# SISTEMA DE MONITORAMENTO EM SISTEMAS OPERACIONAIS LINUX

Marcos Paulo Coelho, Daiane Mastrangelo Tomazeti, Carlos Eduardo Pagani
[1]Superior em Tecnologia em Análise e Desenvolvimento de Sistemas – Instituto Federal de São Paulo - Campus Hortolândia (IFSP)
marcosiss7@hotmail.com, {daianetomazeti, pagani}@ifsp.edu.br

***Abstract***. *The project will develop a Linux Operating System (OS) monitoring script, developed in shell, the scripts collect data from the machine, such as, for example, CPU (Central Processing), memory and disk space usage (filesystems), which are the basic resources for an operating system to work correctly, the threshold to be stipulated is not part of the scope of this work. The web interface was developed in the Angular language, using the incremental method, where each delivery was evaluated and tested by the developer, the relational database used was MySQL.*

***Resumo***. *O projeto desenvolverá um script de monitoração do Sistema Operacional (SO) Linux, desenvolvido em shell, o scripts coleta dados da máquina, como, por exemplo, CPU (Central de Processamento), memória e utilização de espaço em disco (filesystems), que são os recursos básicos para que um sistema operacional funcione corretamente, o threhold a ser estipulado não faz parte do escopo desse trabalho. A interface web, foi desenvolvida na linguagem Angular, utilizando o método incremental, onde cada entrega foi avaliada e testada pelo desenvolvedor, o banco de dados relacional utilizado foi MySQL.*

## 1. Introdução

No mundo da computação é de suma importância que os administradores de Sistemas Operacionais sejam capazes de identificar com antecedência os possíveis problemas e falhas que podem ocorrer durante o funcionamento de um Sistema Operacional. Dentro desse mundo de computação, existem ferramentas cuja função é realizar de maneira precisa, o monitoramento de aplicações e os servidores onde essas aplicações estão implantadas. Esse tipo de ferramenta é muito importante para qualquer organização e visam que suas aplicações e Sistemas Operacionais permaneçam em pleno funcionamento, auxiliando assim os

administradores a obterem informações necessárias caso haja a necessidade de tomada rápida de decisão.

Neste cenário complexo onde estão interligados roteadores, *switches* e servidores, a tarefa de gerenciar esses dispositivos e garantir não só que os mesmos funcionem perfeitamente, mas também que estejam configurados de forma correta, pode não parecer convidativa. (MAURO, DOUGLAS 2001).

Um sistema de informação bem configurado se torna muito eficiente e facilita a troca de informação, auxiliando em uma rápida tomada de decisão, causando assim uma alta produtividade. É justamente a saúde do Sistema Operacional que ajuda a garantir um perfeito funcionamento de suas aplicações e banco de dados, consequentemente um bom funcionamento das organizações.

## 1.1 **Justificativa**

O antigo modelo de apenas um computador atendendo todas as necessidades de um ambiente já está ultrapassado, esse modelo foi substituído por um conjunto de servidores e aplicações que funcionam em conjunto com rede de dados e banco de dados.

Esse conjunto de "sistemas" é de grande importância para a organização, impactando diretamente em seu negócio e seguimento, é muito importante que os administradores de Sistemas Operacionais, consigam antecipar algumas falhas ou problemas provenientes do funcionamento do Linux, conseguindo assim corrigir em tempo hábil falhas de configurações do SO.

A motivação desse trabalho é desenvolver uma ferramenta flexível que venha colaborar e facilitar o trabalho dos administradores de Sistemas Operacionais, tentando antecipar situações críticas que podem impactar diretamente no Sistema Operacional.

## 1.2 **Objetivos Gerais**

O objetivo desse trabalho é o desenvolvimento de um *script* em *shell* exclusivamente para a monitoração dos recursos do SO Linux em conjunto com uma interface *web*.

## 1.3 **Objetivos Específicos**

Monitoria dos recursos de memória RAM, processamento (CPU), e espaço em disco (*filesystem*), em tempo real, do Sistema Operacional Linux.

O *script* permitirá a coleta de dados de um ou mais IP's (*Internet Protocol* – protocolo de rede) cadastrados.

O tempo de coleta dos dados do SO é de um ciclo de um minuto.

A saída do *script* será direcionada para um banco de dados MySQL, contendo as tabelas específicas, o banco de dados será instalado em um SO baseado em GNU/Linux, para esse trabalho foi utilizado Ubuntu.

Os dados de cada máquina são mostrados na tela gráfica e mudarão de cor de acordo com a utilização dos recursos.

O *threshold* (limite) a ser estipulado não faz parte do escopo desse trabalho, ficando essa responsabilidade para o analista.

A interface *web* utilizará uma consulta (s*elect*) para apresentação dos dados na tela.

Os dados da monitoração serão encaminhados para um banco de dados MySQL, que armazenará os dados monitorados em tabelas específicas para futuras consultas e históricos (*logs*) caso haja a necessidade de uma busca. Os dados armazenados conterão o consumo do período selecionado assim como a data da mesma.

## 2. Trabalhos Correlatos

São projetos existentes que podem possuir similaridades com a proposta realizada pelo trabalho em questão. Ao longo das pesquisas foram utilizados como base para se ter uma ideia de como o projeto poderia ser construído, são eles:

A) Tiago Silva Leal, curso de Ciências da Computação, Santa Cruz do Sul: Nesse trabalho o autor descreve possíveis problemas relacionados a tráfego de rede de dados, suas possíveis causas, como, por exemplo, a má configuração de equipamentos, ou colisões de dados, assim como natureza de entrada e saída dos dados, falha em roteadores e tráfego de vírus pela rede. O autor desenvolve métodos de monitoração em tempo real, para que seja verificado possíveis falhas na rede. Um dos recursos que o mesmo utiliza é a captura de dados em tempo real, *sniffer* (ou *packet sniffer*), técnica que envolve componentes de *hardware* e *software* capaz de capturar o tráfego de redes cabeadas ou sem fio. O autor utiliza também a captura de tráfego de monitoramento por meio de portas de espelhamento o (*port mirror*) e *SPAN* (*Switched Port Analyzer*), ou seja, configurar as portas onde o tráfego ocorrerá e seguir com a sua monitoração.

B) João Xavier da Silva Neto, gerenciamento e monitoração de redes de computadores com ênfase em disponibilidade de servidor *web* com a ferramenta Zabbix, PUC Goiás: Nesse trabalho o autor explica o gerenciamento de monitoração de redes ligado a disponibilidade de servidores *web*, por meio da ferramenta *Zabbix*, o autor explica em sua obra a funcionalidade da aplicação, assim como suas principais funções e configurações, já que a sua gerência é

100% *web*, quase em tempo real. O autor detalha que a monitoração é possível por meio de um agente instalado no servidor, esse agente captura os dados de monitoração e os envia a um servidor, que por sua vez, replica os dados processados para a interface *web*, semelhante a esse trabalho.

O autor disponibiliza em sua obra o detalhamento da funcionalidade do *Zabbix*, assim como a instalação do agente, as principais configurações, a arquitetura do *software*, a arquitetura distribuída, a arquitetura híbrida de monitoramento de *clusteres*, assim como a funcionalidade de cada um desses tópicos.

O autor apresenta a funcionalidade do Z*abbix*, sua interface *web*, banco de dados, e todas as modalidades de monitoramento e suas configurações, também orienta ao leitor, os ambientes em que a ferramenta é mais eficaz.

Esse trabalho se assemelha aos trabalhos relatados, com a semelhança da monitoração dos ambientes, porém se diferencia, por se tratar apenas de monitoria em ambiente *Linux,* por *scripts,* e não com a instalação de um *software* que faz a varredura no Sistema Operacional. O *script* é instalado em um sistema operacional Linux e por sua vez executa as coletas enviando-as para serem armazenadas em um banco de dados.

## 3. Fundamentação Teórica

Este trabalho, por tratar do desenvolvimento de *script* de monitoração e aplicação *Web* específicos para monitoração de Linux, aborda os conceitos de Desenvolvimento *Web* em conjunto com Banco de dados, Engenharia de S*oftware* e Sistemas Operacionais.

### 3.1 Engenharia de Software

Engenharia de *software* é uma abordagem sistemática e disciplinada para o desenvolvimento de *software* (PRESSMAN, 2006). A engenharia de *software* foca no *software* como produto. Com isso tem-se um incentivo e conhecimento de como focar no desenvolvimento inteiramente para o aluno, um *software* que atenda aos seus requisitos.

### 3.2 Aplicação Web

Segundo Milani e Nielson J. (2006 e 2007) são sistemas de computadores que são executados com a utilização de um navegador, podem ser acessados através de um computador por meio de um endereço URL (*Universal Resource Locator*) e não requer instalação local do cliente visto que estes sistemas operam diretamente no servidor que fornece os dados. Para o

desenvolvimento de um sistema *Web*, pode-se utilizar diversas linguagens de programação, paradigmas de programação e modelos de implementação, para este trabalho, foram utilizados o *HTML* (Linguagem de Marcação de HiperTexto), para estruturar as telas da aplicação de forma a posicionar seus componentes e elementos, o *CSS* (Cascading Style Sheets) foi utilizado para realizar a estilização das telas de forma a facilitar a interface com o usuário, tornando o *layout* mais atrativo e amigável a qualquer um que venha a acessá-lo. Como paradigma de programação, foi utilizado o paradigma de Orientação a Objetos, por meio do *MVC* (*Model-View-Controller*) e *DAO* (*Dynamic Access Object*).

### 3.3 Linguagem de Modelagem Unificada UML

Segundo Gilleanes T. A. Guedes (2011, p.17), *UML (*Linguagem de Modelagem Unificada), é uma linguagem visual utilizada para modelar *softwares* baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. Essa linguagem tornou-se, nos últimos anos, a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de *software*.

### 3.4 Modelagem Incremental

Segundo Pressman (2016, p131) o sistema foi desenvolvido de forma incremental, através da construção de funcionalidades que tinham maior importância para o funcionamento e desenvolvimento de pequenas funcionalidades subsequentes, o que permitiu que o protótipo final fosse sendo construído aos poucos, sem que houvesse uma definição detalhada de como seria.

> Quando se utiliza um modelo incremental, frequentemente o primeiro incremento é um produto essencial. Ou seja, os requisitos básicos são atendidos, porém, muitos recursos complementares (alguns conhecidos, outros não) ainda não são entregues. Esse produto essencial é utilizado pelo cliente (ou passa por uma avaliação detalhada). Como resultado do uso e/ou avaliação, é desenvolvido um planejamento para o incremento seguinte. (Pressman 2016, p.131).

Outra vantagem da utilização do modelo de desenvolvimento incremental, foi a possibilidade da construção dos *scripts* e do desenvolvimento *web* em paralelo, ou seja, foram sendo desenvolvidos e testados, ou seja, sem a necessidade de parada do ambiente de testes.

### 3.5 Especificação de Requisitos

Segundo (Machado, 2016, p.06), Especificação de requisitos é responsável para se ter uma visão mais detalhada do sistema implementado, a melhor forma de fazê-los é com a especificação de requisitos de uma forma abrangente. Um bom levantamento e uma

especificação de requisitos é algo indispensável para os desenvolvedores obterem uma ideia de como o funcionamento do sistema fluirá.

### 3.6 Arquitetura do Sistema

A arquitetura do sistema permite a visualização e o entendimento de como o fluxo do sistema é executado, obedecendo cada etapa e dependências de suas etapas.

Na Figura 1, vemos a funcionalidade do sistema, o *script* é executado no servidor cliente, onde os dados são capturados e armazenados temporariamente no diretório especificado e criado pelo *script install-client.sh,* os dados são enviados para o banco de dados através do *script dbsend.sh,* a aplicação *web* por sua vez captura os dados do banco de dados e os apresenta para o usuário final que o interpreta.

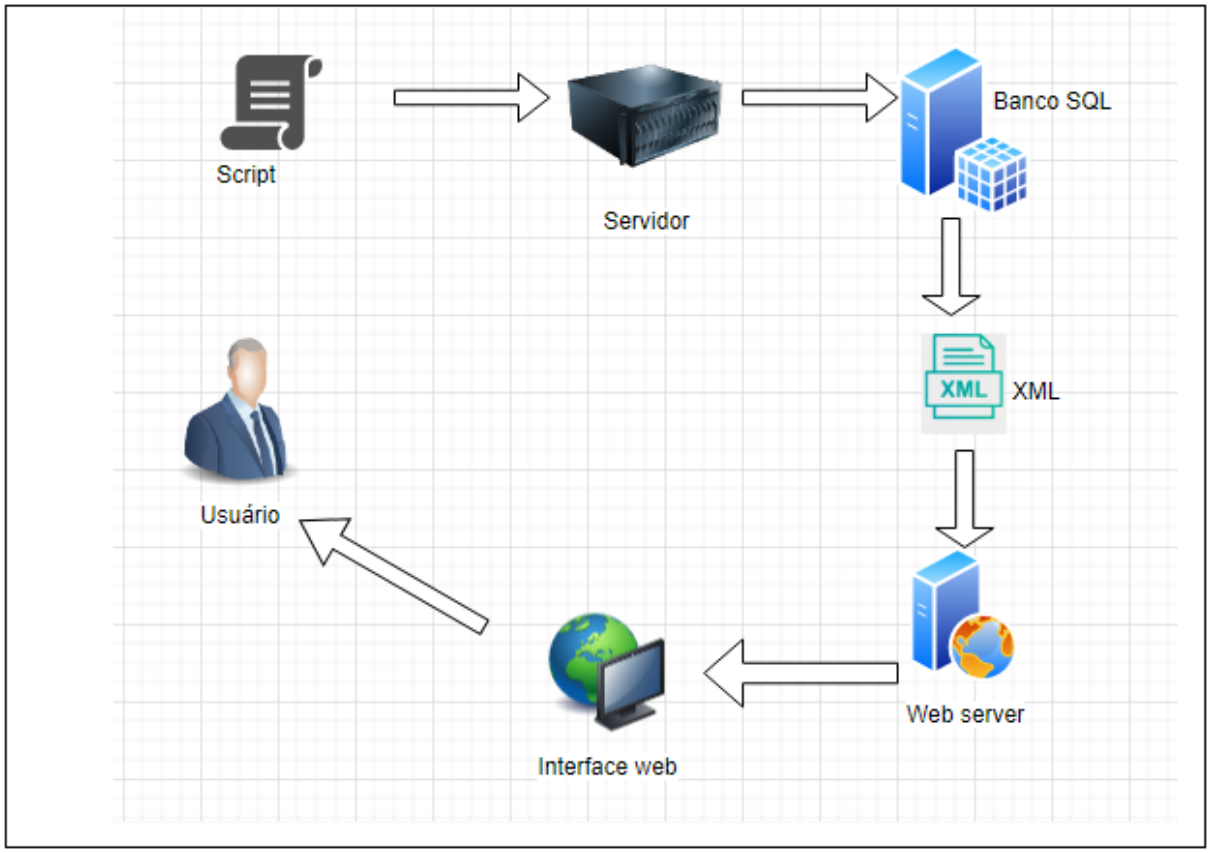



Figura 1: Arquitetura do sistema. Fonte: Elaborado pelo autor.

## 4. Desenvolvimento do Projeto

No desenvolvimento foram utilizados conhecimentos de acordo com o aprendizado em sala de aula. A seguir estão o que utilizamos no desenvolvimento desse projeto.

### 4.1 Construção dos Scripts

Os *scripts* que foram desenvolvidos nesse trabalho são:

- Desenvolvimento do *script* de instalação do banco de dados.
- Execução do *script* de instalação do banco de dados.
- Teste realizado no banco de dados.
- Desenvolvimento dos *scripts* de instalação no servidor cliente.
- Execução do *script* de instalação no servidor cliente.
- Teste realizado no servidor cliente.

### 4.2 Desenvolvimento da Aplicação

Para o desenvolvimento do *software* deste projeto foram desenvolvidas as seguintes etapas:

- Criação do portfólio da aplicação em rascunho com os requisitos funcionais e não funcionais.
- Desenvolvimento da tela de login.
- Desenvolvimento do menu lateral com as informações *Dashboard,* Máquina, Relatório e Usuário.
- Desenvolvimento da tela de *Dashboard*, com as caixas de informações apresentadas na tela.
- Desenvolvimento da tela de *Dashboard* “Máquinas em Monitoramento”
- Testes efetuados na funcionalidade “Máquinas em Monitoramento”.
- Desenvolvimento da tela de *Dashboard* “Usuários Ativos”.
- Testes efetuados na funcionalidade “Usuários Ativos”.
- Desenvolvimento da tela de *Dashboard* “Em Alerta”.
- Testes efetuados na funcionalidade “Em Alerta”.
- Desenvolvimento da tela de *Dashboard* “Estado crítico”.
- Testes efetuados na funcionalidade “Estado crítico”.
- Desenvolvimento da tela de *Dashboard* “Monitoramento”.
- Testes efetuados na funcionalidade “Monitoramento”.
- Desenvolvimento da tela de *Dashboard* “File system”.
- Testes efetuados na funcionalidade “File system”.

### 4.3 Requisitos Funcionais

Requisitos funcionais são os requisitos que especificam como o sistema deve funcionar, eles são definidos para atender a necessidade do usuário e/ou da aplicação. Na Tabela 1 estão os requisitos funcionais do projeto.

**Tabela 1. Tabela dos Requisitos Funcionais. Fonte: Elaborado pelo autor.**

| |
|---|
| RF01 *Script* de coleta – O *script* coleta.sh coleta os dados de monitoração da CPU, da memória RAM e do *Filesystem*. |
| RF02 *Script* de coleta – O analista insere o número de IP do servidor a ser monitorado no arquivo: /var/monitoria/*srvhost* |
| RF03 *Script* de coleta - A função rsync -rav envia os dados coletados para o servidor de banco de dados através do comando: rsync -rav --password-file=/etc/rsyncd.passwd monitor@$srvhost::data /var/monitoria/coleta/coleta/$srvhost |
| RF04 Aplicação *web* - Funcionalidade desempenhada pelo usuário administrador, ou seja, cadastrar novos usuários, excluir usuários e alterar usuários. |
| RF05 Aplicação *web* – Funcionalidade desempenhada pela aplicação, exibir os menus laterais e superior. |
| RF06 Aplicação *web* – Funcionalidade executada pela aplicação, consultar dados de monitoria coletados pelo *script* e enviados ao banco de dados através do comando *SELECT* no banco de dados. |
| RF07 Aplicação *web* – Funcionalidade executada pela aplicação, alterar a cor das informações caso o *threshold* estipulado ultrapasse o valor. |

### 4.4 Requisitos Não Funcionais

Requisitos não funcionais são aqueles que não estão diretamente relacionados a funcionalidades de um sistema. Na Tabela 2 estão os requisitos não funcionais do projeto.

**Tabela 2. Tabela dos Requisitos Não Funcionais. Fonte: Elaborado pelo autor.**

| |
|---|
| RNF01 – Os arquivos install-server e install-client devem ser armazenados dentro do mesmo *filesystem,* pois a execução dos *scripts isntall-server.sh* e *install-client.sh* dependem de um conjunto de arquivos necessários para a perfeita configuração. |
| RNF02 – O *script install-client.sh* – instala o agente de monitoria e cria os *filesystens,* necessários para a coleta dos dados monitorados em seguida efetua a |

| |
|---|
| configuração na *CRONTAB*. |
| RNF03 – O *script install-client.sh* – cria os filesystens necessários para armazenamento temporário dos dados coletados, os *filesystens* já são pré-configurados internamente no *script.* |
| RNF04 - O *script install-server.sh* – instala e configura o banco de dados no servidor de banco, criando as tabelas necessárias para o armazenamento dos dados coletados. |
| RNF05 – A aplicação deve ser uma aplicação *web*. |
| RNF06 – A aplicação *web* utiliza a função *SELECT* para capturar os dados coletados e apresentá-los na tela do usuário. |
| RNF07 – O *threshold* a ser estipulado é de responsabilidade do analista. |

### 4.5 Shell

Segundo Gleydson Mazioli da Silva (2022, p.26), *shell* é o programa responsável em interpretar as instruções enviadas pelo usuário e seus programas ao sistema operacional (o *kernel*). Ele que executa os comandos lidos do dispositivo de entrada padrão (teclado) ou de um arquivo executável. É a principal ligação entre o usuário, os programas e o kernel. O GNU/Linux possui diversos tipos de interpretadores de comandos, entre eles posso destacar o *bash, ash, csh, tcsh, sh,* etc. Entre eles o mais usado é o *bash*.

### 4.6 Angular

Segundo Daniel Schmitz e Douglas Lira (2015, p.ii), A*ngular* é um *framework* mantido pelo *Google* e possui algumas particularidades interessantes, que o fazem um *framework* muito poderoso.

Angular é uma plataforma e *framework* de código aberto para construção da interface de aplicações usando HTML, CSS e, principalmente, JavaScript, criada pelos desenvolvedores da Google.

### 4.7 MySQL

Segundo Milani (2006, p.24), além de ser extremamente rápido, pelo fato de armazenar os dados em tabelas do modo ISAM (código de baixo nível), o MySQL é altamente confiável.

Para armazenamento de dados, será utilizado o sistema de banco de dados relacional MySQL, onde os dados serão gravados na base de dados que o próprio sistema fornecerá como entrada.

O MySQL ajudará também no retorno de dados, ou seja, na consulta feita pelo banco e exibida na tela do usuário, tendo a vantagem de escalabilidade e confiabilidade, ou seja, um número ilimitado de utilização por usuários simultâneos; capacidade de manipulação de tabelas com mais de 50.000.000 de registros; alta velocidade de execução de comandos.

Por possuir essas características, a ferramenta é indicada para uso em aplicações em todas as áreas de negócio independentemente do tamanho de sua aplicação.

A Figura 2 representa a Modelagem Conceitual do banco de dados do sistema de monitoramento em Sistema Operacional Linux.

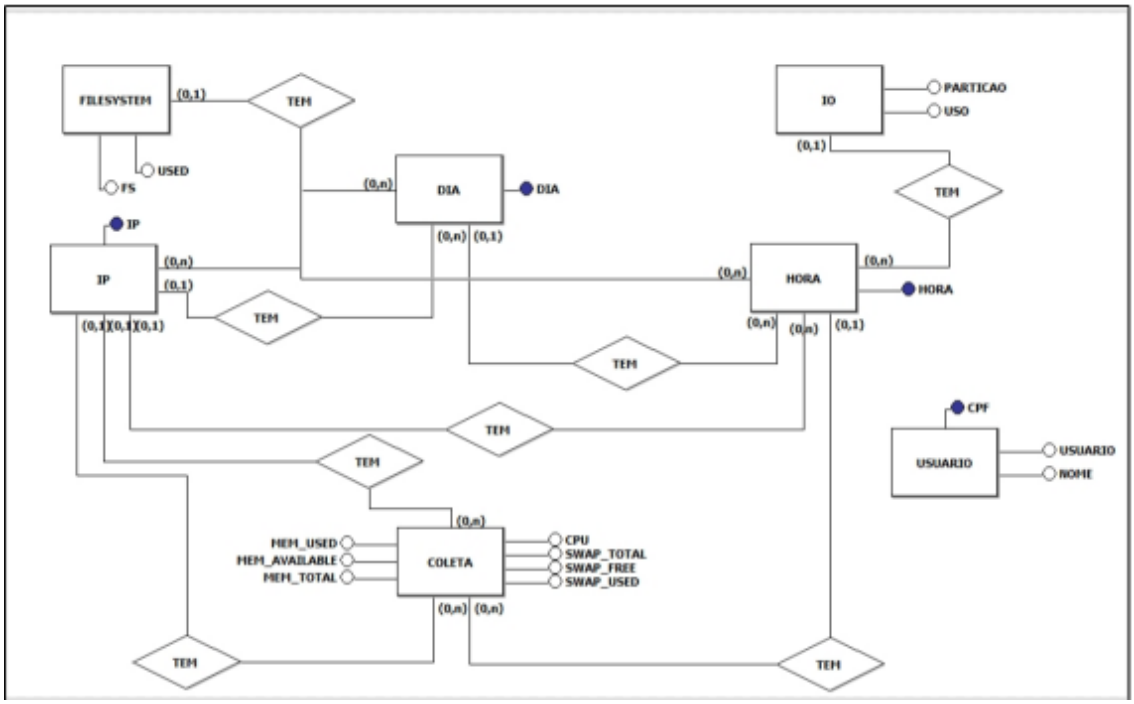



Figura 2: Modelo Conceitual do Banco de Dados. Fonte: Elaborado pelo autor.

### 4.8 *JavaScript*

Para um sistema mais dinâmico e flexível contaremos com a aplicação J*avaScript*, que é uma linguagem de programação criada em 1995 por Brendan Eich, como propósito de oferecer os processos de páginas *web* mais dinâmicos, tornando seu uso mais agradável. Em tese é necessário apenas de um navegador para realizar *scripts*. Como explicado por Preston (2016, p.02), o *JavaScript* por ser, uma linguagem de *script*, não funciona por si só, na verdade ele é executado no navegador *web*, que é responsável por executar o *JavaScript* instalado e ativado por padrão.

### 4.9 Execução do *script* no servidor de banco de dados e servidor *cliente*

Os *scripts* a serem executados são *install-server.sh* para o servidor de banco de dados e *install-client.sh* para os servidores cliente.

Nessa etapa não há a necessidade de efetuar configurações manuais, os *scripts* efetuam todas as configurações necessárias, a única exigência é que os arquivos *install-server.rar* e *instal-client.rar* sejam descompactados no mesmo diretório, recomenda-se que seja descompactado no /tmp.

#### 4.9.1 *Scripts* de instalação e configuração a serem executados dentro do servidor de banco de dados.

A Figura 3 mostra os arquivos que devem ser extraídos dentro de uma pasta no Linux, o **instal-serverl.sh** utiliza dos arquivos para a correta instalação automática do banco de dados, cabendo ao usuário executar somente o *script* utilizando o comando **\instal-serverl.sh**.

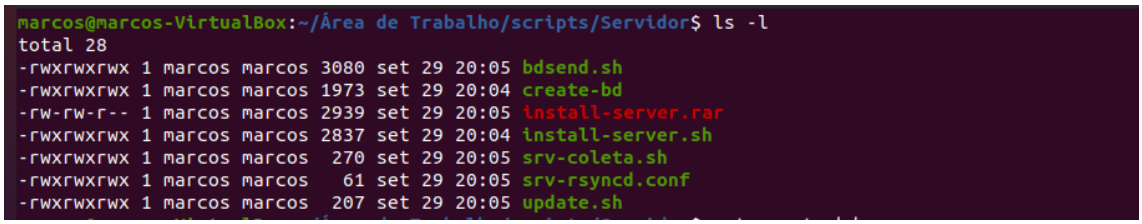



Figura 3: Local onde os *scripts* devem ficar armazenados. Fonte: Elaborado pelo autor.

As Figuras de 4 e 5 mostram as principais funcionalidades do *script* de envio da coleta para o banco de dados.

A Figura 4 atribui os valores das variáveis setadas na construção do *script*.

```
5   bdsend="mysql --defaults-extra-file=/home/monitor/monitor@localhost.cnf -u monitor monitoria -e"
6   dia=`date -I`
7   hora=`date | awk '{print $5}'`
```

Figura 4: Atribuição de variáveis. Fonte: Elaborado pelo autor.

A Figura 5 captura os IP's dos servidores sendo monitorados através do for.

Os comandos **cat /var/monitoria/coleta/$x/df | awk '{print $1}' > /var/monitoria/coleta/$x/fs** e **cat /var/monitoria/coleta/$x/df | awk '{print $2}' > /var/monitoria/coleta/$x/fs-uso** separam os dados do DF para poder preencher o banco de dados.

As linhas de comando **cat /var/monitoria/coleta/$x/io | awk '{print $1}' > /var/monitoria/coleta/$x/io-part** e **cat /var/monitoria/coleta/$x/io | awk '{print $2}' > /var/monitoria/coleta/$x/io-uso** separam os dados do IO para poder preencher o banco de dados.

O comando **$bdsend "INSERT INTO ip VALUES ('$x');"** atualiza as tabelas "ip, dia e hora", para que as tabelas relacionadas “coleta e filesystems” possam ser atualizadas.

Foi utilizado um *while* dentro da estrutura do "for" para que o contator se atualizasse, isso permitiu a atualização da tabela de *filesystem*.

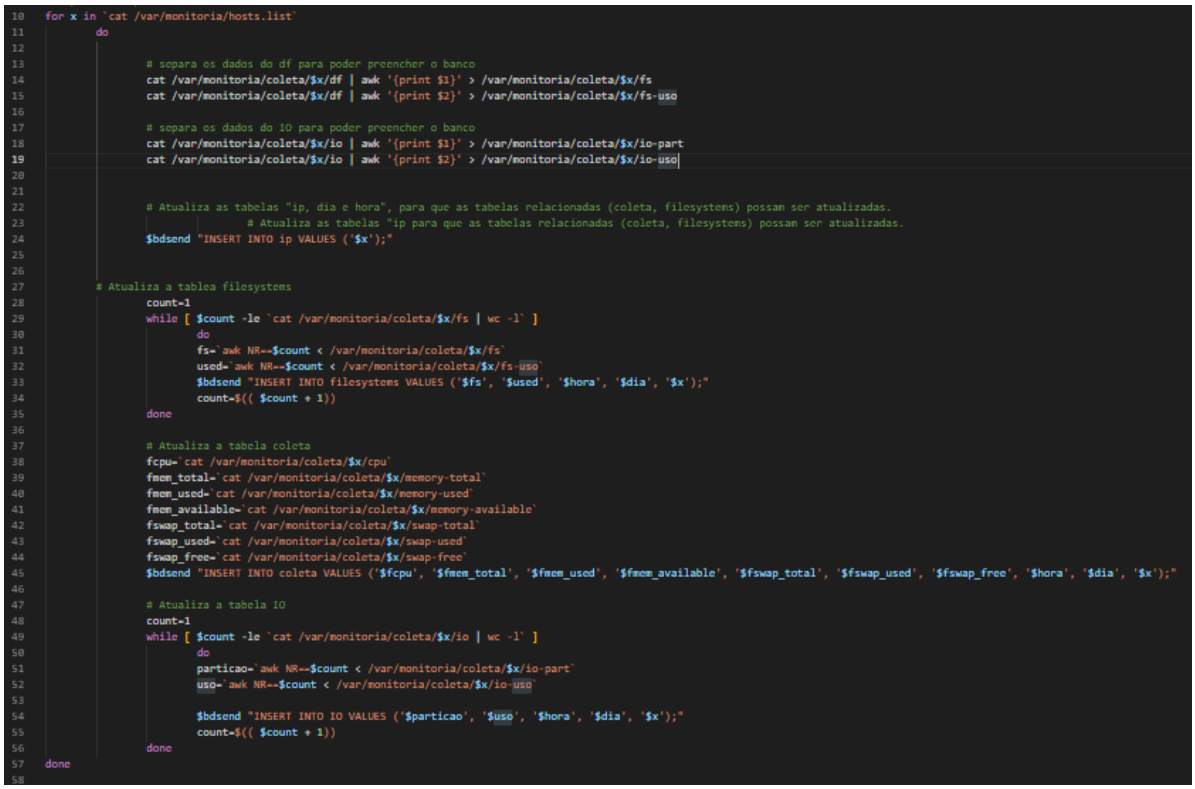
```
count=1
        while [ $count -le `cat /var/monitoria/coleta/$x/fs | wc -l` ]
            do
            fs=`awk NR==$count < /var/monitoria/coleta/$x/fs`
            used=`awk NR==$count < /var/monitoria/coleta/$x/fs-uso`
                $bdsend "INSERT INTO filesystems VALUES ('$fs', '$used', '$hora', '$dia', '$x');"
            count=$(( $count + 1))
        done
```

dentro da mesma estrutura de "for" foi feito os comandos para a atualização da tabela coleta.

```
fcpu=`cat /var/monitoria/coleta/$x/cpu`
        fmem_total=`cat /var/monitoria/coleta/$x/memory-total`
        fmem_used=`cat /var/monitoria/coleta/$x/memory-used`
        fmem_available=`cat /var/monitoria/coleta/$x/memory-available`
        fswap_total=`cat /var/monitoria/coleta/$x/swap-total`
        fswap_used=`cat /var/monitoria/coleta/$x/swap-used`
        fswap_free=`cat /var/monitoria/coleta/$x/swap-free`
            $bdsend "INSERT INTO coleta VALUES ('$fcpu', '$fmem_total', '$fmem_used', '$fmem_available', '$fswap_total', '$fswap_used', '$fswap_free', '$hora', '$dia', '$x');"
```

Foram executados os comandos dentro da mesma estrutura de "for" para atualização da tabela de IO.

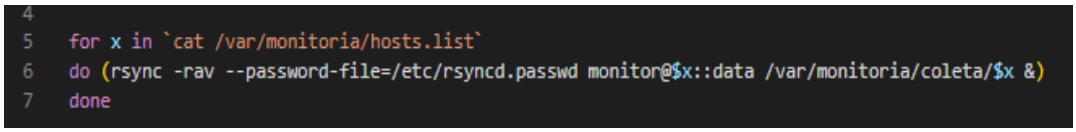
```
        count=1
                while [ $count -le `cat /var/monitoria/coleta/$x/io | wc -l` ]
                    do
                    particao=`awk NR==$count < /var/monitoria/coleta/$x/io-part`
                    uso=`awk NR==$count < /var/monitoria/coleta/$x/io-uso`

                      $bdsend "INSERT INTO IO VALUES ('$particao', '$uso', '$hora', '$dia',
'$x');"
                    count=$(( $count + 1))
                done
        done

        echo '=================================================='
        echo ''
        echo 'SCRIPT FINALIZADO'
        echo ''
        echo '=================================================='
        echo '=================================================='
```



```
for x in `cat /var/monitoria/hosts.list`
        do

                # separa os dados do df para poder preencher o banco
                cat /var/monitoria/coleta/$x/df | awk '{print $1}' > /var/monitoria/coleta/$x/fs
                cat /var/monitoria/coleta/$x/df | awk '{print $2}' > /var/monitoria/coleta/$x/fs-uso

                # separa os dados do IO para poder preencher o banco
                cat /var/monitoria/coleta/$x/io | awk '{print $1}' > /var/monitoria/coleta/$x/io-part
                cat /var/monitoria/coleta/$x/io | awk '{print $2}' > /var/monitoria/coleta/$x/io-uso


                # Atualiza as tabelas "ip, dia e hora", para que as tabelas relacionadas (coleta, filesystems) possam ser atualizadas.
                        # Atualiza as tabelas "ip para que as tabelas relacionadas (coleta, filesystems) possam ser atualizadas.
                $bdsend "INSERT INTO ip VALUES ('$x');"


        # Atualiza a tablea filesystems
                count=1
                while [ $count -le `cat /var/monitoria/coleta/$x/fs | wc -l` ]
                        do
                        fs=`awk NR==$count < /var/monitoria/coleta/$x/fs`
                        used=`awk NR==$count < /var/monitoria/coleta/$x/fs-uso`
                        $bdsend "INSERT INTO filesystems VALUES ('$fs', '$used', '$hora', '$dia', '$x');"
                        count=$(( $count + 1))
                done

                # Atualiza a tabela coleta
                fcpu=`cat /var/monitoria/coleta/$x/cpu`
                fmem_total=`cat /var/monitoria/coleta/$x/memory-total`
                fmem_used=`cat /var/monitoria/coleta/$x/memory-used`
                fmem_available=`cat /var/monitoria/coleta/$x/memory-available`
                fswap_total=`cat /var/monitoria/coleta/$x/swap-total`
                fswap_used=`cat /var/monitoria/coleta/$x/swap-used`
                fswap_free=`cat /var/monitoria/coleta/$x/swap-free`
                $bdsend "INSERT INTO coleta VALUES ('$fcpu', '$fmem_total', '$fmem_used', '$fmem_available', '$fswap_total', '$fswap_used', '$fswap_free', '$hora', '$dia', '$x');"

                # Atualiza a tabela IO
                count=1
                while [ $count -le `cat /var/monitoria/coleta/$x/io | wc -l` ]
                        do
                        particao=`awk NR==$count < /var/monitoria/coleta/$x/io-part`
                        uso=`awk NR==$count < /var/monitoria/coleta/$x/io-uso`

                        $bdsend "INSERT INTO IO VALUES ('$particao', '$uso', '$hora', '$dia', '$x');"
                        count=$(( $count + 1))
                done
done
```

Figura 5: Captura dos ip's dos servidores. Fonte: Elaborado pelo autor.

A Figura 6 é referente ao *script* **srv-coleta.sh** para coletar os arquivos dos servidores monitorados, a linha de comando do ***rsync*** capturará os dados monitorados, a linha de comando passa os parâmetros de usuário e senha enviando os dados para o *filesystem*, **/var/monitoria/coleta**, as variáveis $x são referentes ao IP do servidor monitorado.

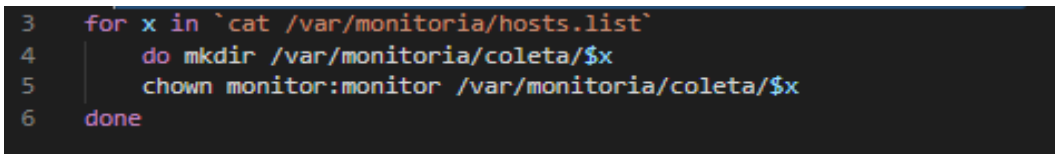


```
4
5   for x in `cat /var/monitoria/hosts.list`
6   do (rsync -rav --password-file=/etc/rsyncd.passwd monitor@$x::data /var/monitoria/coleta/$x &)
7   done
```

Figura 6: Coleta os dados dos servidores monitorados. Fonte: Elaborado pelo autor.

A Figura 7 refere-se ao *script* ***update.sh*** responsável pela criação dos diretórios dos servidores a serem monitorados, através da linha de comando:

**mkdir /var/monitoria/coleta/$x**, onde a variável $x é o ip do servidor a ser monitorado, a linha de comando **chown monitor:monitor /var/monitoria/coleta/$x** altera o *owner* do diretório para monitor.



```
3   for x in `cat /var/monitoria/hosts.list`
4       do mkdir /var/monitoria/coleta/$x
5       chown monitor:monitor /var/monitoria/coleta/$x
6   done
```

Figura 7: Configuração do diretório /var/monitoria/coleta/$x. Fonte: Elaborado pelo autor.

### 4.9.2 Execução do script no servidor cliente

A execução dos *script*s de instalação e configuração para o servidor cliente.

A Figura 8 efetua a instalação dos **sysstat** e **rsync** do Linux através das linhas de comando abaixo.

```
# instalar o iostat, rsync
apt-get install sysstat -y
apt-get install rsync -y
```

Figura 8: Instalação do sysstat e rsync. Fonte: Elaborado pelo autor.

A Figura 9 mostra a instalação e a configuração da tarefa na ***crontab***.

```
if [ `crontab -l |grep coleta.sh` -eq '']; then
crontab -l | { cat; echo "# Serviço de monitoração, gera os arquivos - Cliente"; echo "* * * * * bash /usr/sbin/monitoria/coleta.sh"; } | crontab -
else
echo '=================================================='
echo ''
echo 'CRONTAB JA ESTA CONFIGURADA'
echo ''
echo '=================================================='
fi
```

Figura 9: configuração da *crontab*. Fonte: Elaborado pelo autor.

A Figura 10 configura o **rsync (/etc/rsyncd.conf e /etc/rsyncd.passwd)** através dos comandos abaixo.

```
cp ./rsyncd.conf /etc/rsyncd.conf
echo monitor:senha > /etc/rsyncd.passwd
chmod 0640 /etc/rsyncd.conf
chmod 0600 /etc/rsyncd.passwd
```

Figura 10: Configuração do rsync. Fonte: Elaborado pelo autor.

A Figura 11 executa a inicialização do **rsync**.

```
27   systemctl enable rsync; systemctl restart rsync
28
```

Figura11: Executar a inicialização do rsync: Fonte: Elaborado pelo autor.

A Figura 12, cria o usuário da monitoração.

```
30   useradd -c "Usuario de monitoração" monitor
31
```

Figura 12: Criar usuário da monitoração. Fonte: Elaborado pelo autor.

A Figura 13 executa o comando para criar o diretório do *script* **coleta.sh**

```
33   mkdir -p -p /usr/sbin/monitoria
34
```

Figura 13: Cria diretório /usr/sbin/monitoria. Fonte: Elaborado pelo autor.

A Figura 14 executa o comando para mover o *script* **coleta.sh** para o diretório **usr/sbin/monitoria** e altera o *owner*.

```
36  cp ./coleta.sh /usr/sbin/monitoria/coleta.sh
37  chmod +x /usr/sbin/monitoria/coleta.sh
38
```

Figura 14: Move o *script* coleta.sh e altera o *owner*. Fonte: Elaborado pelo autor.

A Figura 15, executa o comando para criar diretório de coleta do *script* no **/var/monitoria/coleta**, alerta o *owner* do diretório.

```
40  mkdir -p /var/monitoria/coleta/
41  chown -R monitor:monitor /var/monitoria/coleta
```

Figura 15: cria diretório e altera o owner. Fonte: Elaborado pelo autor.

A Figura 16, executa o comando para criar o arquivo ***srvhost*** onde será configurado o ip do servidor a ser monitorado, altera também o ***owner*** do arquivo para monitoria.

```
44  touch /var/monitoria/srvhost
45  chown monitor:monitor /var/monitoria/srvhost
46
```

Figura 16: Cria arquivo de configuração *srvhost* e altera o *owner*. Fonte: Elaborado pelo autor.

A Figura 17 mostra o conjunto de comandos executados para configurar o ip do servidor de monitoração no cliente, a execução dessa etapa requer a atuação do usuário, pois há a necessidade de informar o IP do servidor de monitoração.

```
48  echo '=================================================='
49  echo ''
50  echo 'Digite o IP do servidor de monitoracao'
51  read srvip
52  echo $srvip > /var/monitoria/srvhost
53  echo ''
54  echo 'IP do servidor de monitoracao configurado!'
55  echo ''
56  echo '=================================================='
```

Figura 17: Configuração do IP do *cliente* no arquivo *srvhost*. Fonte: Elaborado pelo autor.

A Figura 18 exibe parte do comando executado ***./install-server.sh***, onde o *script* executado solicita no final da execução a informação do IP do servidor de monitoração, ou seja, o servidor de banco de dados, essa é a única interação do usuário no momento da instalação.

```
A descompactar sysstat (12.2.0-2ubuntu0.2) sobre (12.2.0-2ubuntu0.1) ...
Configurando sysstat (12.2.0-2ubuntu0.2) ...
A processar 'triggers' para man-db (2.9.1-1) ...
A processar 'triggers' para systemd (245.4-4ubuntu3.15) ...
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Pacotes sugeridos:
  openssh-server
Os pacotes a seguir serão atualizados:
  rsync
1 pacotes atualizados, 0 pacotes novos instalados, 0 a serem removidos e 364 não atualizados.
É preciso baixar 0 B/318 kB de arquivos.
Depois desta operação, 7.168 B de espaço em disco serão liberados.
(Lendo banco de dados ... 189162 ficheiros e directórios actualmente instalados.)
A preparar para desempacotar .../rsync_3.1.3-8ubuntu0.4_amd64.deb ...
A descompactar rsync (3.1.3-8ubuntu0.4) sobre (3.1.3-8) ...
Configurando rsync (3.1.3-8ubuntu0.4) ...
A processar 'triggers' para man-db (2.9.1-1) ...
A processar 'triggers' para systemd (245.4-4ubuntu3.15) ...
./install-client.sh: linha 10: [: número excessivo de argumentos
======================================================

CRONTAB JA ESTA CONFIGURADA

======================================================
Synchronizing state of rsync.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable rsync
useradd: usuário 'monitor' já existe
======================================================

Digite o IP do servidor de monitoraçao
```

Figura 18: Configuração do IP do servidor de banco de dados. Fonte: Elaborado pelo autor.

### 4.10 Desenvolvimento da Aplicação Web

A aplicação *web* desenvolvida em Angular é a plataforma de interação com o usuário, ela é responsável por apresentar ao usuário de forma clara e simplificada o resultado das monitorações dos servidores "clientes".

A Figura 19 representa a tela de login do usuário já criado, caso o usuário não esteja criado, é necessária sua criação.

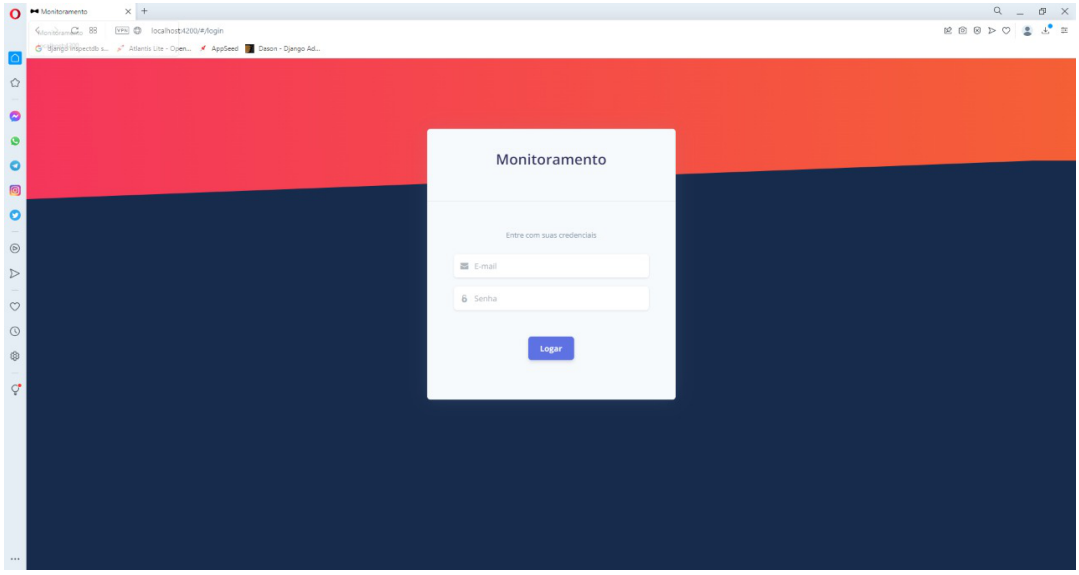



Figura 19: Tela de login do usuário. Fonte: Elaborado pelo autor.

A Figura 20, tela de *dashboard,* representa o sistema contendo o menu lateral esquerdo, o próprio menu de *dashboard,* menu de máquinas, onde são cadastradas e gerenciadas novas máquinas, relatórios, onde podem ser gerados relatórios e menu de usuários, onde podem ser gerenciados os usuários.

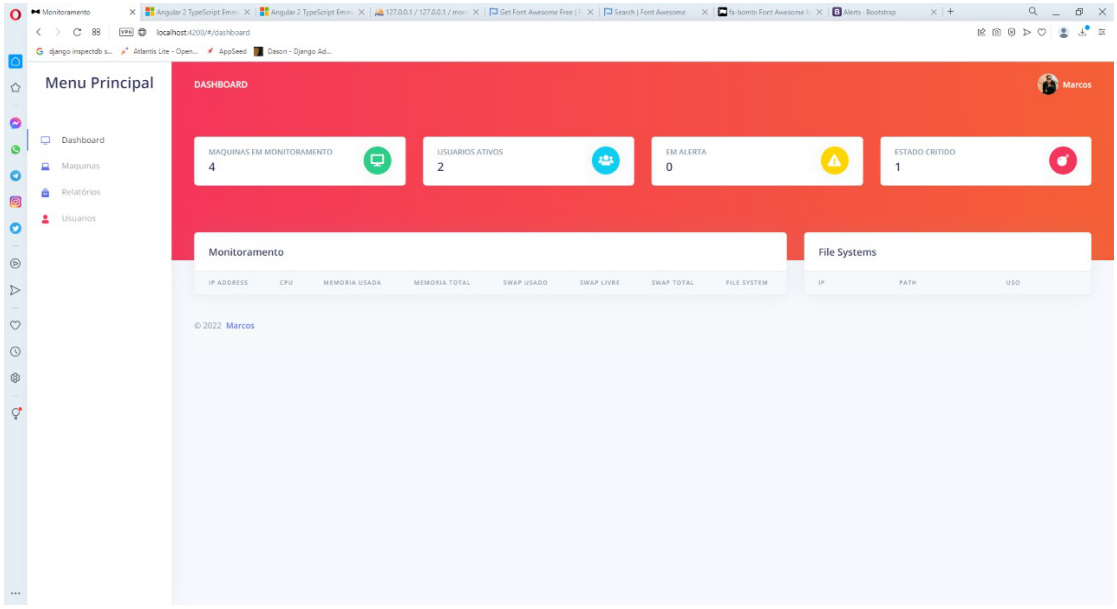



Figura 20: Tela do *dashboard* de monitoração. Fonte: Elaborado pelo autor.

A Figura 21 mostra a tela de monitoramento onde podemos visualizar os *IP's* cadastrados no sistema, visualizando assim as colunas *CPU*, Memória Total, *Swap* usado, *Swap* Livre, *Swap* total e *File System*, já com a monitoração ativa, no menu superior temos as visualizações "MÁQUINAS EM MONITORAMENTO", "USUÁRIOS ATIVOS", "EM ALERTA", "ESTADO CRÍTICO".

Sempre que a monitoração identificar uma alteração no consumo que atinja de 85% a 95% de utilização dos recursos, aparecerá no campo "EM ALERTA" e o número de servidores que apresentam o problema.

Sempre que a monitoração identificar uma alteração no consumo de recursos acima de 95% de utilização dos recursos, aparecerá no campo "ESTADO CRÍTICO" e o número de servidores que apresentam o problema.

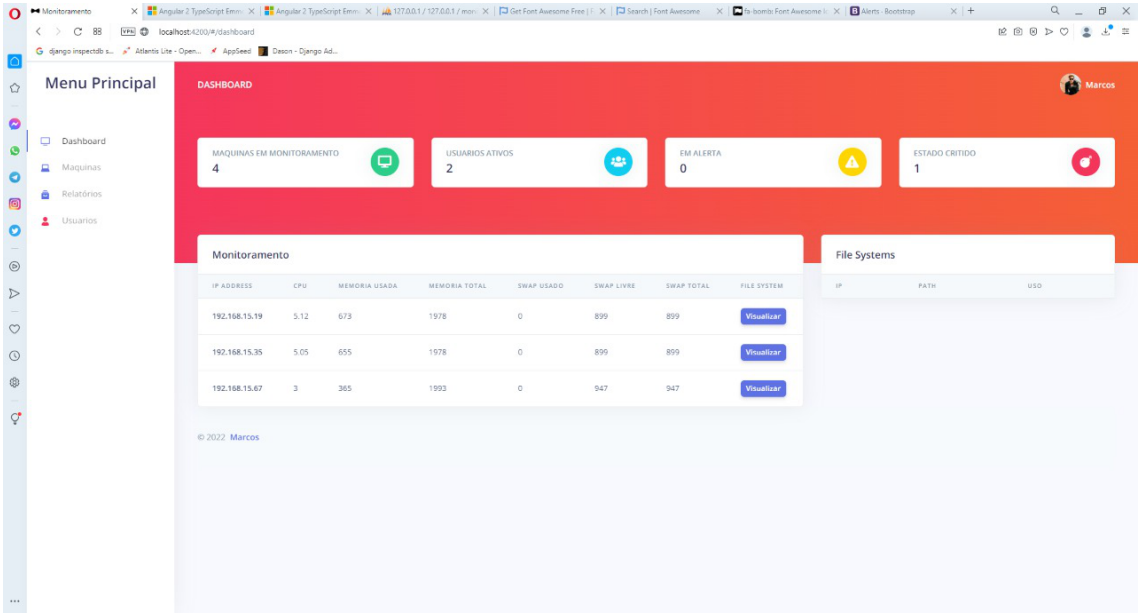



Figura 21: Tela de monitoração com os *IP's* cadastrados já sendo monitorados. Fonte: Elaborado pelo autor.

A Figura 22 mostra o *dashboard,* a monitoração ativa alarmando 100% de *filesystem,* o *"ESTADO CRITICO"* mostra o monitoramento dos *IP's*, contendo um alerta de *filesystem* /snap/core20/1405 com 100% de utilização para o *IP* 192.168.15.19, a Figura 22 mostra a visualização do campo: "ESTADO CRITICO" apresentando um servidor com alerta.

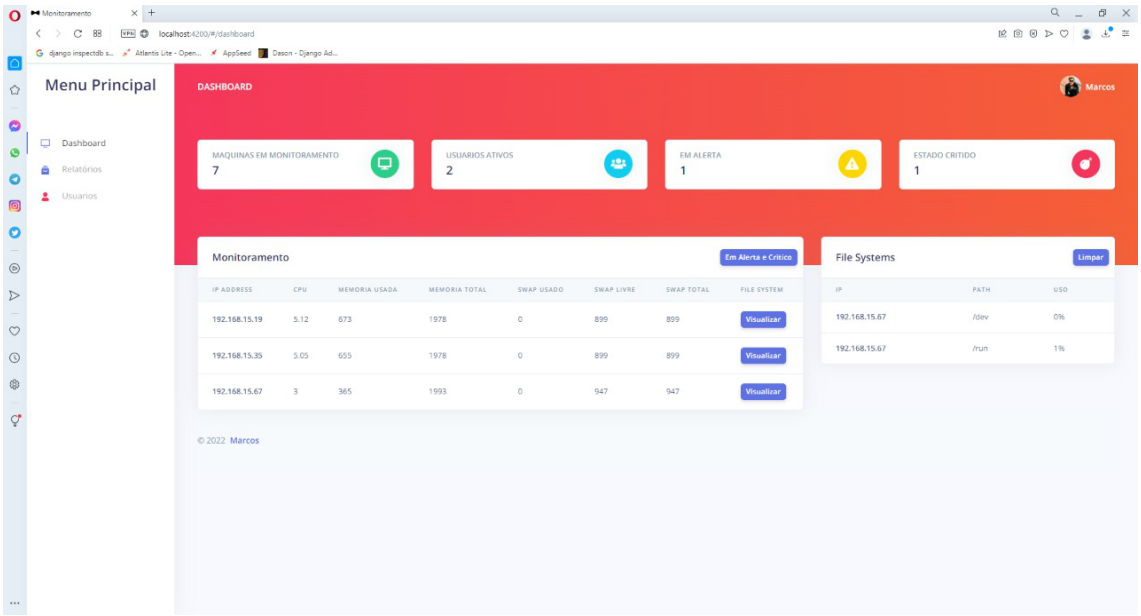



Figura 22: Tela de monitoração apresentando um servidor com alerta. Fonte: Elaborado pelo autor.

A Figura 23 mostra a tela *dashboard* com monitoração ativa alarmando 86% /dev “EM ALERTA”, mostra também o monitoramento dos *IP's*, contendo um alerta de *filesystem* /dev com 86% de utilização para o *IP* 192.168.15.35, a Figura 23 mostra a visualização do campo “EM ALERTA” apresentando um servidor com alerta, diferentemente da figura 22, o alerta não está destacado em cor diferente.

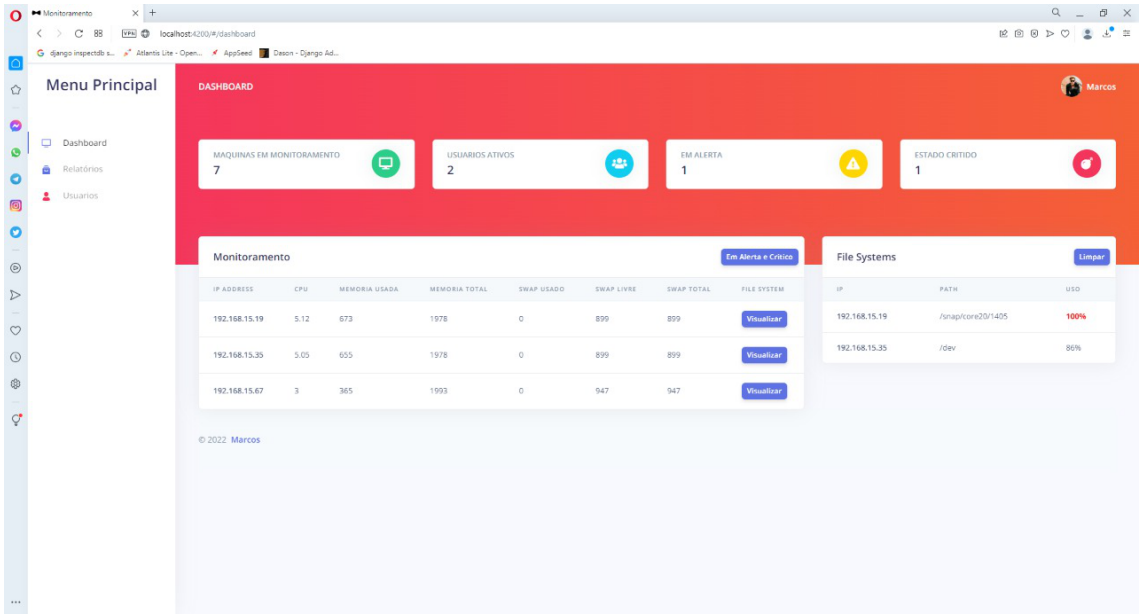



Figura 23: Tela de alerta de *File System* critico em 100% de utilização. Fonte: Elaborado pelo autor.

## 5. Considerações Finais

Foi criado um *script* somado a uma interface gráfica, para monitoração do sistema operacional Linux, com intuito de monitorar a máquina e prevenir que a mesma sofra com consumo de recursos, foram utilizadas as disciplinas e tecnologias aprendidas no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas e serviços de redes, para auxiliar no desenvolvimento do sistema.

A maior dificuldade encontrada durante a construção do sistema foi integração da interface *web* com os dados coletados pelo banco de dados em tempo real, cada vez que um IP foi adicionado ao servidor para efetuar a leitura e transmitir para o banco de dados, havia falhas na transmissão, a sincronia não era eficaz, o problema foi corrigido com a mudança nas linhas de comando do *rsyncd*.

Para o desenvolvimento deste trabalho e até mesmo destacando as competências e habilidades adquiridas no decorrer do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, houve aplicação de conceitos associados às disciplinas de

Desenvolvimento *Web*, Engenharia de *Software*, Qualidade de *Software*, Sistemas Operacionais, Redes de Computadores, Banco de Dados, Metodologia de Pesquisa e Projeto de Sistemas para planejamento e desenvolvimento teórico-técnico. Outros conhecimentos necessários para o desenvolvimento do trabalho foram agregados através de pesquisa bibliográfica e estudo técnico por meio da ferramenta apresentada.

Para trabalhos futuros, podem ser desenvolvidas novas funções, como, por exemplo, a criação da monitoria de tráfego de rede, inclusão do Sistema Operacional Windows e outras funções.

## 6. Referências Bibliográficas


ACHMITZ Daniel e DOUGLAS Lira, **AngularJS na prática, Crie aplicações web com AngularJS**, versão foi publicada em 2015-07-06.

Desenvolvimento iterativo e incremental? StartUp Sorocaba, 2019. Disponível em: http://startupsorocaba.com/Desenvolvimento-iterativo-e-incremental/. Acesso em: 28, outubro 2022.

FLANAGAN, David. **Javascript**. Estados Unidos: O-Reilly Media, 2011.

GUEDES Gilleanes T. A., **UML 2 Uma abordagem pratica**, obra foi revisada, atualizada e ampliada tomando como base o livro: "UML – Uma Abordagem Prática", ISBN 978-7522-149-5. Junho/2011 Segunda edição Maio/2009 Primeira edição (ISBN: 978-85-7522-193-8).

GUEDES Raphael Melo & Elaine de Mattos Silva, **Introdução ao Uso do Linux**. Rio de Janeiro 4 a Versão, junho de 2006 UERJ. Disponível em: http://www.lee.uerj.br/~elaine/introducao-ao-uso-do-linux.pdf. Acesso em: 23, setembro 2022

JALOTE, P. **An Integrated Approach to Software Engineering**. 3. ed. New York: Springer, 2005, 566p.

LEAL Tiago Silva. **curso de Ciências da Computação**, **Santa Cruz do Sul**. Disponível em: https://repositorio.unisc.br/jspui/bitstream/11624/2156/1/Tiago%20Silva%20Leal.pdf/ Acessado em: 23, setembro 2021.

NETO João Xavier da Silva, **Gerenciamento e Monitoração de redes de Computadores com Enfase em Disponibilidade de Servidor Web com Ferramenta Zabbix, PUC Goiás**. Disponível em:

https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/1795/1/TCC%20II%20-%20Jo%C3%A3o%20Xavier%20-%20Gerenciamento%20e%20Monitora%C3%A7%C3%A3o%20de%20Rede%20com%20%C3%AAnfase%20em%20Disponibilidade%20de%20Servidor%20WEB%20com%20a%20Ferramenta%20Zabbix%20Revisa.pdf/. Acesso em: 23, setembro 2022.

SILVA Gleydson Mazioli, **Guia Foca Linux: Versão 5.02** - segunda, 27 de julho de 2020 data de publicação segunda, 27 de julho de 2020.

SILVA M. **Educação online: teorias, práticas, legislação, formação corporativa**. 2. Ed. São Paulo: Editora Loyola, 2003.

SILVA, M. Samy. **Criando sistemas com HTML: Sites de Alta Qualidade com HTML e CSS**. São Paulo: Editora Novatec, 2008.

SOMMERVILE, Ian. **Engenharia de Software**. 8. Ed. São Paulo: Pearson Prentice Hall, 2007.

SOMMERVILE, Ian. **Engenharia de Software**. 9. Ed. São Paulo: Pearson Prentice Hall, 2011.

# Apêndice I

Os *scripts* podem ser obtidos em:

https://github.com/marcoscoelho123/tcc_final_scriptMonitoraLinux

***Lista 1 Script install-client.sh.***

```
#!/bin/bash
# Script de instalaçao do serviço de monitoria - cliente.
# Pre-requisitos - cliente

# instalar o iostat, rsync
apt-get install sysstat -y
apt-get install rsync -y

# Instalar tarefa na crontab para gerar os arquivos
if [ `crontab -l |grep coleta.sh` -eq '']; then
crontab -l | { cat; echo "# Serviço de monitoraçao, gera os arquivos - Cliente"; echo "* * * * * bash
/usr/sbin/monitoria/coleta.sh"; } | crontab -
else
echo '======================================================'
echo ''
echo 'CRONTAB JA ESTA CONFIGURADA'
echo ''
echo '======================================================'
fi

#Configurar o rsync (/etc/rsyncd.conf e /etc/rsyncd.passwd)
cp ./rsyncd.conf /etc/rsyncd.conf
echo monitor:senha > /etc/rsyncd.passwd
chmod 0640 /etc/rsyncd.conf
chmod 0600 /etc/rsyncd.passwd

# habilitar rsync e iniciar
systemctl enable rsync; systemctl restart rsync

# criar usuario da monitoraçao
useradd -c "Usuario de monitoraçao" monitor

# criar diretorio do script no /usr/sbin/monitoria
mkdir -p -p /usr/sbin/monitoria

# Enviar o script para o diretorio do script e atribuir a permissão de execuçao
cp ./coleta.sh /usr/sbin/monitoria/coleta.sh
chmod +x /usr/sbin/monitoria/coleta.sh

# Criar diretorio de coleta do script no /var/monitoria/coleta
mkdir -p /var/monitoria/coleta/
chown -R monitor:monitor /var/monitoria/coleta

# Cria o arquivo para configurar qual o ip do servidor de monitoraçao
touch /var/monitoria/srvhost
chown monitor:monitor /var/monitoria/srvhost
# Configura o ip do servidor de monitoraçao no cliente
echo '======================================================'
```

```
echo ''
echo 'Digite o IP do servidor de monitoraçao'
read srvip
echo $srvip > /var/monitoria/srvhost
echo ''
echo 'IP do servidor de monitoraçao configurado!'
echo ''
echo '======================================================'

echo '======================================================'
echo ''
echo 'SCRIPT FINALIZADO'
echo ''
echo '======================================================'
```

**Lista 2** ***Script coleta.***

```
#!/bin/bash
# Caminho dos arquivos coletados /var/monitoria/coleta
# Serviço associado monitoria.service | monitoria.service
# coleta date e hora
#
# Especifica qual o servidor de monitoraçao
srvhost=`cat /var/monitoria/srvhost`

# coleta uso de CPU
iostat |awk 'NR==4 {print (($6-100)*(-1))}' > /var/monitoria/coleta/cpu
# Altera a pontuaçao de ',' para '.' , para que o valor possa ser inserido no banco
sed -i 's/,/./g' /var/monitoria/coleta/cpu

# coleta memoria
free -m | awk 'NR==2 {print $2}' > /var/monitoria/coleta/memory-total # memoria total
free -m | awk 'NR==2 {print $3}' > /var/monitoria/coleta/memory-used # memoria usada
free -m | awk 'NR==2 {print $7}' > /var/monitoria/coleta/memory-available # memoria disponivel
free -m | awk 'NR==3 {print $2}' > /var/monitoria/coleta/swap-total # swap total
free -m | awk 'NR==3 {print $3}' > /var/monitoria/coleta/swap-used # swap usada
free -m | awk 'NR==3 {print $4}' > /var/monitoria/coleta/swap-free # swap livre

# coleta file systems
# ponto de montagem e %usada
df -h | awk '{print $6,$5}'| grep -vi use|grep -vi mounted > /var/monitoria/coleta/df

# coleta conexoes ativas
#netstat -natp

# coleta IO
# Coleta a partiçao e o uso dela
iostat -dx | grep -v "loop" | grep -v "fd0" | grep -v "scd" | awk 'NR==4 {print $1,$21}' >
/var/monitoria/coleta/io
# Altera a pontuaçao de ',' para '.' , para que o valor possa ser inserido no banco
sed -i 's/,/./g' /var/monitoria/coleta/io
# Envia os arquivos monitorados para o servidor banco
# rsync -rav --password-file=/etc/rsyncd.passwd monitor@$srvhost::data
/var/monitoria/coleta/coleta/$srvhost
```

**Lista 3 *Script rsync.sh*.**

```
pid file = /var/run/rsyncd.pid
log file = /var/run/rsyncd.log
uid = monitor
gid = monitor
read only = no
[data]
        path = /var/monitoria/coleta
        list = yes
        auth users = monitor
        secrets file = /etc/rsyncd.passwd
```

**Lista 4 *Script install-server-sh***

```
#!/bin/bash
# Pre-requisitos - Servidor
# Instalar o mysql, ubuntu LAMP, phpmyadmin e configurar o php
apt-get install apache2 -y
apt-get install mysql-server -y
apt-get install php libapache2-mod-php php-mysql -y
apt-get install phpmyadmin php-mbstring php-zip php-gd php-json php-curl -y
phpenmod mbstring
systemctl restart apache2

# instalar o iostat, rsync
apt-get install sysstat -y
apt-get install rsync -y

# Instalar tarefa na crontab para captura dos arquivos e atualizaçao do banco
if [ `crontab -l |grep srv-coleta.sh` -eq '']; then
crontab -l | { cat; echo "# Serviço de monitoraçao, caputra os arquivos"; echo "* * * * * sleep 10;
bash /usr/sbin/monitoria/srv-coleta.sh"; } | crontab -
else
echo '======================================================'
echo ''
echo 'CRONTAB SRV-COLETA JÁ ESTÁ CONFIGURADA'
echo ''
echo '======================================================'
fi

if [ `crontab -l |grep bdsend.sh` -eq '']; then
crontab -l | { cat; echo "# Envia a coleta para o banco de dados"; echo "* * * * * sleep 20; bash
/usr/sbin/monitoria/bdsend.sh"; } | crontab -
else
echo '======================================================'
echo ''
echo 'CRONTAB BDSEND JÁ ESTÁ CONFIGURADA'
echo ''
echo '======================================================'
fi
# Configurar o rsync (/etc/rsyncd.conf /etc/rsyncd.passwd)
cp ./srv-rsyncd.conf /etc/rsyncd.conf
echo senha > /etc/rsyncd.passwd
chmod 0640 /etc/rsyncd.conf
chmod 0600 /etc/rsyncd.passwd

# habilitar rsync e iniciar
systemctl enable rsync; systemctl start rsync
```

```
# criar usuario da monitoraçao
useradd -c "Usuario de monitoraçao" monitor

# Criar arquivo de senha do usuario da monitoraçao para acessar o banco
install -m 700 -d /home/monitor
install -m 600 /dev/null /home/monitor/monitor@localhost.cnf
echo "[client]" >> /home/monitor/monitor@localhost.cnf
echo 'password="SenhaInicial"' >> /home/monitor/monitor@localhost.cnf
chown -R monitor:monitor /home/monitor

# criar diretorio do script no /usr/sbin/monitoria
mkdir -p /usr/sbin/monitoria
chown root:monitor /usr/sbin/monitoria

# Enviar o script para o diretorio do script e atribuir a permissão de execuçao
cp ./srv-coleta.sh /usr/sbin/monitoria/
cp ./bdsend.sh /usr/sbin/monitoria/

# Script que cria os diretorios de monitoraçao:
cp ./update.sh /usr/sbin/monitoria/

# Atribui permissão de exeução aos sripts
#chmod +x /usr/sbin/monitoria/srv-coleta.sh
chmod +x /usr/sbin/monitoria/update.sh

# criar diretorio da coleta dos servidores
mkdir -p /var/monitoria/coleta
touch /var/monitoria/hosts.list
chown -R monitor:monitor /var/monitoria

# Cria o banco de dados e o usuario monitor no banco
mysql < create-bd

echo '======================================================='
echo ''
echo 'SCRIPT FINALIZADO'
echo ''
echo '======================================================='
```

**Lista 5 *Script* coleta.sh**

```
#!/bin/bash
# Script para coletar os arquivos dos servidores monitorados
# o comando do rsync ira capturar os dados monitorados.

for x in `cat /var/monitoria/hosts.list`
do (rsync -rav --password-file=/etc/rsyncd.passwd monitor@$x::data /var/monitoria/coleta/$x &)
done
```

**Lista 6 *Script rsyncd.***

```
pid file = /var/run/rsyncd.pid
log file = /var/run/rsyncd.log
```

**Lista 7 *Script update.sh.***

```
#!/bin/bash
# Script que cria os diretorios dos servidores a serem monitorados
for x in `cat /var/monitoria/hosts.list`
```

```
                do mkdir /var/monitoria/coleta/$x
                chown monitor:monitor /var/monitoria/coleta/$x
        done
```

**Lista 8** ***Script dbsend.sh.***

```
        #!/bin/bash
        # Script de envio da coleta para o banco de dados.
        #
        # Atribui os valores das variaveis
        bdsend="mysql  --defaults-extra-file=/home/monitor/monitor@localhost.cnf -u monitor monitoria
-e"
        dia=`date -I`
        hora=`date | awk '{print $5}'`

        # captura os ips dos servidores
        for x in `cat /var/monitoria/hosts.list`
            do
                # separa os dados do df para poder preencher o banco
                cat /var/monitoria/coleta/$x/df | awk '{print $1}' > /var/monitoria/coleta/$x/fs
                cat /var/monitoria/coleta/$x/df | awk '{print $2}' > /var/monitoria/coleta/$x/fs-uso

                # separa os dados do IO para poder preencher o banco
                cat /var/monitoria/coleta/$x/io | awk '{print $1}' > /var/monitoria/coleta/$x/io-part
                cat /var/monitoria/coleta/$x/io | awk '{print $2}' > /var/monitoria/coleta/$x/io-uso

                 # Atualiza as tabelas "ip, dia e hora", para que as tabelas relacionadas (coleta, filesystems)
possam ser atualizadas.
                                                # Atualiza as tabelas "ip para que as tabelas relacionadas (coleta,
filesystems) possam ser atualizadas.
                $bdsend "INSERT INTO ip VALUES ('$x');"

            # Atualiza a tablea filesystems
                count=1
                while [ $count -le `cat /var/monitoria/coleta/$x/fs | wc -l` ]
                    do
                    fs=`awk NR==$count < /var/monitoria/coleta/$x/fs`
                    used=`awk NR==$count < /var/monitoria/coleta/$x/fs-uso`
                    $bdsend "INSERT INTO filesystems VALUES ('$fs', '$used', '$hora', '$dia', '$x');"
                    count=$(( $count + 1))
                done

                # Atualiza a tabela coleta
                fcpu=`cat /var/monitoria/coleta/$x/cpu`
                fmem_total=`cat /var/monitoria/coleta/$x/memory-total`
                fmem_used=`cat /var/monitoria/coleta/$x/memory-used`
                fmem_available=`cat /var/monitoria/coleta/$x/memory-available`
                fswap_total=`cat /var/monitoria/coleta/$x/swap-total`
                fswap_used=`cat /var/monitoria/coleta/$x/swap-used`
                fswap_free=`cat /var/monitoria/coleta/$x/swap-free`
                        $bdsend "INSERT INTO coleta VALUES ('$fcpu', '$fmem_total', '$fmem_used',
'$fmem_available', '$fswap_total', '$fswap_used', '$fswap_free', '$hora', '$dia', '$x');"

                # Atualiza a tabela IO
                count=1
                while [ $count -le `cat /var/monitoria/coleta/$x/io | wc -l` ]
                    do
                    particao=`awk NR==$count < /var/monitoria/coleta/$x/io-part`
```

```
                uso=`awk NR==$count < /var/monitoria/coleta/$x/io-uso`

                $bdsend "INSERT INTO IO VALUES ('$particao', '$uso', '$hora', '$dia', '$x');"
                count=$(( $count + 1))
            done
done

echo '======================================================'
echo ''
echo 'SCRIPT FINALIZADO'
echo ''
echo '======================================================'
```

**Lista 9** ***Script createdb.sh*****.**

```
CREATE USER IF NOT EXISTS 'monitor'@'localhost'
IDENTIFIED BY 'SenhaInicial';

GRANT ALL PRIVILEGES
ON *.*
TO 'monitor'@'localhost';

CREATE DATABASE IF NOT EXISTS monitoria
DEFAULT CHARACTER SET UTF8MB4
DEFAULT COLLATE UTF8MB4_general_ci;

USE monitoria;
CREATE TABLE IF NOT EXISTS ip (
            ip VARCHAR(255) NOT NULL,
            PRIMARY KEY(ip)
) DEFAULT charset = UTF8MB4;


CREATE TABLE IF NOT EXISTS coleta (
            cpu FLOAT(25) NULL,
            mem_total FLOAT(25) NULL,
            mem_used FLOAT(25) NULL,
            mem_available FLOAT(25) NULL,
            swap_total FLOAT(25) NULL,
            swap_used FLOAT(25) NULL,
            swap_free FLOAT(25) NULL,
            hora TIME,
            dia DATE,
            par_ip VARCHAR(255),
            FOREIGN KEY(par_ip) REFERENCES ip(ip)
) DEFAULT charset = UTF8MB4;

CREATE TABLE IF NOT EXISTS filesystems (
            fs VARCHAR(255),
            used VARCHAR(255),
            hora TIME,
            dia DATE,
            par_ip VARCHAR(255),
            FOREIGN KEY(par_ip) REFERENCES ip(ip)
) DEFAULT charset = UTF8MB4;

CREATE TABLE IF NOT EXISTS IO (
```

```
        particao VARCHAR(30),
        uso VARCHAR(30),
        hora TIME,
        dia DATE,
        par_ip VARCHAR(255),
        FOREIGN KEY(par_ip) REFERENCES ip(ip)
) DEFAULT charset = UTF8MB4;
```

# Documento Digitalizado Restrito

## Artigo de TCC do aluno Marcos Paulo Coelho

**Assunto:** Artigo de TCC do aluno Marcos Paulo Coelho
**Assinado por:** Daiane Tomazeti
**Tipo do Documento:** Comprovante
**Situação:** Finalizado
**Nível de Acesso:** Restrito
**Hipótese Legal:** Informação Pessoal - dados pessoais e dados pessoais sensíveis (Art. 31 da Lei nº 12.527/2011)
**Tipo do Conferência:** Cópia Simples

Documento assinado eletronicamente por:

- **Daiane Mastrangelo Tomazeti**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 15/12/2022 15:59:33.

Este documento foi armazenado no SUAP em 15/12/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

**Código Verificador:** 1186963
**Código de Autenticação:** 05dbb57d38