Documento de transferência do Rabbit Monitoring System

Partes do sistema

1. Sistema **Python**
2. Sistema **Web**
3. Banco de dados **MYSQL**

Características das partes do Python

O sistema **Python é dividido em três partes,** a parte principal está cadastrando no Banco de Dados mySQL, em um modelo de dados desatualizado, anterior as modificações que fizemos. Ele está separado em funções para cada parte do software.

- **main.py (core)**
    1. **run_sql_command()** é responsável para fazer a execução do código para executar no mysql
    2. **get_current_date()** é responsável por colher o horário atual pelo comando shell “date”. Ele separa os pontos em um veor e colhe as horas. Dependendo do sistema operacional, o ponto do vetor pode mudar de [3] para [4]. O sistema de automatização dessa parte seria trabalhado
    3. **get_directory() [INUTILIZADA]** é responsável por colher o diretório atual da aplicação. Pensamos em cria-la para executar as outras partes do

software Python.

4. **get_system_type()** é responsável por colher e printar os dados do sistema operacional, sendo os dados: **horário atual, nome do SO e tipo do sistema**
5. **get_memory_ram()** é responsável por colher os dados da memória ram, printar os dados colhidos e inserir os dados no banco de dados, sendo os dados: **total do uso da ram, uso atual, porcentagem e data atual**
6. **get_cpu()** é responsável por colher os dados da CPU, sendo que uma das formas que esses dados são colhidos é por um algorítmo, no qual utiliza o comando **lscpu (**utilizamos para pegar o nome da CPU, suas threads por núcleo, sua arquitetura e seus núcleos .) e inserior os dados no banco de dados, sendo os dados: **Frequência atual da Cpu, temperatura atual e a data atual**
7. **get_disk()** é responsável por colher todos os dados do disco, printa-los e depois inserir no banco de dados, os dados são: **capacidade do disco, espaço livre do disco, espaço usado do disco, porcentagem de uso e a data atual**
8. **get_disk()** é responsável por colher todos os dados do disco, printa-los e depois inserir no banco de dados, os dados são: **capacidade do disco, espaço livre do disco, espaço usado do disco, porcentagem de uso e a data atual**
9. **network()** é responsável por colher os dados de internet e printa-los no terminal, sendo os dados: **endereço ip, hostname, velocidade de download, velocidade de upload**, as duas velocidades de internet tem um problema de conversão.
10. **processo_total()** é responsável por executar constantemente todas as demais funções, utilizando um laço de repetição eterno. Não tinhamos conhecimento da estrutura **with** quando o software foi criado, mas pretendiamos substitui-lo pela

> 📜 OBS: O software tem um pequeno problema que, em alguns terminais, ele acaba piscando. Pelos testes feitos, o terminal que ele melhor funciona é o do KDE . Além disso, o software também **cria um log em texto**

▼ **grafics.py (gráficos)**

1. **att_date()** é responsável por atualizar o horário atual, com um tempo de espera de 1 segundo
2. **config()** é responsável por toda parte de configuração da biblioteca mysql.
3. **select_data(componente)** é responsável por retornar os dados que foram inseridos no Banco de dados, contudo, sua aplicabilidade acabou dando alguns problemas, pois o select acabava desatualizado. Por isso, não estamos pegando os dados do Banco de dados, ainda que pretendiamos fazer a correção desse problema
4. **psutil_data(arr, component)** é responsável por retornar os valores dos componentes atuais, a partir de qual componente que o usuário inseriu, como (cpu, ram, disco).
5. **animate(i)** é o core da aplicação, é responsável por receber os dados e, após isso, prepara para plotar por meio da função plotCharts. Além disso, cada função **%_full_data**, teoricamente, receberia os dados do banco de dados, contudo, o erro no select_data acabou impossibilitando isso. Os gráficos são plotados por meio de uma função que colhe dos dados local, como **%_y**
6. **get_data(arr, item)** é responsável por tratar os itens e retorna-los em um dicionário
7. **handle_data(arr, dict_item)** trata a array e retorna como um float, dict_item é a parte do dicionário que será convertida em float.
8. **plotCharts(x,y, component, limit)** plota os gráficos a partir do componente (cpu, ram, disco). O limit é o limite do eixo x.

▼ **data_analysis.py (análise de dados)**

1. **chart_config(ax,xlabel,ylabel,title)** é responsável por, após escolher o eixo, escolher o nome da linha x, da linha y e do título do gráfico.
2. **temperature_per_frequence()** plota um gráfico scatter, com a utilização da função LinearRegression(), percentente e biblioteca sklearn (utilizada para estatística e IA). Após isso, os dados x,y são trabalhados pelo método **fit**, que trabalha esses dados. Após isso, um novo val. para y é criado, a partir da função **predict**, que prevê linearmente os dados. Após isso, os gráficos são plotados pela função **chart_config()**
3. **frequence_per_time()** recebe um dado da semana, sortados pelo id e limitados pelos 0 → 25 últimos. Depois, trata os valores **string** de dados do .**csv,** convertendo os para **Timestamp**, e depois, repetindo os passos para regressão linear, citados acima.
4. **avg_frequence_weekend_day()** recebe os dados do dia da semana, converte pra dia da semana pela função **get_weekend_day(data)** e plota os gráficos por dia da semana. Está mal otimizado.
5. **get_weekend_day(data)** converte a data em dias para dia da semana, tornando possível plotar o gráfico de média por dia da semana
6. **main()** plota todas as funções e permite um menu para o usuário, para que ele possa escolher uma dessas funções.

> 📜 Os dados do processador.csv tem apenas os dados do processador, o select foi feito para teste. Além disso, essa análise de dados ainda está embrionaria, apenas os dados da CPU estão disponíveis, e, por fim, apenas os dados de quinta_feira estão disponíveis no arquivo.csv

O nosso **Banco de Dados** contém diferentes entidades, e durante o tempo, sofreu algumas alterações. Temos duas versões, e o sistema **Python** ainda utiliza a versão antiga. Pretendiamos trocar para a versão nova. As entidades em nosso banco de dados são:

A nossa página web se encontra estática, sem o login e o cadastro funcionando, contudo, as páginas de **home, cadastro e login** estão responsivas e funcionando em dispositivos **mobile.** Não temos gráficos da dashboard, contudo, temos um portal para seleção de cada ponto do servidor.

Características das partes do Sistema Web

As páginas de Login e Cadastro não possuía nenhuma funções, porém existia na Index (Home) e Dashboard.

Index (Home)

Para deixar mais bonito o nosso projeto, existe uma função chamada **reveal** que ao dar um scroll na página, irá rodar uma função que irá verificar a posição da tela e ao atingir uma certa posição para aparecer o elemento, faz com que adicione a classe “active” em elementos com a classe “reveal” e ao sair do ponto de vista, remove a classe adicionada.

Além dessa função todas as outras são para o carrossel (**moveImageCarrossel**, **setImageCarrosselAnimation**, **setCarrosselTextOnclick** e **tradeImageCarrosselUsingText**), pois era uma nova funcionalidade não testada então por esse motivo existe várias funções, porém é possível diminuir o seu tamanho para algo mais simples. O carrossel funciona através de um elemento **ul** que está como position absolute com imagens dentro dele, ao clicar na imagem, vai capturar o elemento clicado e vai ler o seu **ID**, após isso irá verificar o qual é o número do id para que mude a variável do css, que automaticamente muda a posição da imagem. As funções **tradeImageCarrosselUsingText** e **setCarrosselTextOnclick** servem para modificar a imagem clicando no texto, onde seriam os passos de como funciona o nosso sistema, como um diagrama de visão, mas se quiserem mudar, pode mudar,

uma coisa importante sobre o seu funcionamento, é que ele funciona do mesmo jeito que a imagem (Através do Id do elemento).

Também possui algumas funções no caminho Site/public/assets/js/funcoes.js, as funções **showPassword** e **showRewritePassword** serviam para mostrar a senha que o usuário estava digitando, clicando no icone do fontAwesome que estava do lado da senha, porém não foi implementado a tempo antes que o projeto fosse iniciado. A última função **goToLoginOrSignUp** serve para redirecionar o login utilizando um botão, só uma gambiarra para não ter utilizar o elemento **a**, pois poderia bugar o html que já estava pronto.

Dashboard