**ExecutionsRobot**

**ExecutionsRobot** é um robô do tipo console desenvolvido em C# para gerenciar e executar processos no backend de um sistema de monitoramento de execuções. Ele processa fluxos automaticamente, interage com o banco de dados, atualiza status, insere registros no log e fornece suporte para intervenções manuais realizadas pelo sistema de monitoramento.

**Funcionalidades Principais**

1. **Processamento Automático de Fluxos:**

Executa a procedure spGerenciarExecucoes no banco de dados para verificar e processar execuções em diferentes estados (pendentes, em fila ou em erro).

Prioriza execuções pendentes para otimizar o fluxo de trabalho.

1. **Atualização de Status:**

Atualiza o status das execuções no banco de dados com valores como Concluído, Executando, Erro, ou Em Fila.

Garante que as execuções processadas pelo robô sejam registradas corretamente.

1. **Registro de Observações:**

Adiciona observações relevantes no log das execuções, diferenciando ações automáticas de ações manuais, como:

* + - APROV\_AUTOMATICA para fluxos automáticos.
    - Retomado pelo robô para execuções retomadas após remoções manuais.

1. **Logs de Processamento:**

Gera logs detalhados em um arquivo chamado Log\_exec\_proc.log, localizado no diretório C:\Logs, com mensagens sobre execuções processadas, erros encontrados e ações realizadas.

1. **Suporte para Integração com Monitoramento:**

O robô é integrado ao sistema de monitoramento de execuções, alimentando o frontend com dados atualizados sobre o status dos fluxos.

**Como Funciona**

1. **Inicialização do Robô:**

Ao ser iniciado, o robô executa a procedure spGerenciarExecucoes, que verifica as execuções pendentes no banco de dados.

As execuções são processadas com base no status atual:

* + - Execuções em fila são iniciadas.
    - Execuções com erro podem ser reprocessadas (se configurado).

1. **Atualização de Logs e Banco de Dados:**

A cada execução, o robô atualiza os registros no banco CadastroDB na tabela TBy9\_LogExecucao, informando:

* + - Status atual.
    - Observações.
    - Horários de início e fim.

1. **Geração de Logs:**

Para cada ação realizada, uma mensagem é gravada no arquivo Log\_exec\_proc.log, facilitando a auditoria e o rastreamento de problemas.

**Estrutura do Banco de Dados**

O robô utiliza a tabela TBy9\_LogExecucao no banco de dados **CadastroDB**, com a seguinte estrutura:

| **Coluna** | **Tipo de Dados** | **Descrição** |
| --- | --- | --- |
| ID\_Execucao | INT | Identificador único da execução. |
| NomeProcedure | NVARCHAR(255) | Nome da procedure associada à execução. |
| DataInicio | DATETIME | Data e hora de início da execução. |
| DataFim | DATETIME (NULL) | Data e hora de término da execução (se houver). |
| Status | NVARCHAR(50) | Status da execução (Concluído, Erro, etc). |
| Duracao | INT (NULL) | Duração em segundos. |
| Observacao | NVARCHAR(MAX) | Observações e histórico da execução. |

**Tecnologias Utilizadas**

* **C#**: Linguagem principal para o desenvolvimento do robô.
* **SQL Server**: Banco de dados relacional para armazenar e gerenciar execuções.
* **Stored Procedures**: Para realizar as operações no banco de forma otimizada.
* **Logs em Arquivo**: Utilizando System.IO para registro de logs em tempo real.

**Logs**

Os logs são gravados no arquivo Log\_exec\_proc.log, com mensagens detalhadas de cada ação realizada pelo robô. Exemplo de mensagens:

03/11/2024 14:00:01: Execução 1501 iniciada automaticamente.

03/11/2024 14:00:06: Execução 1501 concluída automaticamente.

## 03/11/2024 14:01:00: Execução 1502 reiniciada manualmente pelo sistema. **Como Rodar**

1. **Pré-requisitos:**

.NET SDK instalado.

Banco de dados SQL Server configurado e conectado.

1. **Configuração:**

Ajuste a string de conexão no código para apontar para o banco CadastroDB.

1. **Execução:**

Compile e execute o programa no terminal:  
  
comando> dotnet run

## Benefícios do Projeto

* **Automatização**: Reduz a necessidade de intervenções manuais no processamento de fluxos.
* **Integração**: Funciona em conjunto com o sistema de monitoramento, permitindo uma visão centralizada do ambiente.
* **Confiabilidade**: Gera logs detalhados, facilitando auditorias e investigações.
* **Escalabilidade**: Pode ser configurado para suportar diferentes cenários de carga e ambientes.

## Possíveis Melhorias Futuras

* Implementar paralelismo para processamento simultâneo de múltiplas execuções.
* Adicionar suporte a notificações (e-mails ou alertas) para erros críticos.
* Permitir configuração dinâmica de parâmetros (ex.: tempo de espera, limite de execuções).