Relatório
Trabalho Pratico 1

Afonso Almeida
Nº 26424

Integração de Sistemas de Informação
Licenciatura em Engenharia de Sistemas Informáticos
2025/26

Índice

Relatório do Projeto Prático I

Processamento ETL de Notificações de Eventos Nucleares

Unidade Curricular: Integração de Sistemas de Informação
Curso: Licenciatura em Engenharia de Sistemas Informáticos
Ano Letivo: 2025/26
Equipa: Afonso Almeida | a26424
Data de Entrega: 18 de outubro de 2025

1. Enquadramento

1.1 Composição da Equipa

A equipa é composta por apenas 1 elemento.

- **Elemento 1:** [Afonso Almeida] – a26424

O projeto foi realizado no âmbito da unidade curricular “Integração de Sistemas de Informação”, que é lecionada no curso “Licenciatura de Engenharia de Sistemas Informáticos” no Instituto Politécnico do Cávado e do Ave.

2. Problema

2.1 Descrição do Cenário

Há central nuclear que possui tem um método muito ineficiente de registar falhas e eventos que acontecem na central.

O registo é feito no formato de um documento de texto onde todos os valores e colunas estão separados por “pipes” (|). Os valores em si estão mal tratados, com vários dados vazios, inuteis e mal formatados.

Este projeto envisiona resolver estes problemas através de um processo ETL que não só lê e trata dos dados num flow na aplicação “KNIME” como também possui uma integração com a plataforma “Node-Red” através do framework “Flask API” em “Python”.

A plataforma vai possibilitar a leitura dinâmica dos dados através do “Knime” onde depois podem ser visualisados de maneira mais apelativa utilizado as ferramentas de criação de dashboards.

2.2 Objetivos Específicos

O projeto tem como objetivo atingir os seguintes critérios:

- Processar ficheiros CSV e TXT com dados de eventos nucleares
- Normalizar dados temporais com diferentes fusos horários
- Extrair informação estruturada de um ficheiro complexo
- Gerar outputs em JSON para usar na integração com APIs
- Implementar dashboards para visualização de dados

2.3 Dados de Entrada

Os dados de entrada do projeto consiste num ficheiro de dados reais, armazenado no website da NRC(Nuclear Regulatory Comission) que licensia e regula o uso de energia nuclear.

Os detalhes dos ficheiros são:

- Webpage do ficheiro: https://www.nrc.gov/cdn/data/eventnotification/event-notification-rpt-lastmonth.txt
- Dados delimitados por “pipes(|)”
- Dados incluem eventos médicos, falhas de equipamento, danos em instrumentos
- 71 Colunas
- 68 Rows

3. Estratégia Utilizada

3.1 Plataformas utilizadas

Plataformas Principais:

- KNIME Analytics Platform : A plataforma Knime permite a criação de workflows para processamento de data.
- Node-RED : A plataforma Node-Red permite a criação de aplicações que recolhem, transformam e ajudam a visualizar data.

Ferramenta Complementar:

- Flask API : A plataforma Flask permite a criação de frameworks para execução de API's do tipo REST.

3.2 Operadores e Processos Principais

3.2.1 Operadores de Transformação

- **CSV Reader**: Leitura através de um link.
- **String Manipulation**: Utilização de regex, replace etc...
- **Column Filter**: Seleção e filtragem de colunas
- **Date & Time Conversion**: Normalização de data e horas
- **JSON Writer**: Conversão de dados para Json
- **Column combiner/divider**: Facilitar a normalização de colunas

3.2.2 Processos de Controlo

- **Job Orchestration**: Orquestração de workflows
- **Error Handling**: Gestão de exceções e dados inválidos
- **Quality Validation**: Validação de qualidade de dados
- **Log Generation**: Geração de logs de processamento

4. Transformações

4.1 Arquitetura do Workflow KNIME

O processo ETL foi implementado através de um workflow estruturado no KNIME Analytics Platform, composto por 4 fases principais:

EXTRAÇÃO →	**LIMPEZA** →	**TRANSFORMAÇÃO** →	**ENVIO/DOWNLOAD**
↓	↓	↓	↓
Csv reader	Nós para filtrar	Normalização dos dados	POST Request

4.2 Operadores de Transformação Implementados

4.2.1 Leitura e parsing de dados

O diagrama KNIME começa por ler os dados:

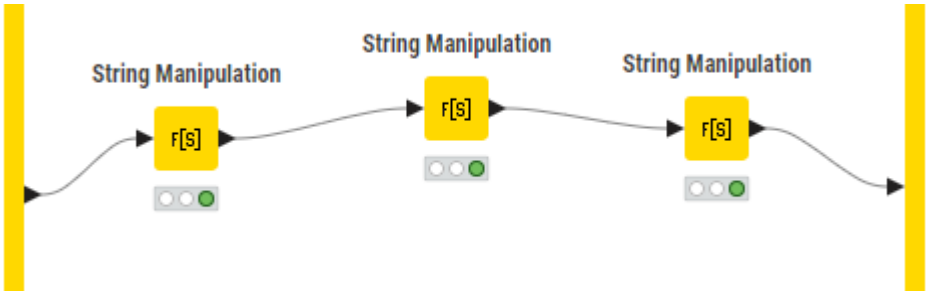
- **Fonte**: https://www.nrc.gov/cdn/data/eventnotification/event-notification-rpt-lastmonth.txt
- **Formato**: Pipe-delimited (|) com 40+ colunas
- **Configuração KNIME**:
  - Delimitador: "|"
  - Deteção automática de schema
  - Processamento de dados semi-estruturados com múltiplas colunas
  - Deteção automática de schema para 40+ colunas
  - Normalização de valores vazios:

String: Substituir com o valor mais frequente

Interger: Substituir com a Mediana

4.2.2 TRANSFORMAÇÃO DAS DATAS E HORAS

- Implementação de padding para corrigir formatos inconsistentes (7 vs 8 dígitos)



Código: padLeft($NomeDaColuna$, 8, "0")

Resultado: Input: "9:16:37" → Output: "09:16:37"

- Divisão da coluna "Last Updated Dt" em componentes individuais para sucessiva combinação depois da normalização de valores.

Last Updated Dt String
9/18/2025 2:20:00 PM
9/12/2025 12:47:00 PM
9/23/2025 10:33:00 AM

→

Last Upd... String	Last Upd... String	Last Upd... String
9/18/2025	2:20:00	PM
9/12/2025	12:47:00	PM
9/23/2025	10:33:00	AM

Last Upd... String	Last Upd... String	Last Upd... String
9/18/2025	2:20:00	PM
9/12/2025	12:47:00	PM
9/23/2025	10:33:00	AM

→

Last Updated Date Date&time (Local)
2025-09-18T14:20
2025-09-12T12:47
2025-09-23T10:33

Resultado: Input "9/18/2025 2:20:00 PM"

↓ Cell Splitter (por espaço)

Date "9/18/2025" | Time "2:20:00" | Periodo "PM"

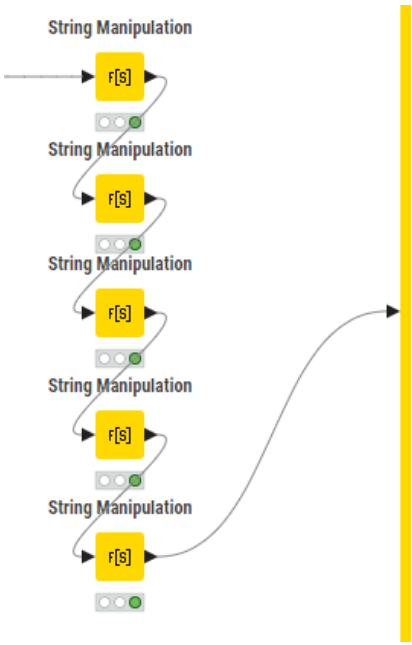
↓ String Manipulation (padding)

Time "02:20:00"

↓ Combinação Date&Time

Output "2025-09-18T14:20"

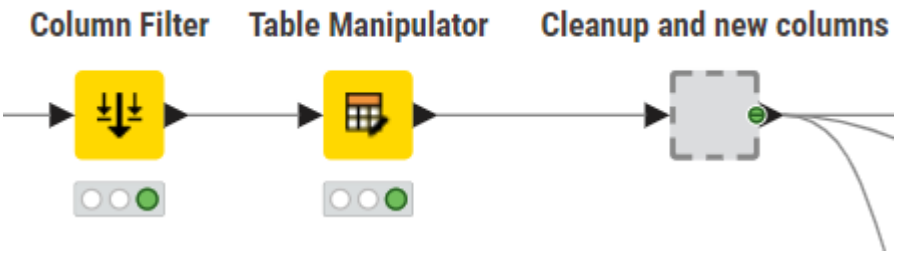
- Conversão de diferentes fusos horários (EDT, CDT, PDF... para UTC)



Código Exemplo: regexReplace($Time Zone$,"EDT","UTC-4")

Resultado : Input: "EDT" → Output: "UTC-4"

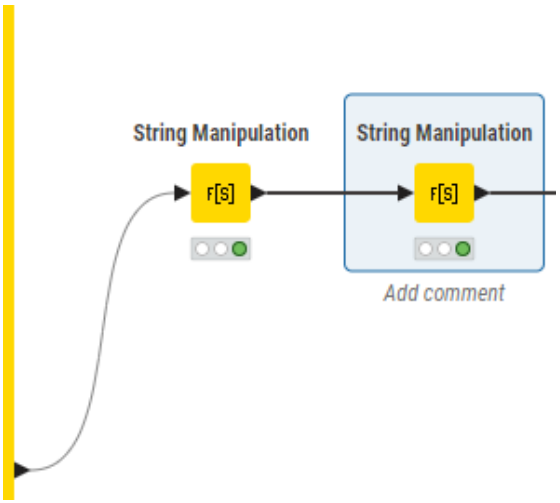
4.2.3 Limpeza de colunas



- Filtragem de colunas irrelevantes usando Column Filter e Table Manipulator

4.2.4 Enriquecimento com expressões regulares

- Criação da coluna "Event Category" através de expressões regulares

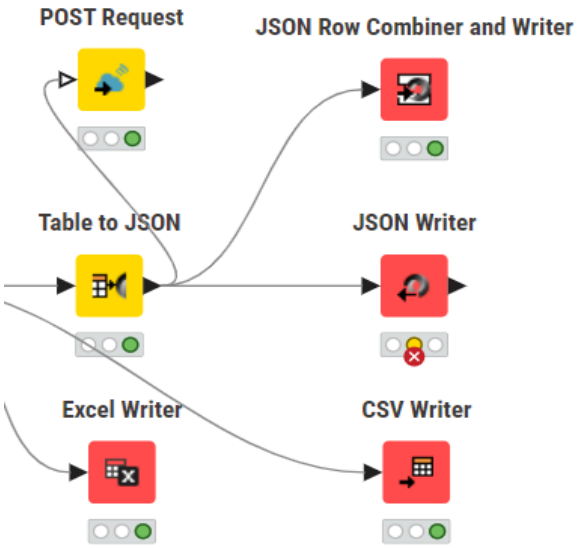


Código Exemplo: regexReplace($Event Text$, ".*?(MEDICAL EVENT |EQUIPMENT FAILURE|DAMAGED|STUCK OPEN|DEVIATION|LOST |STOLEN|LEAKING| SCRAM|TRIP).*", "$1")

Resultado: Input: "AGREEMENT STATE REPORT - MEDICAL EVENT"
Output: "MEDICAL EVENT"

Input: "PART 21 INTERIM REPORT OF DEVIATION"
Output: "DEVIATION"

2. **Conversão, Serialização e Exportação**



- Conversão dos dados para o formato JSON
- Exportação para múltiplos formatos (CSV, Excel, JSON)
- Integração com API REST através do nó POST Request

4.3 Fluxo de Transformação Detalhado

Coluna	Problema	Solução	Resultado
Notification Time	h:mm:ss	Padding left	hh:mm:ss
Time Zone	EDT/CDT/PDT	Padronização	UTC-4/UTC-5/UTC-7
County Name	Vazio	"Unknown"	Dado consistente
Event Text	Categoria implícita	Regex extraction	Categoria explícita

5. Jobs e Integração

5.1 Arquitetura de Integração

Componentes da Arquitetura:

- **KNIME Analytics Platform**: Orquestração principal do workflow ETL
- **Flask API**: Middleware para comunicação e transformação de dados
- **Node-RED**: Plataforma de dashboard e visualização em tempo real
- **Sistema de Logs**: Monitorização e auditoria de todo o processo

5.2 Flask API Gateway

5.2.1 Configuração do Flask

Aqui estãpo estabelecidas as configurações principais do Flask:

NODE_RED_URL = 'http://localhost:1880'

O url onde está o node red.

NODE_RED_ENDPOINT = '/api/knime-data'

O endpoint do url a que o Flask vai enviar os dados

NODE_RED_TIMEOUT = 30 # segundos

O tempo até o node red dar timeout.

*MAX_CONTENT_LENGTH = 16 * 1024 * 1024 # 16MB*

O tamanho máximo que pode ser enviar pelo node red

5.2.2 Endpoints

Endpoint	Método	Função	Códigos de Resposta
/api/forward-to-nodered	POST	Recebe dados do KNIME e encaminha para Node-RED	200, 400, 502, 503, 504, 500
/api/batch-process	POST	Processamento de tipo batch com validação	200, 400, 502, 500
/api/health	GET	Verificação de saúde do sistema	200
/api/status	GET	Status e estatísticas da API	Status e estatísticas da API

5.2.3 Fluxo de processamento da api

1. Receção do payload JSON do KNIME
2. Validação da estrutura e conteúdo dos dados
3. Transformação do payload para formato Node-RED
4. Encaminhamento para Node-RED com timeout
5. Gestão de respostas e erros
6. Geração de logs detalhados

5.3 Node-RED Dashboard

5.4.1 Arquitetura do Dashboard

O flow do node-red segue uma estrotura relativamente simples:

HTTP INPUT → Processamento dos dados → Componentes do dashboard

↓ ↓ ↓

Recebe dados do API Flask

Transformação e cálculo de KPIs

6 Visualizações Simultâneas

5.4.2 Componentes do Dashboard Implementados

1. KPI Cards (Dashboard Principal)

- **Total Events**: Contagem total de eventos processados
- **Critical Events**: Eventos com severidade "High"
- **Medical Events**: Eventos médicos específicos
- **High Risk**: Eventos com risk score ≥ 70

3. All Events Table

- Tabela pesquisável e ordenável
- Filtros por severidade e categoria
- Sistema de cores por prioridade

4. Risk Assessment

- Visão geral de risco consolidada
- Distribuição por severidade (Low/Medium/High)
- Métricas de eventos de alto risco

5. Geographic Distribution

- Distribuição por estados norte-americanos
- Análise regional (Regions 1-4)
- Heat map visual com cores por intensidade

6. Export & Actions Panel

- Exportação para CSV e JSON
- Botão de refresh manual e automático
- Estatísticas detalhadas

5.4.3 Algoritmos de Processamento em Node-RED

- Calculo da severidade:

```
function extractSeverity(eventText, category) {
  // Medical events são sempre alta severidade
  if (category === 'MEDICAL EVENT') return 'High';

  const text = eventText.toLowerCase();

  // Eventos de equipamento e segurança
  if (text.includes('medical event') || text.includes('patient harm') ||
    text.includes('exposure') || text.includes('radiation release')) return 'High';

  if (text.includes('equipment failure') || text.includes('leak') ||
    text.includes('damaged') || text.includes('stuck open')) return 'Medium';

  return 'Low';
}
```

- Cálculo da Risk Score:

```
function calculateRiskScore(category, eventText) {
  let score = 50; // Base score

  // Pontuação baseada na categoria
  if (category === 'MEDICAL EVENT') score += 40;
  if (category === 'EQUIPMENT FAILURE') score += 30;
  if (category === 'DAMAGED' || category === 'LEAKING') score += 25;

  // Pontuação baseada no conteúdo
  const text = eventText.toLowerCase();
  if (text.includes('medical event')) score += 20;
  if (text.includes('patient')) score += 15;
  if (text.includes('exposure') || text.includes('radiation')) score += 25;

  return Math.min(100, Math.max(0, score));
}
```

5.4 Processamento de Dados em Node-RED

- Funções JavaScript personalizadas para transformação de dados
- Cálculo de KPIs em tempo real
- Sistema de severidade automática baseado em:
    - Medical Events → High Severity
    - Equipment Failure → Medium Severity
    - Outros → Low Severity
- Agregações estatísticas por categoria, estado e região

6. Demonstração

Criei um código QR que redireciona para um video da explicação do projeto em desenvolvimento.

Conteúdos do Vídeo:

- Configuração do ambiente KNIME
- Execução do workflow completo
- Demonstração das transformações
- Visualização dos logs do API Flask
- Demonstração do API Flask
- Visualização dos resultados no dashboard

7. Conclusão e Trabalhos Futuros

7.1 Conclusões

O projeto demonstrou com sucesso a aplicação de processos ETL em cenários complexos de integração de dados. Através da plataforma KNIME, foi possível:

- Processar múltiplos formatos de dados com eficiência
- Implementar transformações com expressões regulares
- Garantir a qualidade e consistência dos dados
- Gerar outputs estruturados para integração com o node-red

7.2 Conquistas Técnicas

- **Processamento Multi-formato**: CSV, TXT pipe-delimited, JSON
- **Expressões Regulares**: Categorização automática de eventos
- **Serialização JSON**: Estrutura complexa com metadados
- **API REST Completa**: Autenticação, validação, logging
- **Dashboard em Tempo Real**: Dashbpard com varios elementos para visualização dos dados
- **Integração Multi-plataforma**: KNIME + Flask API + Node-RED

7.3 Trabalhos Futuros

- **Integração com Bases de Dados:** PostgreSQL/MongoDB
- **Integração de mais tabelas:** Mais dados para serem analisados
- **Mais filtros:** Mais opções para visualizar os dados.
- **Mais validações:** Mais operações regex para manter a qualidade dos dados.
- **Machine Learning:** Deteção de anomalias e previsão de eventos
- **Mobile App:** Versão responsive para dispositivos móveis
- **Alertas por Email/SMS:** Notificações automáticas

8. Referências Bibliográficas

8.1 Documentação Técnica

- KNIME Analytics Platform Documentation
- Node-RED Official Documentation

  JSON Schema Specification, JSON.org
- Flask documentation

8.2 Referências Web

- https://nodered.org/ - Plataforma Node-RED
- https://www.knime.com/ - KNIME Analytics Platform
- https://flask.palletsprojects.com/en/stable/ - Flask API
- https://forum.knime.com/t/converting-hss-to-hhss/90183/6 -Ajuda no knime formus