Grave.

Defeitos previstos no **Lang_filter**:

1. Não entrega o serviço para o filtro seguinte:
    - **Domínio:** Conteúdo.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.
    - **Consequência:** Grave.

Defeitos previstos no **Terms_filter**:

1. Não entrega o serviço para o filtro seguinte:
    - **Domínio:** Parada.

- **Detectabilidade:** Sinalizada.
- **Consistência:** Inconsistente.
- **Consequência:** Grave.

Defeitos previstos no **Lac_filter**:

1. Não entrega o serviço para o filtro seguinte:
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.
    - **Consequência:** Grave.

Defeitos previstos no **Geo_filter**:

1. Não entrega o serviço para o filtro seguinte (*RabbitMQ Fila Consumo*):
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.
    - **Consequência:** Grave.
2. Não entrega o serviço para o filtro seguinte (*Mensagem Descartada*):
    - **Domínio:** Conteúdo.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.
    - **Consequência:** Pequena.

Defeitos previstos no **Map_filter**:

1. Não consegue salvar, no banco de dados MySQL, a localização descrita na mensagem:
    - **Domínio:** Parada.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.
    - **Consequência:** Grave.

Defeitos previstos no **Url_filter**:

1. Não entrega o serviço para o banco de dados MongoDB (*Timeout*):
    - **Domínio:** Tempo.
    - **Detectabilidade:** Sinalizada.
    - **Consistência:** Inconsistente.

- **Consequência:** Pequena.

2. Não entrega o serviço para o banco de dados MongoDB (*RabbitMQ Fila Consumo*):

   - **Domínio:** Parada.
   - **Detectabilidade:** Sinalizada.
   - **Consistência:** Inconsistente.
   - **Consequência:** Pequena.

3. Não entrega o serviço para o banco de dados MongoDB (*Endereco Incompleto*).

   - **Domínio:** Conteúdo.
   - **Detectabilidade:** Sinalizada.
   - **Consistência:** Inconsistente.
   - **Consequência:** Pequena.

4. Não entrega o serviço correto para o banco de dados MongoDB (*Geral impressa True, Geral impressa True, Geral gaierror e Dominio Permissao*):

   - **Domínio:** Instável.
   - **Detectabilidade:** Sinalizada.
   - **Consistência:** Inconsistente.
   - **Consequência:** Pequena.

Defeitos previstos no **Update_filter**:

1. Não entrega o serviço para o banco de dados MongoDB:

   - **Domínio:** Parada.
   - **Detectabilidade:** Sinalizada.
   - **Consistência:** Inconsistente.
   - **Consequência:** Pequena.

2. Não entrega o serviço para o filtro seguinte.

   - **Domínio:** Parada.
   - **Detectabilidade:** Sinalizada.
   - **Consistência:** Inconsistente.
   - **Consequência:** Pequena.

Defeitos previstos no **Unload_filter**:

1. Não entrega o serviço para o banco de dados MongoDB:

   - **Domínio:** Parada.
   - **Detectabilidade:** Sinalizada.
   - **Consistência:** Inconsistente.
   - **Consequência:** Pequena.

6.5.5 Análise realizada das falhas e defeitos encontrados nos componentes do Observatório da Dengue

coletor_twitter

1. Coletor_twitter tenta se conectar ao banco de dados MySQL e não consegue;
2. como não houve resposta do banco de dados, considera-se uma indisponibilidade das informações necessárias para coleta dos dados na rede social;
3. esta indisponibilidade se propaga em um outro erro, onde foi prevista a não coleta de nenhum dado da rede social alvo;
4. como não ocorreu nenhuma coleta, não há envio de dados para o middleware RabbitMQ.

É possível visualizar três sistemas nesta descrição:

1. banco de dados MySQL,
2. coletor_twitter e
3. rede social.

As informações estão indisponíveis pelo fato de elas estarem no sistema **banco de dados MySQL** e, sempre que houver uma tentativa de coleta massiva de dados da rede social, haverá um acesso à ele buscando a informação necessária para o sucesso da coleta. A falha foi observada no sistema coletor_twitter, pois é uma **falha dormente**, ou seja, que poderia se tornar ativa por uma **falha externa**. É uma **vulnerabilidade** que existe no sistema coletor_twitter. A falha externa, ocorrida no sistema banco de dados MySQL, pode ter sido um cabo que se soltou, um pico de energia ou uma reinicialização do sistema.

O defeito em relação ao não envio de mensagem para o RabbitMQ não ocorreu devido à um erro propagado na **interface de serviço** da rede social para a **interface do usuário** coletor_twitter, mas sim devido ao detectado na **interface de serviço** do sistema **banco de dados MySQL** com o **usuário** em questão, ou seja, o coletor_twitter.

No módulo **__init__** há a implementação de algumas classes que são utilizadas pelo coletor_twitter, sendo descrita a interação da seguinte forma:

1. ocorre a tentativa de conexão ao middleware RabbitMQ;
2. como não houve resposta do middleware, considera-se uma indisponibilidade do mesmo;
3. após o defeito apresentado na **interface de serviço** do middleware, ocorre a tentativa de conexão com o sistema banco de dados MongoDB;
4. como não houve resposta do banco de dados, considera-se uma indisponibilidade do mesmo pelo fato de não conseguir atualizar e o processo coletor_twitter é interrompido.

É possível visualizar três sistemas nesta descrição:

1. middleware RabbitMQ,
2. banco de dados MongoDB e
3. processo coletor_twitter.

Há a tentativa de conexão, primeiramente, com o sistema middleware RabbitMQ para que os dados coletados sejam colocados diretamente nas filas gerenciadas pelo mesmo. Como ocorreu um erro que se propagou para a **interface de serviço** do RabbitMQ, na **interface do usuário** coletor_twitter não ocorreu a **entrega do serviço**. Com isso, ocorreu uma outra tentativa com o objetivo de entregar o serviço fornecido pela rede social, mas agora para o sistema banco de dados MongoDB. Não foi recebido o serviço na **interface do usuário** coletor_twitter, com isso, o dado foi perdido.

Em relação à última falha prevista neste módulo, observamos que ela sempre vinha acompanhada da indisponibilidade do middleware RabbitMQ. Logo, julgamos que seja uma falha no desenvolvimento do algoritmo. Porém, uma outra falha poderia ser a ausência real do arquivo de *backup*, seguindo o raciocínio de que, se ocorrer a coleta do dado:

1. tenta salvá-lo no middleware RabbitMQ;
2. senão, tenta salvá-lo no arquivo de *backup*;
3. senão, tenta salvá-lo no banco de dados MongoDB.

reader_filter

1. O processo reader_filter pega a mensagem do filtro de nome semelhante para processá-la;
2. é verificado se a palavra está decodificada corretamente, caso contrário, ocorre um erro chamado de *Encoding Errors*.
3. é detectada a palavra *limit* na mensagem recebida, indicando que alcançou o limite de dados que podem ser coletados pela API da rede social alvo ou a palavra *disconnect*, indicando que a rede social foi desconectada durante a coleta do dado;
4. como a mensagem não está completa, o serviço não é entregue para o processo seguinte.

É possível visualizar três sistemas nesta descrição:

1. rede social,
2. middleware RabbitMQ e
3. processo reader_filter.

Logo que a mensagem é percebida pelo processo, há a verificação do tipo de codificação da mensagem e, no caso de estar incorreta, não prossegue o processamento. Tendo o processamento seguido normalmente, ocorre a verificação da existência das palavras *limit* ou *disconnect*. Caso uma destas palavras seja encontrada, o serviço não é entregue para o processo seguinte. *Ou seja, a rede social não entregou o dado de maneira correta por uma limitação da API disponibilizada e ocorreu uma interrupção do serviço da rede social*. Olhando apenas os eventos gerados pela Ferramenta de Inspeção chegaríamos a esta conclusão, porém sempre que ocorre uma reinicialização do processo reader_filter, ocorre erro de **limite**, **desconectado** e *EncodingErrors*.

Com esta análise, concluímos que se trata de uma **falha dormente** ativada pela reinicialização do processo reader_filter, ou seja, a **falha externa** é a interrupção do processo por algum motivo humano diverso ou pela **interrupção do serviço** do middleware RabbitMQ.

geo_filter

1. O processo geo_filter tenta se conectar ao banco de dados MySQL;
2. caso não consiga, realiza outra tentativa de conexão com o banco de dados MySQL;
3. quando, novamente, não há conexão com o banco de dados MySQL, é detectado o erro propagado **interface de serviço** do banco na **interface do usuário** geo_filter;
4. com a indisponibilidade do banco de dados MySQL, ocorre um erro na fila gerenciada pelo middleware RabbitMQ, pois não foi possível verificar no banco de dados se há ou não uma localização geográfica disponível.

É possível visualizar três sistemas nesta descrição:

1. banco de dados MySQL,
2. middleware RabbitMQ e
3. processo geo_filter.

As informações das localizações geográficas estão indisponíveis pelo fato de elas estarem no sistema **banco de dados MySQL** e, sempre que houver uma tentativa de verificar se existe a localização, haverá um acesso à ele para que seja possível enviar a mensagem para a próxima etapa. A falha foi observada no sistema geo_twitter, pois é uma **falha dormente**, ou seja, que poderia se tornar ativa por uma **falha externa**. É uma **vulnerabilidade** que existe no sistema geo_twitter. A falha externa, ocorrida no sistema banco de dados MySQL, pode ter sido um cabo que se soltou, um pico de energia ou uma reinicialização do sistema. Com a não confirmação da existência de uma localização geográfica, a mensagem fica na espera para ser processada.

url_filter

1. O processo url_filter expande as URLs existentes, porém as vezes o tempo para expansão não é suficiente;
2. as vezes o problema não é de tempo, mas sim a formação da mesma;
3. com a indisponibilidade do banco de dados MongoDB, ocorre um erro na fila gerenciada pelo middleware RabbitMQ, pois não foi possível atualizar o banco de dados.

É possível visualizar três sistemas nesta descrição:

1. banco de dados MongoDB,
2. middleware RabbitMQ e
3. processo url_filter.

A mensagem recebida pelo processo geo_filter não conseguiu expandir totalmente no tempo delimitado, tornando a *falha dormente* ativa no processo, que ativou um erro e se propagou para a **interface de serviço** do processo, não entregando o serviço para o banco de dados MongoDB. Quando o erro é geral, ou seja, da formação da mensagem, o erro também se propaga

para a interface de serviço e não entrega o serviço para o banco de dados MongoDB. Porém, quando ocorre indisponibilidade do banco de dados MongoDB, a fila gerenciada pelo middleware RabbitMQ de nome semelhante ao processo não pode ser consumida, fazendo com que o processo fique se repetindo neste ponto e que a Ferramenta de Inspeção produza vários eventos informando que a mensagem não foi consumida.

Em relação ao tempo cedido para a expansão e à formação da mensagem, são defeitos que dificilmente ocorrem, variando de uma à três vezes por dia, no máximo. Já a indisponibilidade do banco de dados MongoDB ocorre na **interface de serviço** do mesmo, que ativa a antes **falha dormente** do processo geo_filter, impossibilitando o consumo da mensagem da fila gerenciada pelo middleware RabbitMQ, não ocorrendo, assim, a atualização no banco de dados.

unload_filter

1. Com a indisponibilidade do banco de dados MongoDB, não atualiza a coleção *unload* e não retira os dados da coleção *padrão*.

A indisponibilidade do banco de dados MongoDB ocorre na **interface de serviço** do mesmo, tornando a **falha dormente** do processo unload_filter ativa, impossibilitando o consumo da mensagem da fila gerenciada pelo middleware RabbitMQ, não ocorrendo, assim, a atualização no banco de dados.

6.6 Quantidade de mensagens recebidas por cada componente da sequência

1. **Coletor_twitter:**
   - Quantidade total de dados coletados: 23875;
     - Dados enviados ao RabbitMQ: 17680.
     - Dados enviados ao MongoDB: 6195.
2. **Reader_filter:**
   - Quantidade de dados recebidos: 17681.
   - Quantidade de dados enviados para o filtro lang: 17681.
3. **Lang_filter:**
   - Quantidade de dados recebidos: 17681.
     - Quantidade de dados enviados para o filtro unload: 14229.
     - Quantidade de dados enviados para o filtro terms: 3452.
4. **Terms_filter:**
   - Quantidade de dados recebidos: 3452.
     - Quantidade de dados enviados para o filtro lac: 3452.
5. **Lac_filter:**

- Quantidade de dados recebidos: 3452.
  - Quantidade de dados enviados para os filtros geo e update: 3405.
  - Quantidade de dados enviados para o filtro unload: 47.

6. **Geo_filter:**

   - Quantidade de dados recebidos: 3428.
     - Quantidade de dados enviados para o filtro map: 2030.
     - Quantidade de dados sem localização geográfica: 1052.
     - Quantidade de dados sem localização geográfica no banco de dados MySQL: 326.

7. **Map_filter:**

   - Quantidade de dados recebidos: 2030.

8. **Update_filter:**

   - Quantidade de dados recebidos: 3408.
     - Quantidade de dados enviados para o filtro url: 754.
     - Quantidade de dados enviados para o banco de dados MongoDB: 2654.

9. **Url_filter:**

   - Quantidade de dados recebidos: 218791.

10. **Unload_filter:**

    - Quantidade de dados recebidos: 14283.

Quantidade de mensagens de erros em cada componente

1. **Coletor_twitter:**

   - Quantidade total de erros detectados: 234;
     - **MySQL Conexao** ou *Falha Conexao MySQL* (erro de conexão com o banco de dados MySQL): 60.
     - **Twitter Coleta inativa** ou *Falha ColetaTwitter* (não conseguiu estabelecer conexão com a rede social): 56.
     - **MongoDB atualizar** (não conseguiu enviar dados ao banco de dados MongoDB): 98.
     - **RabbitMQ Offline** ou *Falha RabbitMQOffline* (não conseguiu enviar dados ao middleware RabbitMQ): 10.
     - **RabbitMQ Arquivo backup** ou *Falha RabbitMQBackupFile* (não conseguiu enviar dados ao arquivo de backup): 10

2. **Reader_filter:**

   - Quantidade de erros detectados: 450.

- **Coleta limite** ou *LimiteColeta limit* (erro emitido ao encontrar a palavra *limit* no dado): 340.
- **desconectado** ou *Desconectado disconnect* (erro emitido ao encontrar a palavra *disconnect* no dado): 100.
- **EncodingErrors** (erro encontrado quando não decodifica o dado recebido de maneira correta): 10.

3. **Lang_filter:**
   - Nenhum erro detectado.
4. **Terms_filter:**
   - Nenhum erro detectado.
5. **Lac_filter:**
   - Nenhum erro detectado.
6. **Geo_filter:**
   - Quantidade de erros detectados: 284.
     - **MySQL Conexao** ou *Falha Conexao MySQL* (erro de conexão com o banco de dados MySQL): 267.
     - **MySQL Conexao** novamente (erro de conexão com o banco de dados MySQL na segunda tentativa): 12.
     - **RabbitMQ Fila Consumo** ou *Falha Fila RabbitMQ* (erro ao consumir a fila gerenciada pelo middleware RabbitMQ): 5.
7. **Map_filter:**
   - Nenhum erro detectado.
8. **Update_filter:**
   - Nenhum erro detectado.
9. **Url_filter:**
   - Quantidade de erros detectados: 218049.
     - **Timeout** ou *TimeoutUrl* (erro quando o tempo não é suficiente para expansão da url): 9.
     - **Geral impressa True** ou *ErroGeralURL* (erro quando não consegue expandir a url): 4.
     - **RabbitMQ Fila Consumo** ou *Falha Fila RabbitMQ* (erro ao consumir a fila gerenciada pelo middleware RabbitMQ): 218036.
10. **Unload_filter:**
    - Quantidade de erros detectados: 7.
      - **RabbitMQ Fila Consumo** ou *Falha Fila RabbitMQ* (erro ao consumir a fila gerenciada pelo middleware RabbitMQ): 7.

6.6.1 Análise realizada dos dados recebidos pelos componentes com a quantidade de erros encontrados

A totalidade dos dados que chegam ao RabbitMQ através do processo *Coletor_twitter* e dos que chegam ao processo *Reader* possuem a diferença de um a mais no último, tendo sido ocasionado, provavelmente, pelo fato de algum dado não ter sido entregue ao RabbitMQ pelo Coletor_twitter e sim pelo banco de dados MongoDB.

A totalidade dos dados registrados que saem do filtro *lac* e chegam ao filtro *geo* se diferenciam em vinte e três. Três desses dados já existiam no processo *Geo* aguardando para serem processados, como pode-se observar na totalidade de mensagens que saem do processo Geo. Os outros vinte dados são devidos à interações que ocorreram devido a *interrupção do serviço* do banco de dados MySQL antes do dia 30/08/2013, como mostra a Figura 6.12.

Figura 6.12: Os vinte erros encontrados no componente Geo_filter

A totalidade dos dados registrados que chegaram ao processo Url foi muito alta, porém justificada ao ser feita a análise da quantidade de erros que foram produzidos neste processo. Chegaram 218791 dados e foram produzidos 218036 erros de **RabbitMQ Fila Consumo**. Realizando a diferença, chega-se ao resultado de 755 dados, um a mais que a quantidade esperada.

E finalmente, a totalidade de dados registrados que chegaram ao processo Unload diferenciou-se em sete da esperada. Foi justificada exatamente pela quantidade de erros gerados neste processo, sete erros de **RabbitMQ Fila Consumo**.