**Documentação do Projeto: Discord Notifier**

**Visão Geral**

Este projeto define uma classe Python chamada DiscordNotifier, que é responsável por enviar notificações para um servidor Discord usando um webhook. O webhook é configurado diretamente na classe, e o método enviar\_notificacao permite enviar mensagens de diferentes tipos de status (informação, sucesso ou erro) com informações sobre um processo específico.

**Objetivo**

O objetivo dessa classe é permitir que sistemas automatizados, como pipelines de dados, monitoramento de servidores ou qualquer outra aplicação, envie notificações em tempo real para um canal do Discord sempre que um processo seja iniciado, concluído ou se ocorrer algum erro.

**Tecnologias Utilizadas**

* **Python 3.11**: A linguagem principal do projeto.
* **requests**: Biblioteca utilizada para enviar requisições HTTP.
* **datetime**: Módulo da biblioteca padrão do Python para manipulação de datas e horas.

**Estrutura do Código**

**1. Importações**

python

CopiarEditar

import requests

from datetime import datetime

As bibliotecas importadas são:

* requests: Para realizar requisições HTTP (POST) ao webhook do Discord.
* datetime: Para formatar o timestamp atual no padrão ISO8601.

**2. Classe DiscordNotifier**

A classe principal do projeto é a DiscordNotifier, que contém os métodos necessários para enviar as notificações ao Discord.

**Métodos**

* **\_\_init\_\_(self)**:
  + Inicializa a classe, configurando a URL do webhook do Discord.
  + A URL do webhook está hardcoded no código (o que é algo a ser melhorado para segurança).

python

CopiarEditar

def \_\_init\_\_(self):

self.webhook\_url = "https://discord.com/api/webhooks/1333091809139621939/aMNr3JAmrpzklttIUStqdp8W2BNPU91GNRzHPaPe3NjBiqHs69vVqbyw34I\_FY0Aaf01"

* **enviar\_notificacao(self, mensagem, processo, status="info")**:
  + Este método é responsável por enviar uma notificação ao Discord.
  + Parâmetros:
    - mensagem: Texto a ser enviado na notificação.
    - processo: Nome do processo ou tarefa que está sendo notificada.
    - status: Status da mensagem ("info", "sucesso" ou "erro"). O valor padrão é "info".
  + O método seleciona a cor para o embed com base no status fornecido e monta a estrutura da mensagem.
  + A requisição HTTP é enviada utilizando o método requests.post para a URL do webhook configurado na inicialização.

python

CopiarEditar

def enviar\_notificacao(self, mensagem, processo, status="info"):

cores = {

"info": 3447003, # Azul

"sucesso": 3066993, # Verde

"erro": 15158332 # Vermelho

}

timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

conteudo = {

"username": "Bender", # Nome do bot

"embeds": [

{

"title": f"Processo: {processo.capitalize()}",

"description": mensagem,

"color": cores.get(status, 3447003), # Cor padrão: azul

"timestamp": timestamp,

}

]

}

resposta = requests.post(self.webhook\_url, json=conteudo)

if resposta.status\_code == 204:

print("Notificação enviada com sucesso!")

else:

print(f"Erro ao enviar notificação: {resposta.status\_code} - {resposta.text}")

**3. Exemplo de Uso**

O código abaixo mostra um exemplo de uso da classe DiscordNotifier para enviar uma notificação informando que um processo de ETL foi iniciado.

python

CopiarEditar

if \_\_name\_\_ == "\_\_main\_\_":

notifier = DiscordNotifier()

notifier.enviar\_notificacao("O processo de ETL foi iniciado.", processo='XPTO', status="info")

**Fluxo do Processo**

1. A classe DiscordNotifier é inicializada.
2. O método enviar\_notificacao é chamado com os parâmetros necessários: uma mensagem, um nome de processo e um status.
3. O método cria a estrutura da mensagem em formato JSON e envia uma requisição POST para o webhook do Discord.
4. Caso a requisição seja bem-sucedida (status HTTP 204), uma mensagem de sucesso é impressa. Caso contrário, é impresso o erro com o código de resposta.

**Melhorias Possíveis**

* **Segurança**: Atualmente, a URL do webhook está hardcoded no código. Seria melhor extrair essa URL de uma variável de ambiente ou um arquivo de configuração para evitar exposições acidentais.
* **Personalização**: O nome do usuário ("Bender") é fixo. Isso pode ser personalizado para refletir melhor o nome do sistema ou do bot.
* **Logs**: Implementar logs mais detalhados (por exemplo, utilizando a biblioteca logging) para registrar todas as notificações enviadas e possíveis falhas.
* **Tratamento de Erros**: Melhorar o tratamento de exceções em casos de falhas na requisição HTTP.