Requisitos Process Monitor

Link Consulting – Tecnologias de Informação, S.A.

Janeiro 2025

Índice

1 Introdução

Este documento contém os requisitos do módulo “Process Monitor” do projeto eProcess, sob a forma de casos de uso.

2 Descrição Geral

2.1 Descrição Process Monitor

O **Process Monitor** é um sistema centralizado concebido para **registar**, **analisar** e **monitorizar** eventos gerados por motores de processos (ou *engines*). Este sistema permite a recolha de dados em tempo real, como o início e fim de atividades, alarmes, e outros eventos relevantes, armazenando-os numa base de dados estruturada.

Através de técnicas de **Process Mining** (utilizando a biblioteca open-source PM4Py), o Monitor é capaz de analisar os dados recolhidos, gerando métricas de desempenho, identificando violações de SLA (Service Level Agreements), e criando visualizações de processos (ex.: diagramas BPMN). Os resultados são disponibilizados em ficheiros JSON, que integrando com o "Process Designer", são utilizados para dashboards (blueprints).

O sistema é modular, composto por **três** componentes principais:

1. **Módulo de Receção de Mensagens:** Recebe notificações dos motores de processos via Kafka.
2. **Base de Dados:** Armazena dados de eventos, modelos de processos, e alarmes.
3. **Módulo de Análise:** Processa os dados utilizando PM4Py para gerar métricas e visualizações.

3 Casos de Uso

3.1 Registo de Eventos

- **Atores**: Motores de Processo, Monitor de Processos.
- **Pré-condições**:
  - Os motores de processo estão em execução e a gerar eventos.
  - O Monitor de Processos está operacional e pronto para receber mensagens.
- **Fluxo**:
  1. Um motor de processo gera um evento (ex.: início/fim de uma atividade ou alarme).
  2. O motor envia uma mensagem (ex.: via Kafka) para o Monitor de Processos.
  3. O Monitor de Processos recebe a mensagem e regista o evento na tabela **EventLog**.
- **Pós-condições**:
  - O evento é registado na base de dados.
  - O evento está disponível para análise futura.

3.2 Análise de Dados

- **Atores**: Monitor de Processos, Analista de Dados.
- **Pré-condições**:
  - Os dados dos eventos foram registados na tabela **EventLog**.
  - O Monitor de Processos tem acesso à biblioteca PM4Py.
- **Fluxo**:
  1. O Analista de Dados solicita uma análise (ex.: caso específico, análise de atividades ou análise global do processo).
  2. O Monitor de Processos extrai os dados da tabela **EventLog**.
  3. O Monitor de Processos processa os dados utilizando o PM4Py para calcular métricas (ex.: duração média, violações de SLA).
  4. Os resultados são gerados em formato JSON.
- **Pós-condições**:
  - Os resultados da análise estão disponíveis em formato JSON.
  - As métricas e visualizações estão prontas para revisão.

3.3 Geração de Saída em JSON

- **Atores**: Monitor de Processos, Analista de Dados, Sistemas Externos.
- **Pré-condições**:
  - A análise de dados foi concluída.
- **Fluxo**:
  1. O Monitor de Processos gera um ficheiro JSON com base no modo de análise (caso específico, análise de atividades ou análise global do processo).
  2. O ficheiro JSON inclui métricas, detalhes das atividades e visualizações (ex.: diagramas BPMN em Base64).
  3. O ficheiro JSON é disponibilizado ao Analista de Dados ou a sistemas externos.
- **Pós-condições**:
  - Um ficheiro JSON com os resultados da análise é criado.
  - O ficheiro pode ser utilizado para dashboards, relatórios ou processamento adicional.

3.4 Arquitetura Modular

- **Atores**: Administrador de Sistemas, Monitor de Processos.
- **Pré-condições**:
  - O sistema Monitor de Processos está instalado e configurado.
- **Fluxo**:
  1. O Administrador de Sistemas configura o **Módulo de Receção de Mensagens** para receber mensagens dos motores.
  2. O **Módulo de Base de Dados** é configurado para armazenar registos de eventos, modelos de processo e dados de alarmes.
  3. O **Módulo de Análise** é configurado para processar dados utilizando scripts Python e o PM4Py.
- **Pós-condições**:
  - O Monitor de Processos está pronto para receber, armazenar e analisar dados de processos.

3.5 Estrutura da Base de Dados

- **Atores**: Administrador de Sistemas, Monitor de Processos.
- **Pré-condições**:
  - A base de dados está instalada e acessível.

- **Fluxo**:
    1. O Administrador de Sistemas cria as tabelas necessárias (EventLog, ProcessTemplate, AlarmTemplate, AlarmInstance).
    2. O Monitor de Processos começa a registar eventos e alarmes nas respetivas tabelas.
- **Pós-condições**:
    - A base de dados é preenchida com dados de processos e alarmes.
    - Os dados estão prontos para análise.

3.6 Comunicação por Mensagens

- **Atores**: Motores de Processo, Monitor de Processos.
- **Pré-condições**:
    - O Kafka está configurado e em execução.
    - Os motores de processo estão a gerar eventos.
- **Fluxo**:
    1. Um motor de processo gera um evento (ex.: início/fim de uma atividade ou alarme).
    2. O motor envia uma mensagem Kafka para o Monitor de Processos.
    3. O Monitor de Processos processa a mensagem e atualiza a base de dados.
- **Pós-condições**:
    - O evento é registado na base de dados.
    - O Monitor de Processos está pronto para processar a próxima mensagem.

3.7 Modos de Análise

- **Atores**: Analista de Dados, Monitor de Processos.
- **Pré-condições**:
    - Os dados dos eventos estão disponíveis na tabela **EventLog**.
- **Fluxo**:
    1. O Analista de Dados seleciona um modo de análise (caso específico, análise de atividades ou análise global do processo).
    2. O Monitor de Processos extrai os dados relevantes e processa-os utilizando o PM4Py.
    3. Os resultados são gerados em formato JSON.
- **Pós-condições**:

- Os resultados da análise estão disponíveis em formato JSON.
- O Analista de Dados pode rever os resultados.

3.8 Visualizações

- **Atores**: Analista de Dados, Monitor de Processos.
- **Pré-condições**:
  - A análise de dados foi concluída.
- **Fluxo**:
  1. O Monitor de Processos gera visualizações (ex.: diagramas BPMN, Heuristics Net) utilizando o PM4Py.
  2. As visualizações são codificadas em Base64 e incluídas no ficheiro JSON de saída.
- **Pós-condições**:
  - As visualizações estão disponíveis no ficheiro JSON.
  - O Analista de Dados pode visualizar os diagramas.

3.9 Monitorização de SLA

- **Atores**: Monitor de Processos, Motores de Processo.
- **Pré-condições**:
  - Os modelos de alarme estão definidos na tabela **AlarmTemplate**.
- **Fluxo**:
  1. O Monitor de Processos deteta uma violação de SLA com base nos limiares definidos.
  2. A violação é registada na tabela **AlarmInstance**.
  3. A violação é incluída nos resultados da análise.
- **Pós-condições**:
  - As violações de SLA são registadas e disponíveis para análise.
  - Alertas podem ser acionados com base nas violações.

3.10 Integração com PM4Py

- **Atores**: Analista de Dados, Monitor de Processos.
- **Pré-condições**:
  - O PM4Py está instalado e configurado.

- **Fluxo**:
    1. O Analista de Dados solicita uma análise.
    2. O Monitor de Processos utiliza o PM4Py para processar os dados e gerar métricas e visualizações.
- **Pós-condições**:
    - Os resultados da análise estão disponíveis em formato JSON.
    - As métricas e visualizações estão prontas para revisão.

3.11 Análise Comparativa de Ferramentas de Process Mining

- **Atores**: Administrador de Sistemas, Analista de Dados.
- **Pré-condições**:
    - Várias ferramentas de process mining (ex.: PM4Py, ProM, Apromore) estão disponíveis.
- **Fluxo**:
    1. O Administrador de Sistemas avalia as ferramentas com base na tabela de comparação fornecida.
    2. O Analista de Dados seleciona a ferramenta mais adequada para a análise.
- **Pós-condições**:
    - A ferramenta selecionada é utilizada para process mining.
    - Os resultados da análise são gerados.

3.12 Fluxos de Exemplo

- **Atores**: Motores de Processo, Monitor de Processos.
- **Pré-condições**:
    - Os motores de processo estão em execução e a gerar eventos.
- **Fluxo**:
    1. Um motor envia uma mensagem "START" para o Monitor de Processos.
    2. O Monitor de Processos regista o evento e verifica se existem alarmes.
    3. O motor envia uma mensagem "END", e o Monitor de Processos atualiza o registo do evento e verifica violações de SLA.
- **Pós-condições**:
    - Os eventos são registados, e os alarmes são monitorizados.

- Os resultados da análise estão disponíveis.

3.13 Escalabilidade e Flexibilidade

- **Atores**: Administrador de Sistemas, Monitor de Processos.
- **Pré-condições**:
  - O sistema está em execução e a processar dados.
- **Fluxo**:
  1. O Administrador de Sistemas adiciona novas tabelas (ex.: UserTemplate, GroupTemplate) para suportar funcionalidades adicionais.
  2. O Monitor de Processos adapta-se às novas tabelas e continua a processar dados.
- **Pós-condições**:
  - O sistema é expandido com novas funcionalidades.
  - O processamento de dados continua sem interrupções.

3.14 Considerações Finais

- **Atores**: Administrador de Sistemas, Analista de Dados.
- **Pré-condições**:
  - O Monitor de Processos está operacional.
- **Fluxo**:
  1. O Administrador de Sistemas garante que o sistema está centralizado e pronto para a recolha de dados.
  2. O Analista de Dados utiliza o sistema para gerar ficheiros JSON para monitorização e dashboards.
- **Pós-condições**:
  - O sistema está totalmente operacional e a gerar resultados de análise.
  - Os ficheiros JSON estão disponíveis para utilização futura.