**Documentação – doc\_alerta\_tracking.py**

**Área relacionada**

* TI\_MONITORAMENTO

**Frequência**

* O script é executado de acordo com a necessidade, sem uma frequência específica.

**Servidor**

* Não especificado

**Resumo da aplicação**

Esta aplicação consiste em um script Python (doc\_alerta\_tracking.py) que funciona como um monitor de saúde (health check) para o banco de dados do sistema de Tracking. Seu único objetivo é verificar se o banco de dados TRACKING está online e respondendo a conexões. Para isso, ele tenta se conectar e executar uma consulta extremamente simples. Se a conexão ou a consulta falharem, o script assume que o banco de dados está com problemas e imediatamente dispara um alerta. O alerta é inserido na tabela alerta\_mensagens (no banco TORRE\_CONTROLE) para notificar a equipe de TI responsável sobre a indisponibilidade.

**Pré-requisitos**

* **Bibliotecas Python:**
* **Não Nativas:** python-dotenv. Nota: Requer um driver de banco de dados (como pyodbc ou psycopg2), que é utilizado dentro do módulo 'tools'.
* **Nativas:** os, sys, datetime.
* **Arquivos Necessários:**
* Módulo 'tools' acessível via path.
* Um arquivo .env no diretório de ferramentas para armazenar as credenciais dos bancos de dados.
* **Parâmetros de Linha de Comando:** O script precisa ser executado com parâmetros que definem o template da mensagem e os destinatários: `[template] [telefone] [telefone] ...`

**Fluxo de Execução Detalhado**

**Inicialização e Leitura de Parâmetros:** O script define os caminhos de sistema, carrega as variáveis de ambiente e lê os parâmetros passados via linha de comando para determinar o modelo da mensagem e a lista de contatos que devem ser notificados em caso de falha.

**Verificação de Saúde do Banco de Dados (Health Check):** Esta é a etapa central. O script tenta se conectar ao banco de dados TRACKING. Se a conexão for bem-sucedida, ele executa uma consulta mínima (`SELECT 1;`), que não busca dados reais, mas serve apenas para confirmar que o banco está funcional e apto a processar requisições. Toda essa operação é envolvida por um bloco `try...except` para capturar qualquer tipo de erro de conexão ou execução.

**Análise do Resultado:** Uma falha é determinada por duas condições: 1) Se ocorrer qualquer exceção durante a tentativa de conexão ou de execução da consulta. 2) Se a consulta for executada com sucesso, mas não retornar o resultado esperado (uma linha), indicando um comportamento anômalo. Se qualquer uma dessas condições for verdadeira, uma flag interna de erro é ativada.

**Tratamento de Sucesso:** Se, após a verificação, a flag de erro permanecer desativada, o script considera que o banco de dados TRACKING está saudável e encerra sua execução imediatamente, sem realizar nenhuma ação.

**Geração de Alerta em Caso de Falha:** Se a flag de erro for ativada, o script entra no modo de alerta. Ele constrói uma mensagem padrão informando qual banco de dados (TRACKING) parou de responder e o horário da ocorrência.

**Enfileiramento do Alerta:** O script então se conecta a um segundo banco de dados, o TORRE\_CONTROLE. Ele percorre a lista de destinatários recebida como parâmetro e, para cada um, insere uma nova linha na tabela `alerta\_mensagens`. Essa linha contém a mensagem de falha e os detalhes para o envio, efetivamente "enfileirando" o alerta para ser despachado por um sistema de notificação separado.

**Finalização:** Após registrar todos os alertas, a conexão com o banco TORRE\_CONTROLE é fechada, e o script termina.

**Funções Auxiliares Principais**

A lógica do script é executada de forma linear em um fluxo único e não define funções auxiliares próprias. Ele depende inteiramente de funções do módulo externo 'tools' para tarefas padronizadas, como a conexão com os bancos de dados (t.connect\_db), o fechamento dessas conexões (t.close\_db) e o registro de logs (t.log).