Instituto Politécnico do Cávado e Ave
Licenciatura em Engenharia de Sistemas Informáticos

Relatório Técnico - Trabalho Prático I
Integração de Sistemas de Informação (ISI)

Tema:

Sistema de Análise e Monitorização de Criptomoedas

Aluno: Rodrigo Cruz
Nº de estudante: 27971

Docentes:
Óscar Ribeiro

Outubro de 2025

Conteúdo

Capítulo 1

Enquadramento

Este trabalho foi desenvolvido no âmbito da Unidade Curricular de **Integração de Sistemas de Informação (ISI)**, do curso de **Engenharia de Sistemas Informáticos**. O seu principal objetivo é aplicar e consolidar os conceitos de integração de dados e processos de **ETL (Extract, Transform, Load)** com recurso a ferramentas profissionais e *open-source*, nomeadamente o **KNIME**, o **PostgreSQL** e o **Node-RED**.

O projeto tem como foco a recolha, transformação e análise de dados de criptomoedas provenientes de uma **API pública**, permitindo posteriormente a sua visualização dinâmica através de **dashboards interativos**.

Ferramentas Utilizadas

- **KNIME Analytics Platform** – para extração e transformação de dados (ETL);
- **PostgreSQL** – como base de dados relacional e persistência dos dados transformados;
- **Node-RED** – para criação de dashboards e visualização dos dados;
- **APIs REST (CoinGecko)** – como fonte de dados JSON;

Capítulo 2

Problema

O objetivo consiste em desenvolver um sistema que permita:

1. Recolher dados atualizados sobre as principais criptomoedas;
2. Tratar e transformar a informação para posterior análise;
3. Armazenar os dados em base relacional para consultas otimizadas;
4. Visualizar indicadores de mercado, tais como:
    - Preço atual (€);
    - Market Cap;
    - Volume 24h;
    - Percentagem de dominância;
    - Volatilidade 24h;

Esta solução enquadra-se nos desafios de integração e monitorização de dados financeiros, alinhando-se com os objetivos da Unidade Curricular e com o contexto empresarial de análise de grandes volumes de dados em tempo real.

Capítulo 3

Estratégia Utilizada

O sistema foi desenvolvido com uma arquitetura de três camadas:

1. **Extração:** recolha de dados JSON e CSV a partir da API pública;
2. **Transformação:** limpeza, normalização, categorização e cálculo de métricas no KNIME;
3. **Load:** inserção dos dados tratados em tabelas PostgreSQL;
4. **Visualização:** dashboards criados no Node-RED.

Foram ainda aplicadas:

- Expressões regulares para normalização e limpeza de texto;
- Criação de logs automáticos;
- Operações de *join*, *filter*, *group by* e cálculos percentuais;
- Exportação dos dados em formato JSON e CSV.

Capítulo 4

ETL

Nesta secção é detalhado o processo de **extração, transformação e carga (ETL)** desenvolvido no ambiente **KNIME Analytics Platform**, com ênfase especial na fase de **Load**. O sistema foi concebido para integrar múltiplas fontes de dados, normalizar a informação recolhida e carregar os resultados num **sistema de base de dados PostgreSQL**, garantindo a integridade e consistência da informação.

4.1 Extração de Dados

A fase de extração teve como principal objetivo a recolha de dados provenientes de duas fontes distintas:

1. **API pública CoinGecko:** utilizada para recolher informação em tempo real sobre as principais criptomoedas, incluindo campos como preço atual, capitalização de mercado (*market cap*), volume transacionado, variação percentual nas últimas 24 horas, e dados de oferta em circulação;

2. **Ficheiros CSV:** obtidos manualmente a partir do portal CoinGecko, contendo dados históricos de preço, volume e capitalização de mercado por moeda, em intervalos diários.

A Figura 4.1 ilustra o excerto do *workflow* referente à recolha de dados. No KNIME, o nó `GET Request` foi utilizado para realizar chamadas HTTPS à API do CoinGecko, seguido por um nó `JSON to Table`, que converte a resposta JSON em colunas estruturadas. Cada campo relevante (por exemplo, `symbol`, `price`, `market_cap`, `total_volume`, `last_updated`) foi mapeado e posteriormente transformado.

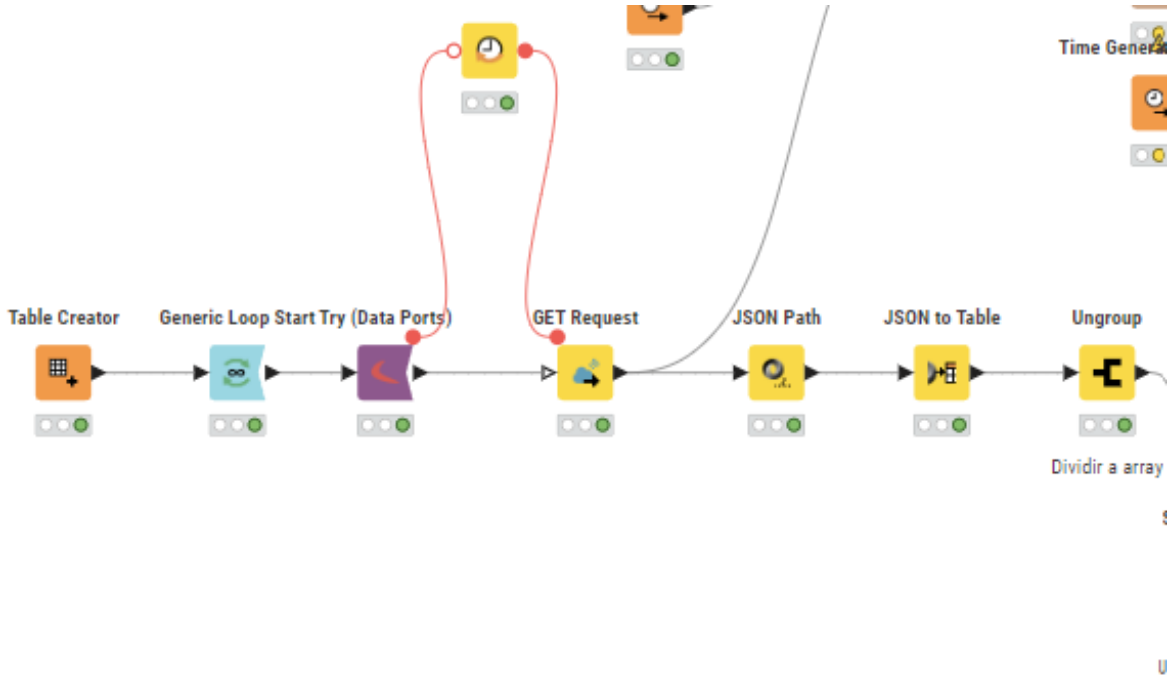

Figura 4.1: Fluxo de extração de dados via API CoinGecko.

É possível ver na 4.2 que para os ficheiros CSV, foi utilizado o nó `CSV Reader`, configurado para percorrer automaticamente uma diretoria contendo múltiplos ficheiros, cada um associado a uma moeda específica (por exemplo: `btc.csv`, `eth.csv`, `sol.csv`, `ada.csv`, `doge.csv`). Esta estratégia permitiu consolidar o histórico de várias moedas numa única tabela agregada.

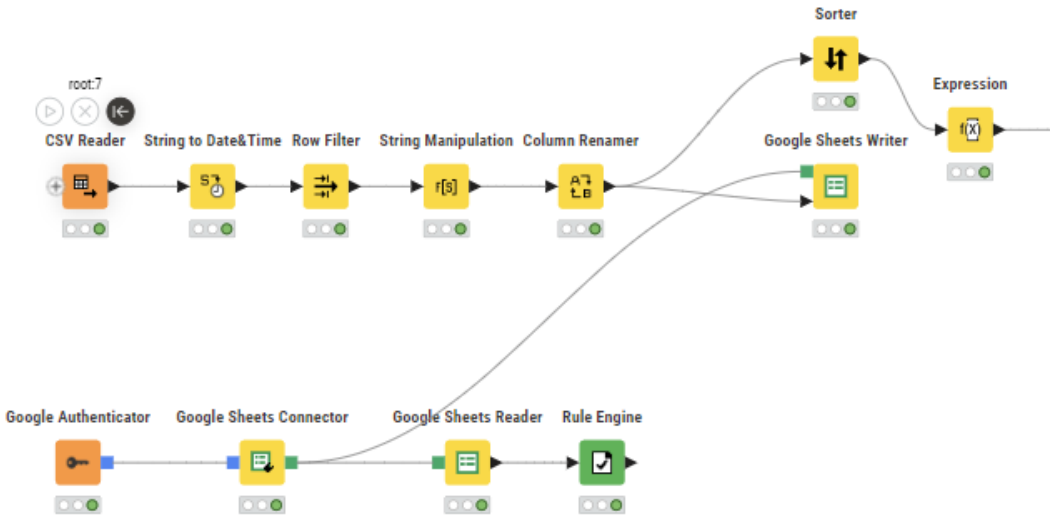

Figura 4.2: Fluxo de extração de dados via API CoinGecko.

4.2 Transformações Aplicadas

Após a extração, foi implementada uma sequência de transformações de dados para garantir qualidade e coerência. Entre as principais operações realizadas destacam-se:

- **Normalização de símbolos:** utilizando expressões regulares no nó `String Manipulation` para remover caracteres inválidos e uniformizar os identificadores das moedas;
- **Conversão de tipos:** aplicação do nó `String to Number` para transformar campos numéricos provenientes do JSON em formato `double`, garantindo compatibilidade com operações matemáticas;
- **Formatação de valores:** inclusão de separadores de milhar e limitação a duas casas decimais;
- **Criação de métricas calculadas:**
  - `price_change_24h%` – variação percentual de preço nas últimas 24h;
  - `ROI_24h` – retorno percentual diário calculado manualmente;
  - `market_dominance` – percentagem de dominância no total de capitalização do mercado;
  - `supply_ratio` – percentagem de moedas em circulação face à oferta total;

Cada transformação foi validada individualmente utilizando nós de pré-visualização e verificações com `Rule Engine` e `Missing Value`, assegurando a inexistência de valores nulos ou inconsistentes.

4.3 Gestão de Erros e Registo de Logs

Para garantir a robustez do processo de integração, foi implementado um mecanismo de controlo de falhas e registo de logs no ambiente KNIME. Este módulo assegura que o sistema responde adequadamente a interrupções de rede, falhas temporárias na API ou atrasos na resposta do servidor CoinGecko.

A Figura 4.3 apresenta o subfluxo responsável por este controlo, composto por um conjunto de nós que permitem reexecutar automaticamente a chamada à API até que uma resposta válida seja recebida ou que um número máximo de tentativas seja atingido.

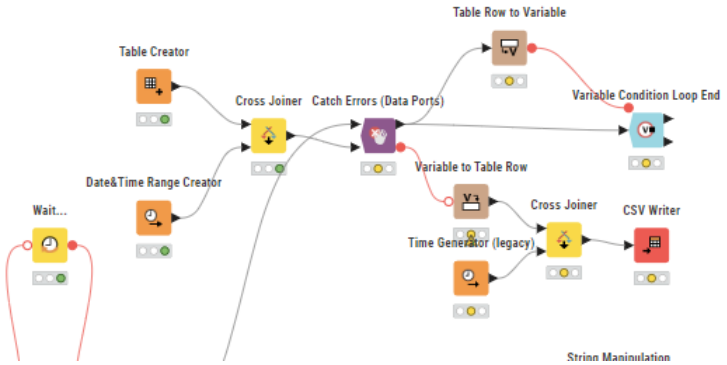

Figura 4.3: Subfluxo de controlo de erros, logs e repetição de execução da API.

O processo inicia-se com o nó `DateTime Range Creator`, responsável por gerar uma sequência temporal de tentativas. Este é combinado com o `Table Creator` e um `Cross Joiner`, criando as combinações necessárias para gerir o ciclo de repetição.

O nó `GET Request` é o elemento central desta estrutura, responsável pela comunicação direta com a API do *CoinGecko*. Durante o seu funcionamento, este nó regista automaticamente os **logs de execução e de erro**, incluindo:

- Códigos de resposta HTTP (ex.: 200);
- Mensagens de erro devolvidas pelo servidor da API;
- Ligações rejeitadas;

Estes logs são capturados pelo nó `Catch Errors (Data Ports)`, que processa as exceções e encaminha as mensagens de erro para um ficheiro local através do nó `CSV Writer`. Assim, o sistema mantém um histórico completo das interações com a API, incluindo as causas das falhas e o número de tentativas realizadas até uma resposta bem-sucedida.

Em caso de erro, o fluxo volta ao início do ciclo através de um `Variable Condition Loop`, reexecutando o pedido após um intervalo de espera definido no nó `Wait...`. A utilização dos nós `Variable to Table Row` e `Table Row to Variable` garante que as variáveis de estado são atualizadas dinamicamente entre as iterações.

Este processo garante:

- A execução contínua até que os dados sejam obtidos com sucesso;
- A documentação automática dos erros ocorridos.

4.4 Load

Concluídas as transformações, procedeu-se à fase de carga para o sistema de gestão de base de dados **PostgreSQL**. O processo foi realizado em três etapas:

1. **Criação das tabelas:** através do nó `DB Table Creator`, definindo explicitamente o esquema da base de dados (*schema*) e o nome da tabela. Foram criadas as seguintes estruturas:
   - **fact_market_data** – tabela principal contendo os dados atuais e métricas calculadas;
   - **fact_history** – tabela destinada ao histórico de preços diários;
   - **dim_coin** – tabela dimensional com informação descritiva de cada moeda (símbolo, nome, categoria).
2. **Ligação ao PostgreSQL:** utilizando o nó `DB Connector`, configurado com parâmetros locais (`host=localhost`, `port=5432`, `database=ETL_database`, `user=postgres`).
3. **Inserção de dados:** o nó `DB Writer` foi responsável por inserir os registos provenientes do KNIME nas tabelas criadas. Foi configurado para sobrescrever a tabela em execuções de teste e para adicionar novas linhas em execuções incrementais.

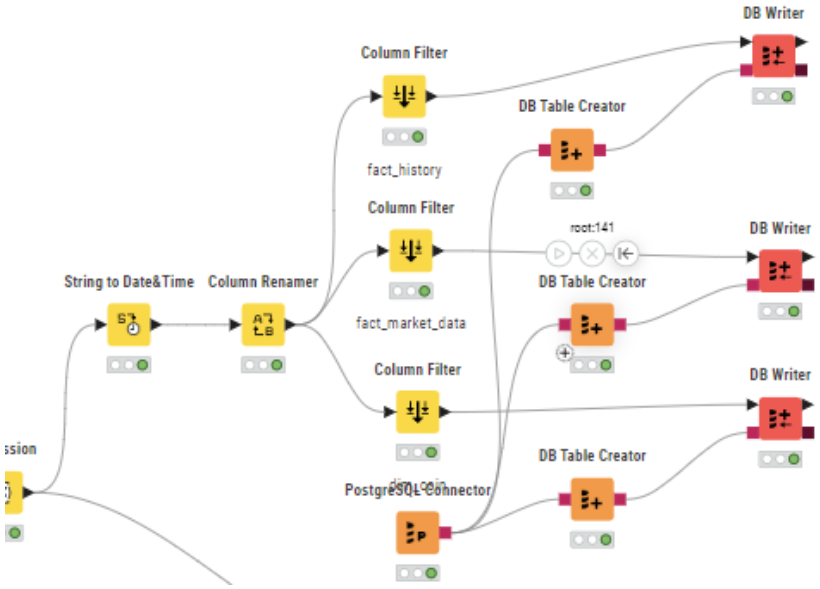

Figura 4.4: Processo Load dos dados transformados para a base de dados PostgreSQL.

Após a conclusão do processo, foi verificado o sucesso da carga através de consultas SQL diretas, utilizando:

```
SELECT COUNT(*) FROM fact_market_data;
SELECT symbol, price, market_dominance FROM fact_market_data ORDER BY market_dominance
```

Estas consultas confirmaram a correta inserção e coerência dos dados carregados, permitindo posteriormente a integração direta com o **Node-RED** para visualização interativa.

4.5 Load para Google Sheets

Para complementar o processo de carga no sistema PostgreSQL, foi implementada uma segunda camada de exportação de dados, destinada à plataforma **Google Sheets**, permitindo assim a partilha dinâmica e a consulta rápida dos dados transformados. Esta integração foi realizada através dos nós `Google Authenticator`, `Google Sheets Connector` e `Google Sheets Writer`.

A Figura 4.5 apresenta o subfluxo responsável pela ligação ao Google Sheets, ilustrando as fases de autenticação, transformação e escrita dos dados.

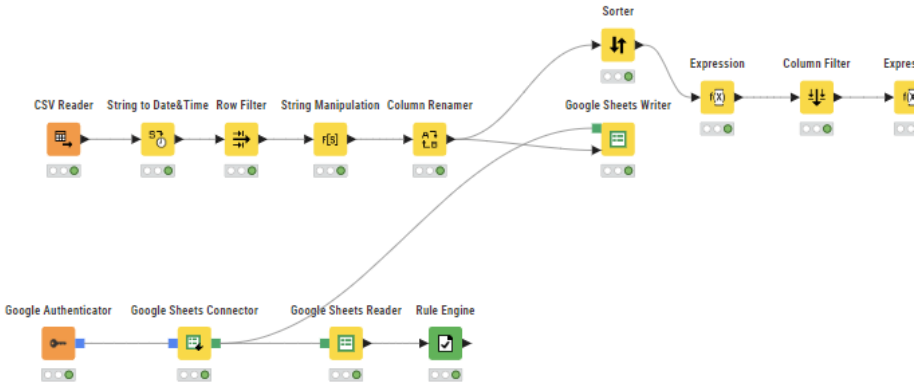

Figura 4.5: Processo de autenticação e exportação de dados para o Google Sheets.

O processo inicia-se com o nó `Google Authenticator`, que utiliza as credenciais OAuth2 para autenticar o utilizador de forma segura, garantindo o cumprimento das normas de segurança impostas pela *Google Identity Platform*.

Após a autenticação, o nó `Google Sheets Connector` estabelece a ligação entre o ambiente KNIME e a API do Google Sheets, permitindo o acesso controlado às folhas de cálculo partilhadas.

De seguida, o nó `Google Sheets Writer` é responsável pela escrita dos dados na folha de cálculo previamente configurada, substituindo ou acrescentando os registos conforme os parâmetros definidos. Este processo é precedido por uma sequência de preparação e filtragem de dados `CSV Reader`, `String to DateTime`, `Row Filter`, `String Manipulation` e `Column Renamer` garantindo que os dados carregados no Google Sheets mantêm coerência com o modelo relacional existente na base de dados principal.

Este método de exportação foi adotado com o objetivo de:

- Facilitar o acesso dos *stakeholders* aos dados mais recentes sem necessidade de aceder diretamente à base de dados;
- Proporcionar visualizações e cálculos rápidos através das funcionalidades nativas do Google Sheets;
- Garantir redundância e cópia de segurança dos dados tratados;

- Promover transparência e colaboração entre equipas.

A integração com o Google Sheets representa uma prática recomendada em pipelines ETL modernos, permitindo que dados internos de sistemas empresariais coexistam com plataformas de análise em nuvem.

4.6 Envio Automático de Notificação com caminho para o Ficheiro Final

Após a conclusão bem-sucedida do processo de extração, transformação e carga (ETL), foi implementado um mecanismo de envio automático de notificação por correio eletrónico, utilizando o nó `Send Email` do KNIME Analytics Platform. O objetivo desta funcionalidade é comunicar ao utilizador responsável que o processo terminou corretamente e fornecer o caminho direto para o ficheiro Excel final que contém os dados consolidados das moedas analisadas.

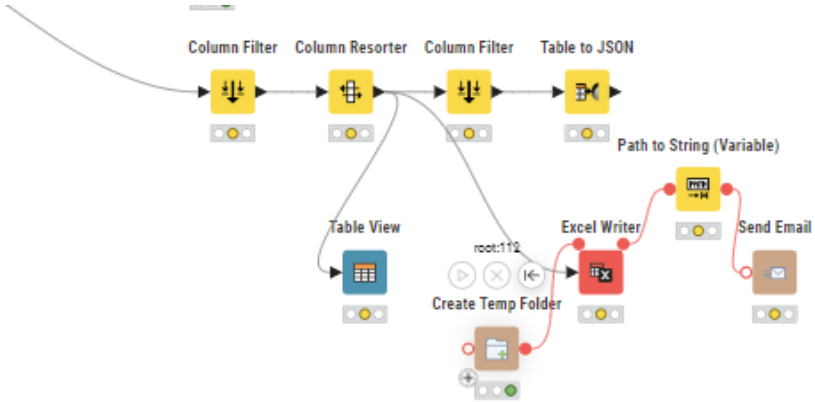

Figura 4.6: Subfluxo de envio de email automático com o caminho do ficheiro Excel final.

O nó `Send Email` foi configurado para operar de forma automática, sendo ativado no final do fluxo principal de execução. A mensagem enviada contém uma estrutura simples e informativa, composta por:

- O assunto do e-mail, que indica o estado da execução (*"ETL concluído com sucesso"*);
- O corpo da mensagem, com um breve texto explicativo;
- O caminho completo para o ficheiro Excel gerado (por exemplo: `C:\ETL_Results\final_data.xls`
- O registo temporal (`timestamp`) da conclusão da execução.

A informação do caminho é passada ao nó `Send Email` através de uma *Flow Variable*, criada dinamicamente após a execução do nó `Excel Writer`, garantindo que a mensagem contém o endereço exato do ficheiro gerado naquela instância. Dessa forma, evita-se a necessidade de procurar manualmente o ficheiro no sistema, promovendo uma automatização completa da comunicação pós-processamento.

A Figura 4.6 ilustra a estrutura deste subfluxo, onde o nó `Send Email` é ligado diretamente à fase final do processo de carga. Esta ligação garante que o e-mail apenas é enviado após a escrita bem-sucedida do ficheiro Excel.

Esta abordagem apresenta várias vantagens operacionais:

- Elimina a necessidade de monitorização manual do pipeline;
- Garante que o utilizador é informado assim que o ficheiro é gerado;

- Facilita o acesso direto ao resultado final do processo;
- Permite futuras extensões, como envio de anexos ou notificações múltiplas.

4.7 Visualização de dados

Com o objetivo de complementar a análise numérica e permitir uma interpretação visual dos dados recolhidos, foram criados diversos gráficos de linha (*Line Plots*) no KNIME Analytics Platform, representando a evolução temporal do preço de cada criptomoeda incluída no estudo.

A Figura 4.7 ilustra o subfluxo responsável por esta componente, no qual é aplicado um `Row Filter` para cada moeda, seguido de um nó `Line Plot` individual. Esta estrutura modular permite gerar automaticamente um gráfico distinto para cada ativo digital analisado: *Bitcoin (BTC)*, *Ethereum (ETH)*, *Solana (SOL)*, *Cardano (ADA)* e *Dogecoin (DOGE)*.

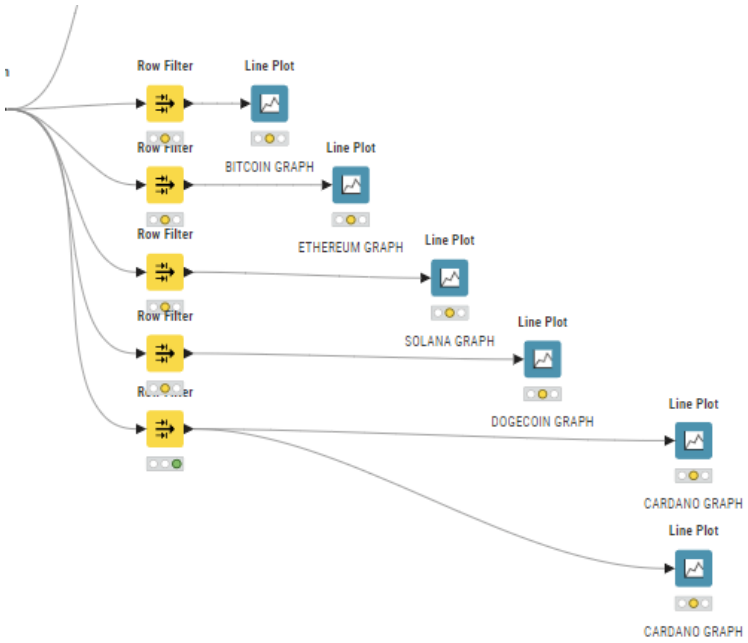

Figura 4.7: Subfluxo de geração de gráficos de linha individuais para cada moeda.

Cada `Line Plot` foi configurado para representar:

- O eixo X (`snapped_at`) como variável temporal;
- O eixo Y (`price`) representando o valor do preço em euros (€);
- Linhas contínuas com suavização visual, permitindo identificar oscilações ao longo do tempo;
- Títulos e legendas personalizados para cada gráfico, facilitando a leitura e comparação.

A opção por gráficos individuais, em vez de um único gráfico agregado, permite observar com maior detalhe as variações e comportamentos específicos de cada moeda, sem sobreposição de dados. Esta abordagem favorece a clareza e torna o processo de análise mais eficiente, especialmente em contextos de comparação temporal.

Exemplo na Figura 4.8 de mudança do preço ao longo do tempo.

Figura 4.8: Exemplo de gráfico

4.8 Implementação e Visualização em Node-RED

O Node-RED foi utilizado como camada de visualização e monitorização do sistema desenvolvido, desempenhando o papel de *front-end* interativo para a análise dos dados provenientes da base de dados *PostgreSQL*.

4.8.1 Integração com a Base de Dados PostgreSQL

Para o acesso aos dados armazenados, foi utilizada a ligação entre o nó `PostgreSQL` e os módulos de configuração da base de dados. Estes módulos realizam consultas SQL a tabelas como `fact_market_data`, `fact_info_data` e `coin_history`, que contêm informações sobre preços, volumes de mercado e indicadores financeiros das criptomoedas.

Um exemplo típico de consulta utilizada é:

```
SELECT id_coin, price, market_cap, total_volume, supply_ratio
FROM fact_market_data
ORDER BY market_cap DESC
LIMIT 5;
```

Este tipo de query permite apresentar, em tempo real, as cinco criptomoedas com maior capitalização de mercado, cujos resultados são posteriormente formatados e exibidos num componente de tabela (`ui_table`) do *dashboard*.

4.8.2 Estrutura dos Fluxos (Flows)

O desenvolvimento em Node-RED foi organizado em vários *flows*, cada um com uma função analítica específica. A Figura 4.9 apresenta a disposição geral dos principais fluxos criados, nomeadamente:

- **Top 5 Criptomoedas:** mostra as cinco moedas com maior *market cap* atual;
- **ROI 24h:** apresenta a variação percentual de preço nas últimas 24 horas;
- **Fact Info Data:** exibe dados técnicos de cada moeda (máximos, mínimos, ROI e categorias);
- **Gráfico BTC e ETH:** representam, em gráfico de linhas, a evolução temporal dos preços históricos das moedas;

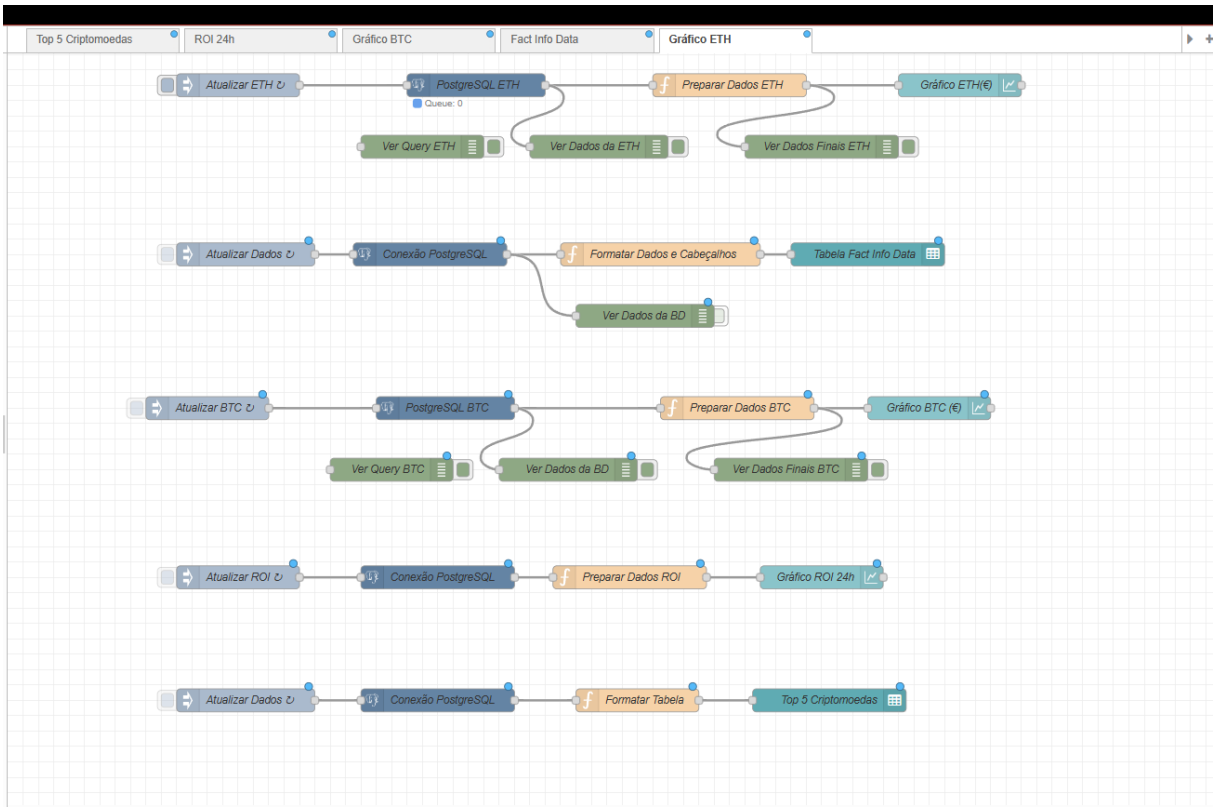

Figura 4.9: Visão geral dos fluxos implementados no Node-RED.

4.8.3 Formatação e Tratamento de Dados

Antes de apresentar os resultados ao utilizador, foi realizada uma etapa de formatação através de nós do tipo `Function`. Estes nós aplicam transformações nos dados devolvidos pelo PostgreSQL, ajustando os valores para o formato europeu com separador de milhar por ponto e vírgula para decimais e adicionando o símbolo do euro (€) sempre que aplicável. Exemplo de função usada para formatação:

```
function formatEuro(num, decimals = 2) {
        return num
            .toFixed(decimals) // garante as casas decimais
            .replace('.', ',') // substitui ponto por vírgula
            .replace(/\B(?=(\d{3})+(?!\d))/g, '.') + ' €';
    }
```

Esta abordagem garante a consistência dos dados apresentados e melhora a legibilidade no painel visual.

4.8.4 Dashboards e Visualização

Foram criados vários painéis interativos (*dashboards*) permitindo a visualização de dados em tempo real. Os principais elementos incluem:

- **Tabelas Dinâmicas:** mostram dados financeiros e métricas de desempenho;
- **Gráficos de Linhas:** representam a evolução temporal de preços (BTC, ETH);

- **Indicadores de Atualização Automática:** cada `Inject Node` está configurado para executar periodicamente, garantindo atualização constante.

A Figura 4.10 ilustra o fluxo responsável pela representação da evolução de preço da Ethereum (€), composto pelos nós: `Inject`, `PostgreSQL`, `Function` e `UI Chart`.

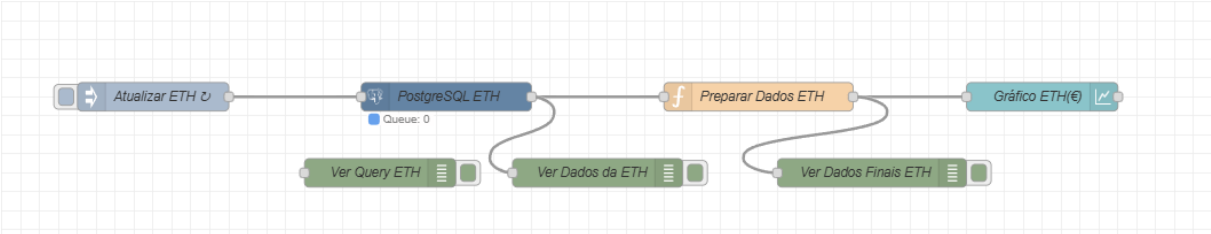

Figura 4.10: Fluxo de consulta e visualização do preço histórico da Ethereum no Node-RED.

Figura 4.11: Dashboard de preços.

4.8.5 Conclusão da Componente Node-RED

A utilização do Node-RED permitiu a criação de um ecossistema visual, interativo e modular para a análise de dados de criptomoedas, potenciando a integração entre a camada de base de dados e a de visualização. Esta abordagem revelou-se eficiente e intuitiva, promovendo a observação em tempo real de métricas de desempenho e facilitando a compreensão das tendências de mercado.

O resultado final é um conjunto de *dashboards* funcionais e informativos, essenciais para suportar decisões analíticas.

Capítulo 5

Conclusão

O desenvolvimento deste projeto permitiu aplicar, de forma prática, os conceitos de Integração de Sistemas de Informação e de processos ETL (Extract, Transform, Load) no contexto da análise e monitorização de criptomoedas. A implementação de um pipeline completo no **KNIME Analytics Platform** demonstrou como é possível recolher, tratar e visualizar dados provenientes de múltiplas fontes, garantindo consistência, fiabilidade e automatização.

Na fase de **Extração**, recorreu-se à **API pública do CoinGecko** para recolha de dados em tempo real incluindo preços, capitalização de mercado, volume e variação percentual em 24 horas e a ficheiros **CSV históricos**, que complementaram a análise temporal das moedas estudadas. Esta abordagem assegurou uma integração híbrida, combinando dados atualizados com registos passados, permitindo comparações mais completas e análises de tendências.

Foram ainda aplicadas expressões regulares (*Regex*) para normalização de símbolos e validação de campos numéricos e temporais, garantindo qualidade e coerência nos dados antes da carga final.

A fase de **Load** envolveu dois destinos distintos:

1. A base de dados **PostgreSQL**, utilizada como repositório principal, onde foram criadas tabelas dimensionais e factuais de forma estruturada;

2. A integração com o **Google Sheets**, configurada com autenticação OAuth2 através do nó `Google Authenticator`, permitindo disponibilizar os dados em ambiente colaborativo e acessível em tempo real.

Além disso, foi configurado um mecanismo de **logs automáticos** que monitoriza a execução do fluxo, através do nó `Catch Errors`, registando eventuais falhas de ligação ou erros na API em ficheiros `CSV`. Adicionalmente, foi implementado o envio de uma notificação automática por email, contendo o caminho do ficheiro Excel gerado, informando o utilizador da conclusão bem-sucedida do processo.

Em termos de resultados, o sistema demonstrou ser capaz de:

- Automatizar a recolha e tratamento de dados financeiros;
- Garantir a consistência e atualização dos registos;
- Disponibilizar informação visual e interpretável em diferentes plataformas;
- Permitir uma monitorização contínua e extensível a novas fontes de dados.

Conclui-se que o projeto atingiu todos os objetivos propostos, evidenciando a importância da integração de dados e da automação na gestão de informação em tempo real. O trabalho desenvolvido reforça a aplicabilidade prática das metodologias de ETL e demonstra o potencial de ferramentas open-source como o KNIME e o Node-RED para soluções de monitorização financeira.

Como desenvolvimento futuro, considero pertinente a inclusão de mecanismos de previsão de preços baseados em aprendizagem automática e a integração de alertas em tempo real para variações de mercado, reforçando a utilidade operacional do sistema no apoio à decisão.

Trabalhos futuros

- Implementação de modelos de previsão de preços (Machine Learning);
- Automatização da atualização em tempo real;
- Envio de alertas (e-mail ou Telegram) em caso de variação anómala de preço.

Capítulo 6

Referências

- KNIME Documentation. (2024). *KNIME Analytics Platform User Guide*.
- PostgreSQL Global Development Group. (2024). *PostgreSQL Documentation 16*.
- CoinGecko API. (2025). *Cryptocurrency Market API Reference*.