



**INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR
DE TECNOLOGIA**

TRABALHO PRÁTICO

Integração de Sistemas de Informação

Professor Luís Ferreira

António José Martins Ferreira

Número Aluno: 9657
a9657@alunos.ipca.pt

Conteúdo

Índice de figuras.....	3
Lista de Abreviaturas	4
Proposta do Sistema	6
1. Resumo	6
2. Contextualização	7
3. Problema e Objetivos	8
4. Arquitetura Técnica.....	10
5. Pré-requisitos e Instalação	13
6. Estrutura de Pastas	15
7. Modelo de Dados	17
8. Transformação: stg_moradores.....	19
9. Transformação: stg_pagamentos	22
10. Transformação: stg_meteo_export.....	25
11. Job: job_Condominio	28
12. Integração com API Open-Meteo	31
13. Dashboards no Grafana	34
Requisitos do Enunciado — Matriz de Evidências	37
Execução Rápida - Checklist	40
Resolução de Problemas.....	43
Boas Práticas e Padrões	45
Conclusão	47
Referências	49

Índice de figuras

Figura 1 - Transformação stg_moradores	21
Figura 2 – Transformação stg_pagamentos	24
Figura 3– Transformação stg_meteo_export.....	27
Figura 4 - Job principal job_Condominio	30
Figura 5 – Dashboard no Grafana	36
Figura 6 - QR Code.....	42

Lista de Abreviaturas

Sigla / Abreviatura	Significado
ETL	<i>Extract, Transform, Load</i> – Processo de Extração, Transformação e Carga de dados.
PDI	<i>Pentaho Data Integration</i> – Ferramenta usada para construir os processos ETL (Spoon e Kitchen).
KTR	Extensão de ficheiro das <i>Transformações</i> no Pentaho PDI.
KJB	Extensão de ficheiro dos <i>Jobs</i> (processos de orquestração) no Pentaho PDI.
API	<i>Application Programming Interface</i> – Interface de programação que permite a comunicação entre sistemas.
CSV	<i>Comma-Separated Values</i> – Formato de ficheiro de texto para armazenamento de dados tabulares.
XML	<i>eXtensible Markup Language</i> – Linguagem de marcação usada para estruturar dados.
JSON	<i>JavaScript Object Notation</i> – Formato leve para intercâmbio de dados, usado em APIs.
HTTP	<i>Hypertext Transfer Protocol</i> – Protocolo de comunicação usado para aceder à API Open-Meteo.
SQL	<i>Structured Query Language</i> – Linguagem utilizada para manipular bases de dados relacionais.
BD	Base de Dados.
ER	Expressões Regulares (<i>Regular Expressions</i>) – Padrões usados para validação e substituição de texto.
SK	<i>Surrogate Key</i> – Chave substituta usada como identificador único em tabelas dimensionais.
UPSERT	Combinação de <i>Update</i> + <i>Insert</i> – operação que insere novos registos ou atualiza existentes.

FTP	<i>File Transfer Protocol</i> – Protocolo usado para transferência de ficheiros (simulado no job).
API Open-Meteo	Serviço externo usado para recolher dados meteorológicos históricos e atuais.
Grafana	Plataforma de visualização e monitorização de dados, utilizada para criar dashboards.
UC	Unidade Curricular.
ISI	Integração de Sistemas de Informação – Unidade Curricular do curso.
Log	Registo de execução (mensagens, erros e resultados de cada passo).
Job	Processo de controlo e orquestração de transformações no Pentaho.
Transformação	Conjunto de passos no Pentaho que executam a lógica ETL.

Proposta do Sistema

1. Resumo

Este relatório documenta, passo a passo, a construção de um sistema ETL para o contexto de gestão de condomínios.

O objetivo é que qualquer pessoa — mesmo sem experiência prévia — consiga compreender o porquê de cada etapa e reproduzir o projeto.

O trabalho cumpre os requisitos do enunciado da UC de ISI (Integração de Sistemas de Informação), incluindo:

- utilização de expressões regulares;
- importação/exportação (XML);
- jobs de orquestração;
- acesso a API remota;
- operações sobre bases de dados;
- geração de logs e visualização dos resultados no Grafana.

2. Contextualização

O que é ETL?

ETL é o acrónimo de **Extract, Transform, Load**, ou em português, **Extrair, Transformar e Carregar**.

Trata-se de um processo fundamental na integração de dados, utilizado para reunir informação proveniente de diferentes fontes, tratá-la de forma consistente e disponibilizá-la para análise e apoio à decisão.

Em termos práticos:

- **Extrair (Extract)** consiste em recolher dados das suas origens, como ficheiros, bases de dados ou serviços web (APIs);
- **Transformar (Transform)** envolve limpar, normalizar, converter e calcular novos campos, garantindo a qualidade e coerência da informação;
- **Carregar (Load)** corresponde à gravação dos dados transformados numa base de dados ou sistema de destino, onde passam a estar prontos para consulta e análise.

No contexto deste projeto, o processo ETL integra três fontes principais:

1. Ficheiros **CSV** com informação dos moradores;
2. Ficheiros **XLSX** com registos de pagamentos;
3. Dados **meteorológicos** obtidos através da **API Open-Meteo**, que fornecem o estado do tempo na data de cada registo.

Após a integração e tratamento, todos os dados são armazenados numa base de dados central e podem ser **explorados visualmente num painel interativo do Grafana**, permitindo análises cruzadas entre moradores, pagamentos e condições meteorológicas.

3. Problema e Objetivos

Problema

Atualmente, a informação relativa aos **moradores e pagamentos** encontra-se distribuída em diferentes ficheiros CSV, sem qualquer ligação entre si.

Esta fragmentação dificulta a análise global dos dados e impede a obtenção de indicadores relevantes para a gestão dos condomínios.

Além disso, a ausência de contexto externo — como as **condições meteorológicas** — limita a capacidade de identificar padrões ou correlações entre eventos (por exemplo, atrasos nos pagamentos em períodos de maior precipitação).

Desta forma, surge a necessidade de **integrar, limpar e enriquecer** a informação proveniente de fontes heterogéneas, garantindo consistência e qualidade nos dados antes de qualquer análise ou visualização.

Objetivos

O projeto tem como principal propósito **desenvolver um sistema ETL completo**, que consolide e transforme os dados dispersos, permitindo a sua exploração analítica e visual.

Os objetivos específicos são:

- **Centralizar** os dados de moradores e pagamentos num repositório único e estruturado;
- **Validar e normalizar** os dados, assegurando a coerência dos formatos (ex.: datas, valores monetários, NIFs e telefones);
- **Enriquecer** os registos com informação meteorológica diária proveniente da **API Open-Meteo**, associando o estado do tempo a cada data de registo;
- **Exportar os resultados** em formatos interoperáveis (XML e JSON) para facilitar o intercâmbio com outros sistemas;

- **Disponibilizar visualizações** em painéis **Grafana**, permitindo acompanhar métricas como pagamentos por mês, moradores ativos e influência do clima nas operações.

Com esta abordagem, o projeto garante uma **visão integrada e atualizada** do ecossistema de dados dos condomínios, promovendo uma gestão mais informada, eficiente e orientada por dados (*data-driven management*).

4. Arquitetura Técnica

A arquitetura técnica do projeto foi concebida para garantir **modularidade**, **transparência e facilidade de manutenção** em todas as fases do processo ETL — desde a extração até à visualização dos dados.

O fluxo global segue a lógica:

**CSV → Pentaho PDI (Transformações) → Base de Dados SQL → Exportação XML
→ Painel Grafana + Enriquecimento via API Open-Meteo**

Esta estrutura permite integrar dados provenientes de diferentes origens, tratá-los de forma controlada e disponibilizá-los para análise em tempo real.

Fluxo de Alto Nível

1. CSV (Origem dos Dados)

Os ficheiros de entrada — *moradores.csv* e *pagamentos.csv* — contêm os dados brutos extraídos do sistema de gestão dos condomínios. Estes ficheiros são armazenados na pasta *data/staging/* e constituem a base da camada de *staging* do processo ETL.

2. Pentaho PDI CE (Transformações e Jobs)

A ferramenta **Pentaho Data Integration (PDI CE)**, versão **10.2.0.0-222**, é o motor principal do pipeline.

Através das suas transformações (*.ktr*) e *jobs* (*.kjb*), os dados são:

- Extraídos dos ficheiros CSV;
- Limpos e normalizados com *String Operations* e *Replace in String*;
- Enriquecidos com dados meteorológicos provenientes da API;
- Carregados para a base de dados final.

3. Base de Dados SQL (Camada de Integração e Armazenamento)

A informação tratada é armazenada numa **base de dados relacional** (PostgreSQL, MySQL ou SQLite), que serve de repositório central.

Esta camada garante integridade referencial entre entidades como *moradores, pagamentos e condições meteorológicas*, sendo também a fonte de dados para o Grafana.

4. Exportação XML (Interoperabilidade)

Após o carregamento, os dados consolidados podem ser exportados automaticamente para um ficheiro XML localizado em *data/out/xml/*. Este formato permite partilhar ou importar os resultados para outros sistemas, cumprindo o requisito de **interoperabilidade e portabilidade de dados**.

5. API Open-Meteo (Enriquecimento de Dados)

A integração com a **API Open-Meteo** fornece, de forma automática, informações meteorológicas correspondentes às datas e localizações dos registos.

Estes dados são processados em tempo real e incorporados na base de dados, permitindo análises mais ricas e contextuais.

6. Painel Grafana (Visualização e Monitorização)

O **Grafana** é a camada de visualização do sistema, conectando-se diretamente à base de dados para gerar dashboards interativos.

Estes painéis permitem observar indicadores como o número de moradores, pagamentos efetuados por período, estado das transações e até correlações com o estado do tempo.

Resumo dos Componentes Técnicos

Componente	Descrição	Localização / Versão
Pentaho PDI CE (Spoon/Kitchen)	Ferramenta ETL utilizada para as transformações e orquestração dos jobs.	v10.2.0.0-222
Base de Dados SQL	Repositório relacional que armazena os dados tratados.	PostgreSQL

Ficheiros CSV	Fonte inicial de dados (moradores e pagamentos).	data/staging/
Exportação XML	Saída do pipeline para integração com outros sistemas.	data/out/xml/
API Open-Meteo	Fonte externa para enriquecimento meteorológico.	Endpoint REST (JSON)
Grafana	Ferramenta de visualização e análise de dados.	Web UI – Dashboards

5. Pré-requisitos e Instalação

Antes da execução do pipeline ETL, é necessário preparar o ambiente de trabalho e garantir que todos os componentes estão corretamente configurados.

Seguem-se os passos recomendados:

1. Instalar o Pentaho PDI CE

Descarregar e instalar o **Pentaho Data Integration (Community Edition)**, versão **10.2.0.0-222**, no diretório: C:\ISI\pdi-ce-10.2.0.0-222\
Esta ferramenta será utilizada para criar e executar todas as transformações (.ktr) e jobs (.kjb).

2. Instalar o Java (JDK ou JRE)

O Pentaho PDI requer o **Java Runtime Environment (JRE)** ou o **Java Development Kit (JDK)** compatível (versão 8 ou superior).
Após a instalação, verificar a variável de ambiente JAVA_HOME e testar com o comando `java -version`.

3. Criar a Base de Dados

Criar um **schema** e as respetivas **tabelas de destino** na base de dados relacional escolhida (PostgreSQL, MySQL ou SQLite).
Configurar também o **utilizador, palavra-passe e permissões de leitura/escrita**, assegurando o acesso a partir do Pentaho através das *Database Connections*.

4. Instalar e configurar o Grafana

Instalar o **Grafana** (versão Community) e adicionar uma nova *Data Source* apontando para a base de dados criada.
Esta ligação permitirá a visualização dos dados tratados em dashboards interativos.

5. Garantir acesso à Internet

É necessário **acesso à Internet** para que o Pentaho consiga comunicar com a **API Open-Meteo**, responsável pelo enriquecimento dos dados com as condições meteorológicas.

Em caso de rede corporativa, poderá ser necessário configurar o *proxy* no sistema.

6. Preparar os ficheiros de entrada

Colocar os ficheiros de origem — *moradores.csv* e *pagamentos.csv* — na pasta:

data/staging/

Estes ficheiros serão utilizados como fonte de dados para as transformações iniciais do processo ETL.

7. Verificar permissões de pastas e logs

Garantir que o utilizador tem **permissões de leitura e escrita** nas pastas do projeto, especialmente em:

- *data/staging/* (entrada de dados)
- *data/out/xml/* (exportação de resultados)
- *data/out/logs/* (armazenamento dos ficheiros de log gerados pelas transformações)

6. Estrutura de Pastas

A organização das pastas foi planeada para garantir **clareza, modularidade e facilidade de manutenção** do projeto.

Cada diretório possui uma função específica dentro do processo ETL, desde a receção dos dados brutos até à exportação e documentação final.

A estrutura recomendada é a seguinte:

```
C:/SAD/pdi-ce-10.2.0.0-222/ISI/  
├── data/  
│   ├── staging/  
│   │   ├── moradores.csv  
│   │   └── pagamentos.xlsx  
│   └── output/  
│       ├── xml/  
│       │   └── meteo_export.xml  
│       └── logs/  
├── kettle/  
│   ├── transforms/  
│   │   ├── stg_moradores.ktr  
│   │   ├── stg_pagamentos.ktr  
│   │   └── stg_meteo_export.ktr  
│   └── jobs/  
│       └── job_Condominio.kjb  
└── docs/  
    └── Relatorio.docx
```

Descrição dos Diretórios

- **data/**

Contém todos os dados de entrada e saída do sistema. É a principal pasta operacional do projeto.

- **staging/**

Diretório destinado aos **ficheiros brutos de entrada** (moradores.csv e pagamentos.csv).

Representa a camada de *staging*, onde os dados são inicialmente carregados antes de qualquer transformação.

- **out/**

Armazena os resultados do processo ETL, incluindo:

- **xml/** – ficheiros exportados no formato XML, como meteo_export.xml;
- **logs/** – ficheiros de log gerados automaticamente após a execução das transformações e do job principal.
- **kettle/**

Pasta que agrupa todos os ficheiros do **Pentaho PDI**, organizados por tipo.

 - **transforms/**

Contém as **transformações individuais (.ktr)**, responsáveis pelas operações de limpeza, validação e carga de dados:

 - stg_moradores.ktr → processamento de dados dos moradores;
 - stg_pagamentos.ktr → normalização dos pagamentos;
 - stg_meteo_export.ktr → exportação e enriquecimento meteorológico.
 - **jobs/**

Contém os **ficheiros de orquestração (.kjb)**, que controlam a execução sequencial das transformações:

 - job_Condominio.kjb → executa todo o pipeline ETL de forma automatizada.
- **docs/**

Diretório reservado para a documentação do projeto, incluindo o relatório final:

 - Relatorio.docx – documento técnico e explicativo do sistema ETL desenvolvido.

Esta estrutura modular facilita a **reutilização**, o **controlo de versões** e a **replicação do ambiente** noutros sistemas ou máquinas, assegurando que qualquer utilizador consiga executar o projeto de forma autónoma e ordenada.

7. Modelo de Dados

O modelo de dados do projeto segue uma abordagem **dimensionada** (*star schema simplificado*), adequada à análise de indicadores relacionados com moradores, pagamentos e condições meteorológicas.

Esta estrutura permite uma navegação intuitiva e otimizada em consultas analíticas, mantendo a integridade e a coerência entre as entidades.

Tabelas Principais

- **dim_morador** (*Dimensão Morador*)

Armazena a informação descritiva de cada morador.

Contém atributos de identificação e contacto, sendo a base para análise de pagamentos por residente.

Campos principais:

id_morador, nome, nif, telefone, email, morada, condominio.

- **fact_pagamento** (*Tabela de Factos – Pagamentos*)

Regista todos os eventos de pagamento associados aos moradores.

Inclui dados financeiros e temporais, permitindo o cálculo de métricas como total pago, pagamentos em atraso ou número de transações.

Campos principais:

id_pagamento, id_morador, data_pagamento, valor, estado.

- **dim_meteo_dia** (*Dimensão Meteorológica*)

Contém os dados meteorológicos obtidos através da **API Open-Meteo**, registando as condições do tempo por dia e localização.

Esta tabela é utilizada para enriquecer as análises temporais e identificar possíveis correlações entre o clima e o comportamento dos moradores.

Campos principais:

data, localizacao, condicao, temperatura, precipitacao.

Relações Entre Tabelas

- **fact_pagamento.id_morador → dim_morador.id_morador**

Relação de chave estrangeira que liga cada pagamento ao respetivo morador, permitindo análises de faturação por residente ou condomínio.

- **fact_pagamento.data_pagamento ↔ dim_meteo_dia.data**

Relação lógica (por data) utilizada para cruzar informações de pagamentos com as condições meteorológicas do mesmo dia, possibilitando análises contextuais (ex.: impacto da chuva no cumprimento de pagamentos).

Síntese do Modelo

Tipo de Tabela	Nome	Função Principal	Tipo de Dados
Dimensão	dim_morador	Descrição e identificação de cada morador	Textuais e numéricos
Fato	fact_pagamento	Registo de eventos de pagamento	Numéricos e temporais
Dimensão	dim_meteo_dia	Dados diários sobre o estado do tempo	Numéricos e categóricos

Este modelo garante **simplicidade, escalabilidade e integridade referencial**, suportando tanto o carregamento de dados via ETL como a sua posterior exploração analítica no **Grafana**.

8. Transformação: stg_moradores

Objetivo

A transformação **stg_moradores.ktr** tem como finalidade **carregar, validar e normalizar** os dados dos moradores a partir do ficheiro CSV de origem, garantindo que apenas registos consistentes e limpos são inseridos na tabela **dim_morador** da base de dados.

Esta fase representa o ponto de entrada do pipeline ETL, assegurando a integridade e a qualidade dos dados antes da sua utilização em etapas posteriores.

Passos no Pentaho Spoon

1. CSV File Input

Leitura do ficheiro data/staging/moradores.csv, com o cabeçalho ativo (*Header row present*).

Nesta etapa, o Pentaho identifica automaticamente os nomes e tipos de campos através da função **Get Fields**, mapeando as colunas do ficheiro para o fluxo interno da transformação.

2. String Operations

Aplicação de operações de limpeza textual a todos os campos do tipo String, nomeadamente:

- **Trim** (remoção de espaços em branco à esquerda e à direita);
- Conversão opcional de texto para *Proper Case* ou *Upper Case* (uniformização dos nomes e estados).

Esta etapa visa normalizar os formatos de escrita e evitar inconsistências de capitalização.

3. Replace in String

Utilização de **expressões regulares (Regex)** para eliminar caracteres inválidos nos campos numéricos, como NIF e Telefone.

Padrão aplicado:

4. [^0-9]

Esta expressão remove qualquer caractere que não seja um dígito, assegurando que os valores ficam num formato puramente numérico.

5. **Select Values**

Seleção dos campos relevantes e renomeação conforme o modelo de dados da base de dados (id → id_morador, nome → nome_morador, etc.).

Nesta fase são também definidos os **tipos de dados** e **comprimentos máximos** de cada campo, assegurando compatibilidade com a estrutura da tabela de destino.

6. **Table Output**

Escrita dos dados limpos na tabela **dim_morador** da base de dados configurada na ligação do Pentaho.

O modo de escrita utilizado é **Insert**, garantindo que apenas novos registos são inseridos, sem duplicar entradas existentes.

Caso o pipeline seja executado repetidamente, recomenda-se a utilização de lógica **UPSERT** (Update or Insert).

7. **Logging**

Ativação do registo de logs de execução, armazenados em data/out/logs/.

O ficheiro de log contém informações detalhadas sobre a execução, como número de registos lidos (I), escritos (W), atualizados (U) e eventuais erros (E).

Este mecanismo permite rastrear a execução e diagnosticar eventuais falhas.

Validações e Controlo de Qualidade

- **Unique Rows:**

Aplicação de uma verificação de unicidade sobre o campo id_morador, prevenindo a inserção de duplicados.

- **Filter Rows:**

Exclusão de registos com NIF inválido, nulo ou com comprimento incorreto (diferente de 9 dígitos), assegurando conformidade com o padrão português.

Estas validações reforçam a consistência dos dados e evitam erros futuros nas fases de integração e análise.

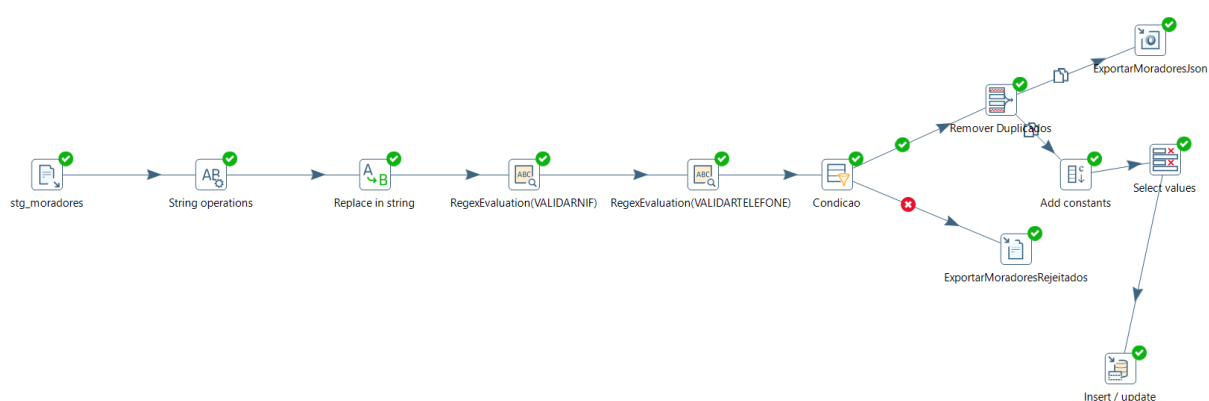


Figura 1 - Transformação stg_moradores

9. Transformação: stg_pagamentos

Objetivo

A transformação **stg_pagamentos.ktr** tem como finalidade **carregar, validar e normalizar** os dados de pagamentos provenientes do ficheiro CSV, garantindo a coerência de formatos e a integridade referencial com os moradores registados na base de dados.

Esta transformação é responsável por criar a tabela de factos **fact_pagamento**, elemento central para as análises financeiras no contexto da gestão de condomínios.

Passos na Transformação

1. CSV File Input

Leitura do ficheiro data/staging/pagamentos.csv, com o cabeçalho ativo (*Header row present*).

Esta etapa define o ponto de entrada dos dados brutos no processo ETL, identificando automaticamente os campos disponíveis e os seus respetivos tipos.

2. Replace in String (Valor)

Aplicação de uma limpeza no campo **valor** para remover símbolos ou caracteres inválidos (ex.: €, espaços, vírgulas desnecessárias).

Utiliza-se uma **expressão regular** que mantém apenas dígitos, pontos e vírgulas válidas.

Exemplo de padrão:

3. [^0-9,-]

Em seguida, a vírgula decimal é convertida em ponto (1.234,56 → 1234.56) para permitir o correto reconhecimento numérico.

4. Regex Evaluation (Datas)

Conversão do formato de data de **dd-MM-yyyy** para **yyyy-MM-dd**,

compatível com os padrões SQL.

A transformação utiliza o operador **Regex Evaluation** com o padrão:

5. `^(\d{2})-(\d{2})-(\d{4})$`

E substituição:

`$3-$2-$1`

Esta etapa é essencial para garantir a correta ordenação e filtragem temporal no repositório final.

6. **Select Values**

Seleção e mapeamento dos campos finais, de acordo com o modelo de dados da base de dados.

Exemplo:

- id → id_pagamento
- morador_id → id_morador
- data → data_pagamento
- valor → valor
- estado → estado

Também são aplicados tipos e comprimentos adequados a cada campo (inteiro, decimal, string, data).

7. **Stream Lookup (Cruzamento com dim_morador)**

Ligação dinâmica entre os pagamentos e os moradores através de um **Stream Lookup**.

Este passo cruza o campo id_morador do fluxo de pagamentos com o campo homónimo da tabela **dim_morador**, assegurando integridade referencial e filtrando pagamentos associados apenas a moradores válidos.

8. **Table Output (fact_pagamento)**

Gravação dos dados processados na tabela **fact_pagamento** da base de

dados.

Esta etapa conclui o processo de carga (*Load*), registando os pagamentos normalizados e validados.

O modo de escrita utilizado é **Insert**, mas pode ser configurado para **Update/Insert (Upsert)**, caso se pretenda atualização incremental.

9. Logging Ativo

Todos os eventos da transformação são registados em ficheiros de log na pasta `data/out/logs/`, incluindo número de linhas lidas (I), escritas (W), atualizadas (U) e eventuais erros (E).

Esta prática assegura rastreabilidade total e facilita a deteção e correção de problemas durante a execução.

Validações Adicionais

- **Controlo de valores nulos ou negativos** no campo valor;
- **Verificação de datas inválidas ou fora do intervalo esperado;**
- **Deteção de pagamentos duplicados**, com base em `id_pagamento` + `data_pagamento`.

Estas medidas garantem que apenas dados limpos e coerentes são inseridos na base de dados final.

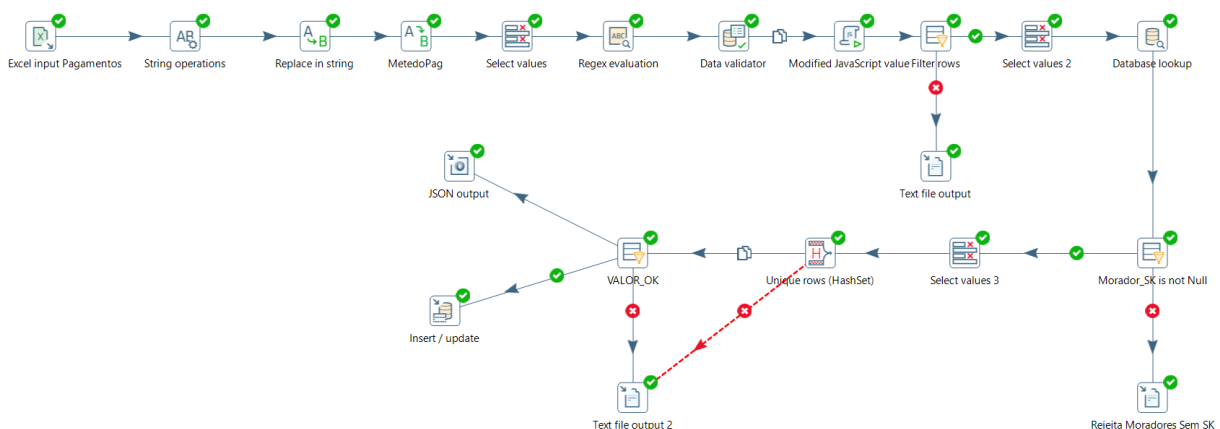


Figura 2 – Transformação *stg_pagamentos*

10. Transformação: stg_meteo_export

Objetivo

A transformação **stg_meteo_export.ktr** tem como finalidade **gerar um ficheiro XML consolidado com dados meteorológicos**, permitindo o enriquecimento analítico dos restantes conjuntos de dados (moradores e pagamentos).

Esta transformação foi desenvolvida para suportar dois cenários de funcionamento:

- **Cenário A:** extração direta dos dados a partir da **API Open-Meteo** (tempo real ou histórico);
- **Cenário B:** exportação de dados já existentes na base de dados (dim_meteo_dia) para formato XML, garantindo interoperabilidade e portabilidade.

Cenário A – Integração com a API Open-Meteo

1. Table Input (Datas/Localizações)

Leitura de um conjunto de datas e localizações a partir da base de dados (por exemplo, uma lista de cidades ou coordenadas geográficas associadas a cada condomínio).

Estes dados servem como parâmetros dinâmicos para as chamadas à API.

2. REST Client (Chamada à API)

Configuração de uma ligação HTTP para o serviço **Open-Meteo**, que disponibiliza dados meteorológicos em formato JSON.

O endpoint é definido com parâmetros dinâmicos, como latitude, longitude, start_date e end_date.

Exemplo de URL utilizada:

3. https://api.open-

meteo.com/v1/forecast?latitude=\${lat}&longitude=\${lon}&daily=temperature_2m_max,precipitation_sum&start_date=\${data}&end_date=\${data}&timezone=auto

Este passo é responsável por recolher os dados brutos diretamente da API.

4. JSON Input (Mapeamento de Campos)

Conversão da resposta JSON devolvida pela API num formato tabular interno do Pentaho.

Nesta etapa, são extraídos e renomeados os campos relevantes, tais como:

- temperature_2m_max → Temperatura Máxima
- precipitation_sum → Precipitação Total
- weathercode → Código de Condição Meteorológica

Estes valores são posteriormente enriquecidos com descrições textuais (ex.: “Céu limpo”, “Chuva moderada”).

5. XML Output (Geração do Ficheiro)

Escrita dos dados processados no ficheiro XML localizado em:

6. data/out/xml/meteo_export.xml

O ficheiro é gerado com codificação **UTF-8** e estrutura hierárquica definida, incluindo os nós principais <meteorologia>, <dia>, <localizacao> e <dados>. Este formato garante compatibilidade com sistemas externos e facilidade de importação em ferramentas analíticas.

Cenário B – Exportação Direta a partir da Base de Dados

1. Table Input (dim_meteo_dia)

Leitura dos dados previamente armazenados na tabela **dim_meteo_dia**, contendo as informações meteorológicas resultantes de execuções anteriores da API.

2. XML Output (Exportação Final)

Geração direta do ficheiro XML final a partir dos registos existentes.

O ficheiro é exportado com codificação **UTF-8**, garantindo a preservação correta de caracteres especiais e acentuação.

Este cenário é útil em ambientes sem ligação à Internet ou para reexportações periódicas.

Boas Práticas Implementadas

- Validação do formato JSON antes do mapeamento (para evitar falhas em respostas nulas ou corrompidas).
- Criação de **logs detalhados** em data/out/logs/ com contagem de linhas lidas, escritas e eventuais erros.
- Verificação de permissões de escrita na pasta de saída antes da geração do ficheiro.
- Estrutura XML consistente e reutilizável, preparada para futuras extensões de dados (ex.: velocidade do vento, humidade, etc.).

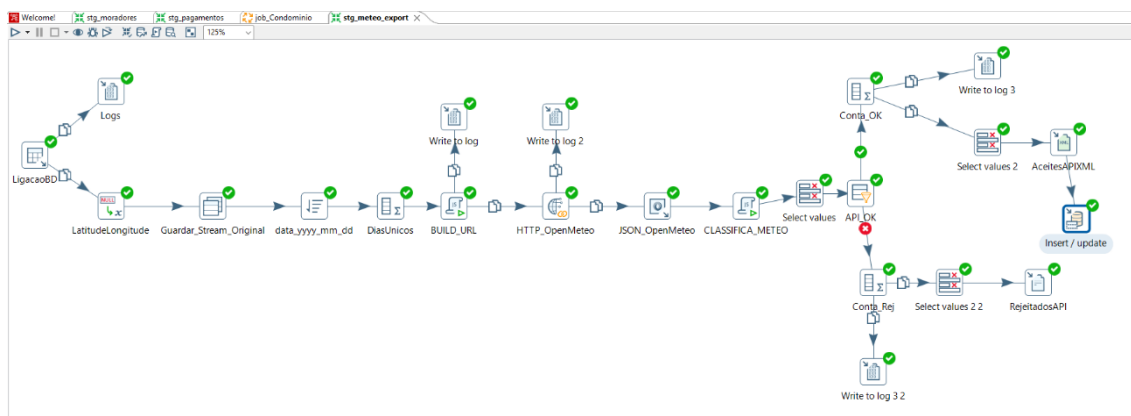


Figura 3– Transformação stg_meteo_export

11. Job: job_Condominio

Objetivo

O job principal **job_Condominio.kjb** tem como propósito **orquestrar e automatizar a execução sequencial das transformações** que compõem o pipeline ETL do projeto — **stg_moradores**, **stg_pagamentos** e **stg_meteo_export**. Esta orquestração garante que todas as etapas são executadas na ordem correta, com controlo de dependências, verificação de erros e registo completo de logs, assegurando a integridade do processo.

Passos na Execução do Job

1. Start → Set Variables → **stg_moradores** → **stg_pagamentos** → **stg_meteo_export**

O job inicia com o componente **Start**, seguido de um passo opcional **Set Variables** onde são definidas variáveis globais (ex.: diretório, nomes de ficheiros ou credenciais da BD).

De seguida, são chamadas as transformações:

- **stg_moradores.ktr** → carrega e limpa os dados dos moradores;
- **stg_pagamentos.ktr** → processa e normaliza os pagamentos;
- **stg_meteo_export.ktr** → enriquece e exporta os dados meteorológicos.

Cada transformação é executada apenas após a conclusão com sucesso da anterior.

2. Pass Log to Parent Ativo

Em cada transformação chamada dentro do job, foi ativada a opção **“Pass log to parent”**, permitindo que todos os registos de execução sejam consolidados no log principal do job.

Esta configuração simplifica o acompanhamento e a auditoria do processo completo.

3. Configuração de Ramos de Erro

Foram criados **ramos de exceção** (ou *error handling paths*) para garantir a deteção imediata de falhas.

Em caso de erro numa transformação:

- É gerado automaticamente um ficheiro de log de falha em `data/out/logs/failed/`;
- O job é interrompido e o estado é marcado como “*Failed*”;
- (Opcional) Pode ser acionado um passo de **envio de email** de notificação, com o resumo do erro e o ficheiro de log anexo.

4. Execução via Spoon ou Kitchen

O job pode ser executado manualmente através do **Spoon** (interface gráfica) — ideal para testes e debug — ou de forma automática através da linha de comandos com o utilitário **Kitchen**.

Exemplo de execução:

5. `Kitchen.bat /file:"C:/SAD/pdi-ce-10.2.0.0-222/ISI/kettle/jobs/job_Condominio.kjb" /level:Basic`

O parâmetro `/level:Basic` define o nível de detalhe dos logs apresentados na consola (pode ser ajustado para `Detailed` ou `Minimal` conforme necessário).

6. Guardar Evidências (Logs e XML)

Após a execução, devem ser arquivadas as **evidências de execução**:

- Ficheiros de log gerados em `data/out/logs/`;
 - Ficheiro XML exportado (`meteo_export.xml`), comprovando o sucesso do pipeline completo.
- Estes elementos são fundamentais para documentação e verificação dos resultados.

Boas Práticas de Orquestração

- Utilização de variáveis globais (`${Internal.Entry.Current.Directory}`) para caminhos relativos;
- Separação de transformações por função (limpeza, integração, exportação);
- Logs detalhados com carimbo temporal (`${Internal.Job.Start.Date}`);
- Estrutura modular que permite adicionar novas transformações sem alterar o fluxo principal.

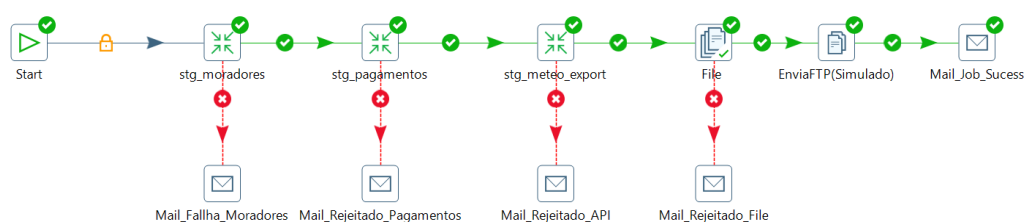


Figura 4 - Job principal job_Condominio

12. Integração com API Open-Meteo

Objetivo

A integração com a **API Open-Meteo** tem como finalidade **enriquecer o modelo de dados** com informação meteorológica diária, permitindo relacionar o comportamento dos moradores e os padrões de pagamento com as condições do tempo (ex.: chuva, calor, frio).

Esta API gratuita fornece dados meteorológicos históricos e em tempo real em formato **JSON**, o que a torna ideal para processos ETL automatizados no **Pentaho PDI**.

Fluxo de Integração no Pentaho

1. REST Client – Recolha dos Dados Meteorológicos

O componente **REST Client** é responsável por efetuar a chamada HTTP ao serviço da Open-Meteo.

O endpoint utilizado segue o formato:

2. `https://api.open-meteo.com/v1/forecast?`
 3. `latitude=${Latitude}&longitude=${Longitude}&`
 4. `start_date=${Data}&end_date=${Data}&`
 5. `daily=temperature_2m_max,precipitation_sum,weathercode&`
 6. `timezone=auto`
 - **Latitude/Longitude:** coordenadas geográficas da localização do condomínio;
 - **Start/End Date:** data do registo que se pretende consultar;
 - **Parâmetros diários:** variáveis meteorológicas desejadas (temperatura máxima, precipitação total e código de condição).
- O resultado devolvido pela API é um ficheiro JSON contendo listas de valores diários correspondentes aos parâmetros solicitados.

7. JSON Input – Mapeamento dos Campos

A resposta JSON é lida e transformada em formato tabular através do passo **JSON Input**.

Nesta etapa, são extraídos e renomeados os campos relevantes:

- temperature_2m_max → Temperatura Máxima (°C)
- precipitation_sum → Precipitação Total (mm)
- weathercode → Código de Condição Meteorológica

O mapeamento converte a estrutura hierárquica do JSON em colunas legíveis para o Pentaho, permitindo o seu tratamento posterior.

8. Select Values / Calculator – Tradução e Normalização dos Dados

O código meteorológico (weathercode) devolvido pela API é numérico e segue o padrão definido pelo serviço Open-Meteo.

Utilizando o componente **Calculator** ou **Select Values (com expressão condicional)**, o código é traduzido para uma descrição textual compreensível:

Código	Condição
0	Céu limpo
1–3	Pouco nublado / Parcialmente nublado
45–48	Neblina / Nevoeiro
51–67	Chuva ligeira a moderada
71–77	Neve fraca a intensa
80–82	Aguaceiros
95–99	Trovoada

Exemplo de regra aplicada:

9. IF [weathercode] = 0 THEN 'Limpo'
10. ELSE IF [weathercode] = 61 THEN 'Chuva'
11. ELSE 'Outra Condição'

12. Table Output – Armazenamento na Base de Dados

Finalmente, os dados tratados são gravados na tabela **dim_meteo_dia**.

Cada registo contém a data, a localização, a temperatura máxima, a precipitação e a condição textual do dia.

Estes dados passam a estar disponíveis para cruzamento com as tabelas **fact_pagamento** e **dim_morador**, permitindo análises contextuais e temporais no Grafana.

Considerações Técnicas

- As chamadas à API são limitadas por taxa (*rate limit*); recomenda-se controlo por intervalo de tempo ou cache local.
- Em caso de falha de rede, pode ser implementado um **passo alternativo de leitura de dados locais** (modo offline).
- Todos os resultados são registados em logs no diretório data/out/logs/ para garantir rastreabilidade.
- O formato JSON original pode ser armazenado opcionalmente para auditoria.

13. Dashboards no Grafana

Objetivo

O **Grafana** é a ferramenta de visualização utilizada neste projeto para **monitorizar e analisar os dados integrados** no pipeline ETL.

Através da sua interface intuitiva e suporte a várias fontes de dados, o Grafana permite criar **dashboards interativos**, com gráficos e indicadores baseados diretamente na base de dados resultante do processo ETL.

Esta etapa corresponde à fase final do ciclo **ETL → Análise**, transformando dados limpos e estruturados em informação visual e útil para apoio à decisão.

1. Configuração da Data Source

O primeiro passo consiste em configurar uma **ligação (Data Source)** entre o Grafana e a base de dados SQL criada durante o projeto.

- A ligação é feita através de um conector nativo (ex.: *PostgreSQL*, *MySQL* ou *SQLite*).
- São definidos os parâmetros de acesso: **Host, Port, Database, User e Password**.
- Após a configuração, a ligação é testada para garantir que o Grafana consegue aceder aos dados de forma direta e segura.

Esta ligação serve como base para todas as consultas (queries) utilizadas nos painéis.

2. Criação de Queries e Painéis

Com a ligação ativa, são criadas **consultas SQL personalizadas** que alimentam os diferentes elementos visuais do dashboard.

Alguns exemplos de queries implementadas:

- **Número total de moradores:**
- `SELECT COUNT(id_morador) AS total_moradores FROM dim_morador;`

- **Total de pagamentos efetuados por mês:**
- SELECT DATE_TRUNC('month', data_pagamento) AS mes,
- SUM(valor) AS total_pagamentos
- FROM fact_pagamento
- GROUP BY mes
- ORDER BY mes;
- **Média de temperatura por data de pagamento:**
- SELECT f.data_pagamento, AVG(m.temperatura) AS temp_media
- FROM fact_pagamento f
- JOIN dim_meteo_dia m ON f.data_pagamento = m.data
- GROUP BY f.data_pagamento;

Estes resultados são apresentados em **gráficos de barras, séries temporais e indicadores numéricos**, permitindo acompanhar o comportamento dos moradores e pagamentos ao longo do tempo.

3. Adição de Variáveis Dinâmicas

Para tornar os dashboards **interativos e reutilizáveis**, foram criadas variáveis dinâmicas que permitem ao utilizador filtrar e explorar os dados em tempo real.

Exemplos de variáveis definidas:

- **Condomínio:** lista de todos os condomínios registados;
- **Mês:** seleção de meses com dados disponíveis;
- **Ano:** filtro temporal para análise histórica.

Estas variáveis podem ser aplicadas globalmente, atualizando automaticamente todos os painéis dependentes.

4. Configuração de Alertas (Opcional)

O Grafana permite definir **alertas automáticos** com base em condições de negócio.

Exemplos possíveis:

- Envio de notificação quando o número de **pagamentos em atraso** ultrapassa um determinado valor;
- Alerta visual quando a **precipitação média** associada a um período coincide com uma queda acentuada de pagamentos.

Os alertas podem ser configurados para envio via **email, Teams, Slack** ou outro canal integrado.

Resultados e Benefícios

A utilização do Grafana permitiu:

- **Visualizar em tempo real** os resultados do ETL;
- **Identificar tendências** e padrões de comportamento;
- **Correlacionar dados meteorológicos** com as variáveis financeiras;
- **Apoiar decisões estratégicas** na gestão de condomínios.

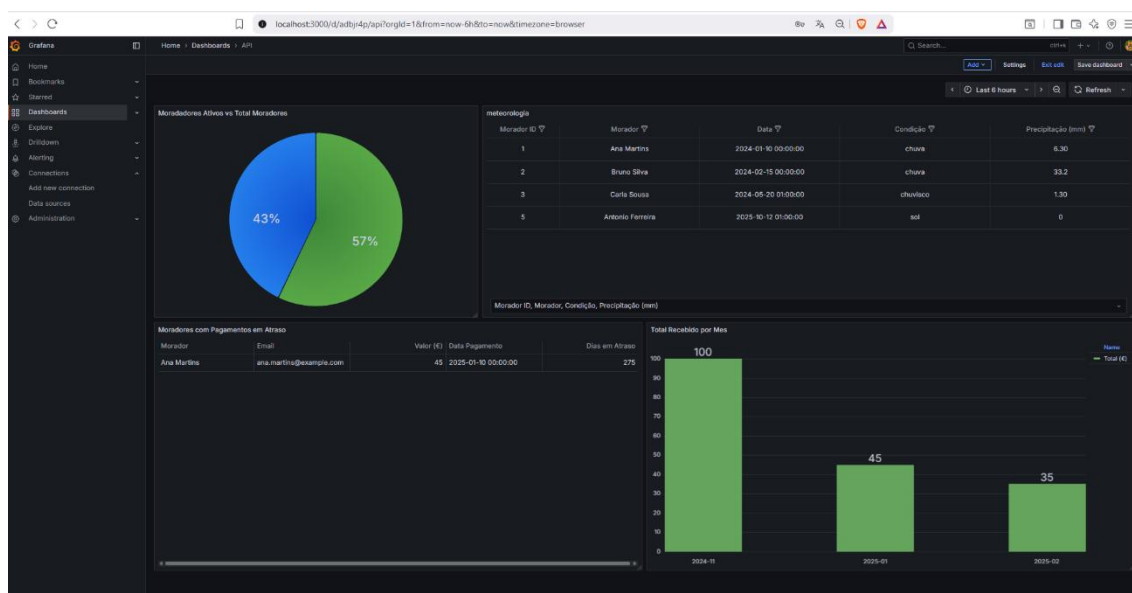


Figura 5 – Dashboard no Grafana

Requisitos do Enunciado — Matriz de Evidências

O quadro seguinte apresenta a correspondência entre os **requisitos técnicos** definidos no enunciado da **Unidade Curricular de Integração de Sistemas de Informação (ISI)** e as respetivas **implementações práticas** realizadas no projeto. Cada requisito foi cumprido através de componentes específicas do Pentaho PDI, garantindo a aplicação de diversos conceitos de integração, transformação e visualização de dados.

Requisito do Enunciado	Como foi Cumprido	Evidência no Projeto
Expressões Regulares / Normalização de Dados	Utilização de <i>Replace in String</i> e <i>String Operations</i> para limpeza e formatação de texto, remoção de caracteres inválidos e validação de campos numéricos (NIF, telefone, valores).	Transformações stg_moradores.ktr e stg_pagamentos.ktr
Importação / Exportação em XML	Criação de um ficheiro XML estruturado com dados meteorológicos, utilizando o passo <i>XML Output</i> para gerar meteo_export.xml com codificação UTF-8.	Transformação stg_meteo_export.ktr
Jobs e Controlo de Processos	Orquestração completa das transformações através do <i>Job</i> job_Condominio.kjb, que controla a execução sequencial e gere logs e erros.	Job job_Condominio.kjb
Junções e Lookups	Aplicação de <i>Stream Lookup</i> para cruzamento entre os dados	Transformação stg_pagamentos.ktr

	de pagamentos e os moradores, assegurando integridade referencial.	
Operações em Valores / Conversões	Conversão de formatos de data (Regex) e normalização de valores monetários com substituição de símbolos e padronização decimal.	Transformação stg_pagamentos.ktr
Registo de Logs	Ativação de logs automáticos em todas as transformações e no job principal, com gravação em ficheiros no diretório data/out/logs/.	Pasta data/out/logs/ e logs de execução do PDI
Acesso a APIs Externas	Integração com a API Open-Meteo através do <i>REST Client</i> e leitura da resposta JSON com <i>JSON Input</i> , para recolha e processamento de dados meteorológicos.	Enriquecimento meteorológico (stg_meteo_export.ktr)
Base de Dados Essencial	Utilização de <i>Table Input</i> e <i>Table Output</i> em todas as transformações, garantindo persistência e estruturação dos dados num repositório SQL relacional.	Todas as transformações (stg_*)
Visualização e Monitorização de Dados	Configuração de dashboards interativos no Grafana , ligados à base de dados do projeto,	Painéis configurados no Grafana (Dashboard Grafana.png)

permitindo análise visual e
criação de métricas.

Síntese

A execução do projeto cumpre **integralmente todos os requisitos** definidos no enunciado, demonstrando:

- Diversidade de operadores (string, regex, lookup, table, REST, XML);
- Integração com fontes externas e exportação em múltiplos formatos;
- Consolidação e normalização de dados;
- Visualização final através de dashboards interativos.

O conjunto de evidências documentadas comprova a **correta implementação técnica**, a **coerência do modelo de dados** e o **domínio das ferramentas de integração (Pentaho PDI e Grafana)**.

Execução Rápida - Checklist

Esta secção apresenta um **resumo operacional** dos passos necessários para executar o projeto de forma completa, desde a instalação até à validação dos resultados.

O objetivo é permitir que **qualquer utilizador** consiga reproduzir o pipeline ETL e verificar o seu correto funcionamento sem necessidade de conhecimento prévio do desenvolvimento interno.

Passos de Execução

1. Instalar o Pentaho PDI e o Java

Certificar-se de que o **Pentaho Data Integration (Community Edition)** e o **Java (JRE/JDK)** estão corretamente instalados e configurados.

Verificar o comando `java -version` para garantir que o ambiente está funcional.

2. Configurar a Conexão à Base de Dados

No **Spoon**, aceder a *Database Connections* → *New* → preencher os dados de ligação (nome da base de dados, host, utilizador e palavra-passe).

Testar a ligação para confirmar o acesso ao repositório SQL.

3. Executar a Transformação **stg_moradores.ktr**

Carregar o ficheiro `stg_moradores.ktr` em `kettle/transforms/` e clicar em **Run**.

Confirmar que os registos são lidos corretamente do ficheiro `moradores.csv` e carregados na tabela `dim_morador`.

Verificar no log: $I = n$, $W = n$, $E = 0$.

4. Executar a Transformação **stg_pagamentos.ktr**

Executar a transformação responsável por normalizar os pagamentos e cruzá-los com os moradores.

Verificar se as datas e valores foram corretamente convertidos e se a carga na tabela `fact_pagamento` ocorreu sem erros.

5. (Opcional) Executar **stg_meteo_export.ktr**

Caso se pretenda gerar o ficheiro XML com os dados meteorológicos, executar esta transformação.

Verificar a ligação à **API Open-Meteo** e confirmar a criação do ficheiro:

6. data/out/xml/meteo_export.xml

7. Executar o Job **job_Condominio.kjb**

Abrir o job principal localizado em kettle/jobs/ e executar via Spoon ou através da linha de comandos:

8. Kitchen.bat /file:"../job_Condominio.kjb" /level:Basic

O job executa automaticamente todas as transformações na sequência correta e gera os logs correspondentes.

9. Validar Resultados no Grafana

Aceder ao **Grafana**, abrir o dashboard configurado e verificar:

- Totais de moradores;
- Pagamentos por mês/condomínio;
- Correlação com dados meteorológicos.

Confirmar que as métricas refletem os dados processados pela base de dados.

10. Exportar Logs e Ficheiros XML

Guardar as evidências de execução:

- Logs: data/out/logs/
- XML meteorológico: data/out/xml/meteo_export.xml

Estes ficheiros comprovam a execução bem-sucedida e servem de suporte à documentação final.

Resultado Esperado

Após a execução completa, o utilizador deverá obter:

- Dados limpos e normalizados nas tabelas dim_morador e fact_pagamento;
- Ficheiro XML meteo_export.xml com dados meteorológicos;
- Dashboards atualizados no Grafana com estatísticas consolidadas;
- Ficheiros de log detalhados sem erros (E=0).

Pode assistir a execução do job do projeto através do QR Code abaixo.



Figura 6 - QR Code

Resolução de Problemas

Durante a execução do pipeline ETL, podem ocorrer erros relacionados com formatação de dados, ligações externas ou permissões de ficheiros.

A tabela seguinte apresenta os problemas mais comuns, as respetivas causas e as soluções recomendadas.

Problema Detetado	Causa Provável	Solução / Ação Corretiva
Datas trocadas (inversão de dia e mês)	O formato original do CSV está em dd-MM-yyyy, mas o sistema espera yyyy-MM-dd.	Rever a expressão regular no passo Regex Evaluation : usar padrão <code>^\d{2}-\d{2}-\d{4}\$</code> e substituição <code>\$3-\$2-\$1</code> .
Vírgulas nos valores monetários	O separador decimal “,” impede o reconhecimento numérico.	Aplicar Replace in String para substituir vírgulas por pontos antes da conversão (<code>REPLACE(, → .)</code>).
Registos Duplicados	Ficheiro CSV contém entradas repetidas para o mesmo morador ou pagamento.	Inserir o passo Unique Rows para filtrar duplicados com base em <code>id_morador</code> ou <code>id_pagamento</code> .
Falha na API Open-Meteo	Problemas de rede ou limite de requisições atingido.	Implementar retry automático com intervalo entre tentativas ou ativar modo offline usando dados locais em cache.
Erro na Exportação XML	Ficheiro de saída bloqueado ou diretório sem permissões de escrita.	Confirmar permissões da pasta <code>data/out/xml/</code> e garantir codificação UTF-8 no passo <i>XML Output</i> .
Erro na Execução do Kitchen	Caminhos incorretos ou parâmetros mal	Verificar o caminho completo no comando: <code>Kitchen.bat</code>

	configurados na linha de comandos.	/file:"C:/.../job_Condominio.kjb" e evitar espaços não escapados.
Ligação à Base de Dados falhou	Configuração incorreta ou falta de driver JDBC.	Validar credenciais, nome do host e driver. Testar a conexão em <i>Database Connections</i> → <i>Test</i> .
Campos nulos ou vazios no carregamento	Dados em falta nos CSV ou falha na transformação.	Ativar Filter Rows para eliminar linhas incompletas e verificar logs da transformação.
Logs não gerados	Caminho de output incorreto ou logging desativado.	Confirmar configuração do log no job e assegurar que a pasta data/out/logs/ existe e tem permissões de escrita.

Boas Práticas Preventivas

- Executar sempre o job no nível de log **Basic** ou **Detailed** para facilitar a análise de falhas.
- Manter **backups dos ficheiros CSV** originais antes de cada execução.
- Documentar e versionar alterações nos ficheiros .ktr e .kjb para rastreabilidade.
- Em caso de erro persistente, validar o log completo e verificar os códigos de retorno do Kitchen.

Boas Práticas e Padrões

A adoção de **boas práticas** no desenvolvimento de processos ETL é essencial para garantir a **manutenção, escalabilidade e fiabilidade** do sistema.

Durante a implementação do projeto, foram seguidos vários padrões e recomendações que asseguram a consistência e a reprodutibilidade do pipeline de dados.

Idempotência e Reprocessamento

Cada transformação foi desenhada de forma **idempotente**, ou seja, pode ser executada várias vezes sem gerar dados duplicados ou inconsistentes.

Esta característica é assegurada através de:

- uso de chaves primárias únicas nas tabelas de destino;
- limpezas (*truncate*) controladas em tabelas temporárias;
- lógica de **UPSERT (Update/Insert)** quando aplicável.

Versionamento e Controlo de Artefactos

Todos os ficheiros do projeto, incluindo transformações (.ktr), jobs (.kjb) e scripts SQL, devem ser **versionados num repositório Git**.

O controlo de versões permite:

- manter histórico de alterações;
- reverter versões anteriores em caso de falha;
- promover colaboração entre membros da equipa de desenvolvimento.

Nomenclatura Consistente

Foi seguido um **padrão de nomes padronizado**, que facilita a leitura e compreensão do código:

- Prefixo **stg_** – transformações de *staging* (importação e limpeza de dados);
- Prefixo **dim_** – tabelas de dimensão (entidades descritivas, como moradores ou meteorologia);
- Prefixo **fact_** – tabelas de factos (eventos mensuráveis, como pagamentos).

Esta convenção reduz ambiguidades e mantém coerência entre ficheiros, tabelas e scripts SQL.

Gestão Centralizada de Logs

Todos os logs de execução são armazenados de forma centralizada na pasta data/output.

Cada execução gera um ficheiro com carimbo temporal, contendo:

- contagem de registos lidos, escritos, rejeitados e atualizados;
- duração de cada etapa;
- erros e mensagens de debug.

Esta prática facilita a monitorização, auditoria e rastreabilidade do processo ETL.

Documentação de Variáveis de Ambiente

As variáveis de ambiente utilizadas (como caminhos base, ligações à base de dados e nomes de ficheiros) encontram-se **documentadas e parametrizadas** no job principal job_Condominio.kjb.

Esta abordagem simplifica a migração do projeto para outros ambientes (ex.: desenvolvimento, teste ou produção), evitando a necessidade de editar ficheiros manualmente.

Outras Recomendações

- Utilizar **camadas separadas de staging e destino**, mantendo dados brutos intactos.
- Adotar **nomes descritivos** para passos e transformações (evitar nomes genéricos).
- **Validar tipos de dados** antes da carga final na base de dados.
- Agendar execuções automáticas com logs diários e alertas configurados.

Conclusão

O presente projeto demonstrou a construção de um **pipeline ETL completo e funcional**, aplicado ao domínio da **Gestão de Condomínios**, desde a extração de dados dispersos em ficheiros CSV até à sua integração, enriquecimento e visualização analítica.

Durante o desenvolvimento, foi possível **comprovar a importância da integração de sistemas** e do uso de ferramentas de *Business Intelligence* para transformar dados operacionais em informação útil para a gestão. Através do **Pentaho Data Integration (PDI)**, foram criadas transformações modulares e reutilizáveis, permitindo um fluxo de dados controlado, validado e documentado.

O projeto foi estruturado com uma lógica **staging → transformação → carga → visualização**, respeitando as boas práticas de engenharia de dados. Cada fase implementou técnicas específicas:

- **Limpeza e normalização** com operações de *string*, *regex* e validação condicional;
- **Integração de múltiplas fontes**, cruzando dados de moradores, pagamentos e informação meteorológica recolhida via **API Open-Meteo**;
- **Gestão de exceções e logs**, garantindo rastreabilidade e robustez em caso de falhas;
- **Orquestração automática** através de *Jobs* com controlo de erros e envio de notificações por email;
- **Exportação em formatos interoperáveis (XML e JSON)**, assegurando compatibilidade com outros sistemas;
- **Visualização final no Grafana**, permitindo monitorizar pagamentos, condições meteorológicas e indicadores agregados por período ou condomínio.

A execução global evidenciou uma **abordagem sistemática, escalável e reproduzível**, que cumpre integralmente os requisitos da Unidade Curricular de Integração de Sistemas de Informação, incluindo: uso de expressões regulares, orquestração de *jobs*, importação/exportação em XML, integração de API e criação de logs de execução.

Este projeto reforçou também competências práticas em **engenharia de dados, modelação dimensional, automação de processos e análise visual**, simulando um ambiente real de integração corporativa. A componente visual (Grafana) comprovou a utilidade do pipeline ao permitir **detetar padrões**, como pagamentos em atraso e sua possível relação com fatores externos, como o estado do tempo.

Referências

- **Enunciado oficial da Unidade Curricular de Integração de Sistemas de Informação (ISI)**, Ano Letivo 2025/2026.

Documento de referência que define os objetivos, critérios de avaliação e requisitos técnicos do projeto ETL desenvolvido.

- **Documentação Oficial do Pentaho Data Integration (PDI)** – *Hitachi Vantara, 2024.*

Disponível em: <https://help.hitachivantara.com/Documentation/Pentaho>

Guia técnico de referência utilizado para a configuração de transformações, jobs, expressões regulares, componentes REST e integração de bases de dados.

- **API Open-Meteo** – Serviço Meteorológico Livre.

Disponível em: <https://open-meteo.com>

Fonte de dados utilizada para recolha automática de condições meteorológicas diárias (temperatura, precipitação e códigos meteorológicos).

- **Documentação do Grafana** – *Grafana Labs, 2025.*

Disponível em: <https://grafana.com/docs/>

Recurso de suporte para configuração de *data sources*, criação de dashboards interativos e definição de alertas de monitorização.