Relatório do trabalho da disciplina de Integração de Sistemas de Informação

Processos de ETL

João Rafael Azevedo Cunha – a21598

Licenciatura em Engenharia Sistemas Informáticos (Pós-Laboral)

Outubro de 2024

ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Afirmo por minha honra que não recebi qualquer apoio não autorizado na realização deste trabalho prático. Afirmo igualmente que não copiei qualquer material de livro, artigo, documento web ou de qualquer outra fonte exceto onde a origem estiver expressamente citada.

João Rafael Azevedo Cunha – a21598

Índice

Índice de Figuras

Enquadramento

Este trabalho, desenvolvido no âmbito da disciplina de Integração de Sistemas de Informação (ISI), tem como objetivo a aplicação e experimentação de ferramentas para processos de ETL (*Extract, Transformation and Load*), essenciais na integração de dados em sistemas de informação.

Para a concretização deste objetivo, o projeto utiliza o *Microsoft SQL Server Integration Services* (MSSIS) como principal ferramenta de desenvolvimento dos processos de ETL, complementando a análise e visualização dos dados através do *Power BI*, uma ferramenta poderosa de *business intelligence*. Desta forma, é possível extrair, transformar e carregar dados de forma eficiente, apresentando uma solução que permite obter insights a partir dos dados sem a necessidade de recorrer a grandes volumes de código. O uso destas ferramentas facilita a implementação de processos automatizados e acessíveis, promovendo uma maior produtividade e reduzindo a complexidade técnica associada à extração de informação.

Neste contexto, decidi utilizar dados públicos no formato *JSON* relacionados com clubes de futebol, ligas e jogo. O objetivo é criar um fluxo de trabalho que permitisse a extração de dados, transformá-los e organizá-los de forma eficiente, destacando as informações mais relevantes. Posteriormente, os resultados foram apresentados de maneira clara e visual, com gráficos que ilustravam o desempenho das equipas e as estatísticas dos jogos.

Foram exploradas várias funcionalidades essenciais no processo de ETL, como o uso de expressões regulares (*Regex*) para limpeza de dados, *joins* para combinar informações de diferentes fontes e instruções if para aplicar regras de transformação e filtragem. Além disso, a conexão ODBC permitiu integrar dados de múltiplas origens, facilitando a unificação e posterior análise no *Power BI*, demonstrando a eficiência do processo ETL no tratamento de grandes volumes de informação.

Problema

Nos dias de hoje, a quantidade de dados disponíveis sobre futebol, que inclui estatísticas de jogos, informações sobre clubes e jogadores, cresce a cada instante. No entanto, esses dados estão frequentemente dispersos em diversas plataformas e formatos, como *JSON*, *CSV* e *APIs*, o que dificulta a sua coleta e análise. Essa fragmentação torna a extração de insights valiosos um grande desafio.

Além disso, muitos dados não estão prontos para análise. Eles precisam passar por processos de limpeza e transformação para que possam ser utilizados de forma eficaz. Mesmo após essa transformação, a visualização das informações pode não ser clara, dificultando a interpretação e a tomada de decisões pelos gestores e analistas.

Diante desse cenário, o objetivo deste projeto é criar um fluxo de trabalho ETL que possibilite a extração, transformação e carregamento eficiente de dados sobre clubes de futebol. Utilizando o *Microsoft SQL Server Integration Services* (MSSIS) e o *Power BI*, o projeto busca abordar os desafios mencionados e apresentar uma solução integrada para facilitar a análise dos dados. Ao abordar esses aspetos, o projeto não só ilustra a aplicação

prática das ferramentas de ETL, mas também mostra como a análise de dados pode impactar a gestão no futebol, transformando dados brutos em insights valiosos e melhorando a tomada de decisões.

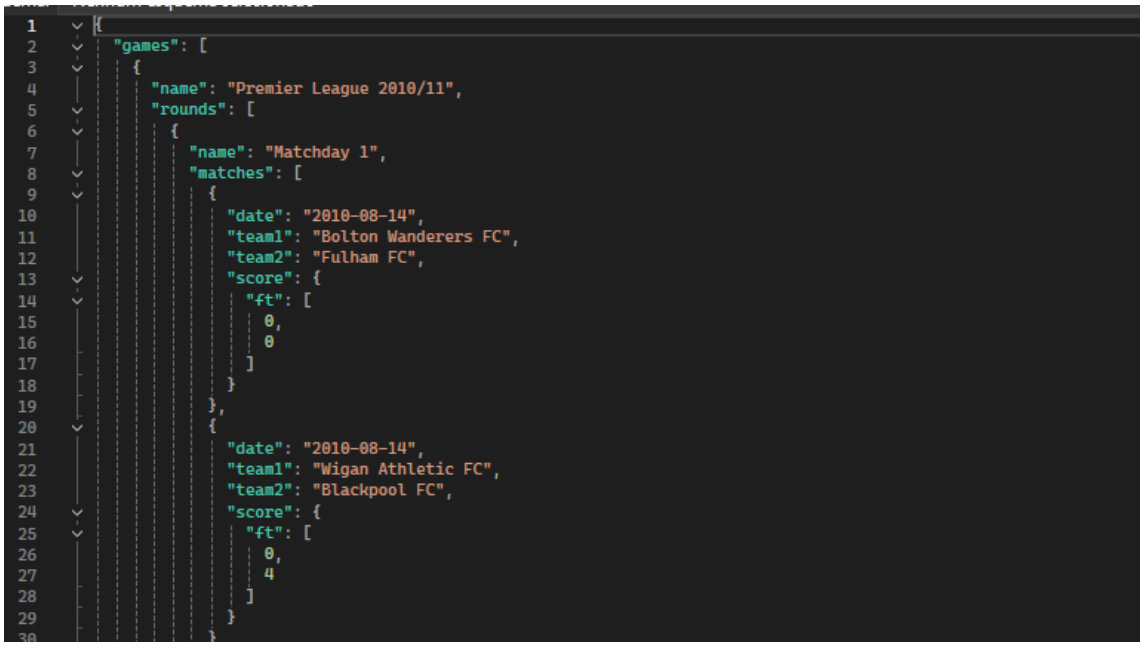

Figura 1 - games.json

Figura 2 - leagues.json

Estratégia Utilizada

Para alcançar os objetivos do projeto, adotei uma estratégia em várias etapas que me guiou ao longo do processo de ETL, tornando-o mais organizado e eficiente. Aqui está um resumo das principais ações que realizei:

1. Escolha dos Dados: Comecei por definir quais dados seriam úteis para a minha análise sobre clubes de futebol. Optei por dados públicos em formato *JSON*, que são ricos em informações e facilmente acessíveis.

2. **Extração de Dados**: Utilizando o *Microsoft SQL Server Integration Services* (MSSIS), criei um fluxo de trabalho que automatizou a extração desses dados. Isso permitiu-me recolher informações de várias fontes de forma rápida e simples, juntando tudo num só lugar.
3. **Transformação e Limpeza**: Após a recolha dos dados foi realizada uma transformação nos dados. Utilizei expressões regulares para limpar informações desnecessárias e garantir que tudo estivesse no formato correto. Também fiz combinações de dados de diferentes fontes, utilizando *joins*, para ter uma visão mais completa. Além disso, apliquei algumas regras para filtrar e organizar os dados conforme as minhas necessidades.
4. **Integração dos Dados**: Com o auxílio de conexões ODBC, consegui unir todas as informações num único conjunto. Isso facilitou bastante a análise, pois agora tinha tudo integrado, evitando perder-me em dados dispersos.
5. **Análise e Visualização**: Na etapa final, utilizei o *Power BI* para criar *dashboards* interativos. Assim, pude apresentar gráficos e relatórios que ilustravam de forma clara o desempenho das equipas e as estatísticas dos jogos, tornando os insights muito mais acessíveis.

Transformações

1. Estrutura da Base de dados

A transformação dos dados começou pela passagem das informações de um ficheiro *JSON* para uma base de dados com ligação *ODBC*. Para hospedar a base de dados, utilizei o site *FreeSQLDatabase*, que fornece uma solução gratuita. Isso permitiu que eu armazenasse e gerisse os dados de forma acessível, sem a necessidade de infraestrutura complexa. A interface do *phpMyAdmin* facilitou a criação e administração das tabelas na base de dados.

Figura 3 - freesqldatabase Logo

O seguinte código exprime a construção das tabelas usadas na base de dados.

```
CREATE TABLE `Json_Jogos` (
  `NomeLiga` varchar(255) DEFAULT NULL,
  `Jogo` varchar(255) DEFAULT NULL,
  `Data` varchar(255) DEFAULT NULL,
  `Equipa_Um` varchar(255) DEFAULT NULL,
  `Equipa_Dois` varchar(255) DEFAULT NULL,
  `Resultado` varchar(255) DEFAULT NULL,
  `ID_VS` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `Json_Clubes` (
  `Nome_Liga` varchar(255) DEFAULT NULL,
  `Clube_Nome` varchar(255) DEFAULT NULL,
  `Clube_Code` varchar(255) DEFAULT NULL,
  `Clube_Pais` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

A tabela Json_Clubes e Json_Jogos tem como função armazenar todos os dados provenientes dos ficheiros games.json e leagues.json sem realizar a alteração dos mesmos. A estratégia desta tabela é armazenar todos os dados num local é realizar a formatação dos dados a partir do mesmo garantindo que não haja alteração de dados nem possibilidade de corromper os dados originais.

Após a realização da transformação dos dados, os mesmos serão depositados nas seguintes tabelas:

```
CREATE TABLE `DimClubes` (
  `ID` int(11) NOT NULL,
  `Nome` varchar(255) NOT NULL,
  `ID_Pais` int(11) NOT NULL,
  `Nome_Codigo` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE `DimCompeticoes` (
  `ID` int(11) NOT NULL,
  `Nome` varchar(255) NOT NULL,
  `Pais_ID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `DimCompeticoes` (
  `ID` int(11) NOT NULL,
  `Nome` varchar(255) NOT NULL,
  `Pais_ID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `DimPais` (
  `ID` int(11) NOT NULL,
  `Nome` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `DimJogo` (
  `ID` int(255) NOT NULL,
  `ID_Liga` int(255) NOT NULL,
  `ID_JOGO` int(255) NOT NULL,
  `Data` varchar(255) NOT NULL,
  `ID_Equipa` int(255) NOT NULL,
  `Resultado` int(255) NOT NULL,
  `ID_VS` int(255) NOT NULL,
  `Vitoria` int(255) NOT NULL,
```

```
  `Temporada` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Estás tabelas tem função de armazenar os dados transformados e prontos para consulta. A transformação e filtragem de dados deve ser feita a rigor e de forma a darem respostas sucintas aos objetivos pretendidos.

2. Jobs e Transformações

Finalizando a construção da base de dados passamos então para a construção do ETL. Quando usado o *Microsoft SQL Server Integration Services* (MSSIS) devemos ter em atenção a sua estrutura de construção, cada processo global é designado por *package/Jobs*, dentro dos mesmos serão executados as tarefas e os processos de *Transform*. Em seguida podemos ver a explicação do cada *package*:

a. Job GetClubes

A seguinte figura retrata o fluxo de controlo do *package* Job GetClubes.

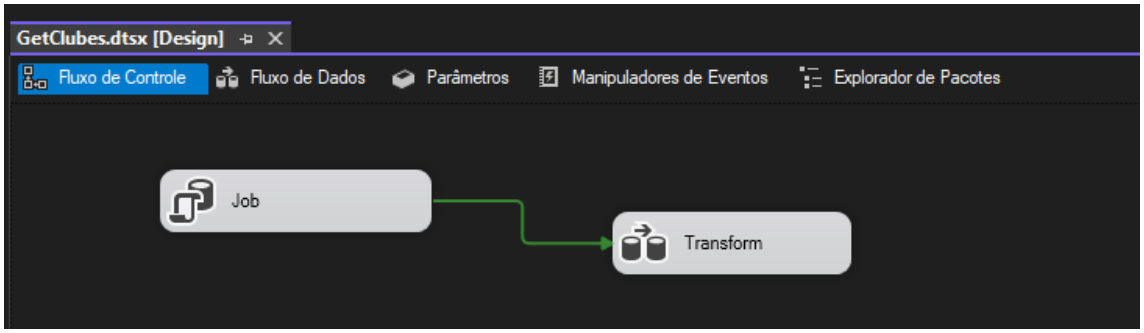

Figura 4 - Fluxo de Controle Job GetClubes

O procedimento Job no presente *package* tem como função realizar um *truncate* na tabela Json_Clubes, fazendo então um *reseat* completo na tabela. Neste tipo de estratégias é necessário não existir ruído que afete o resultado final. Após ter sucesso na sua operação este avança para o processo Transform.

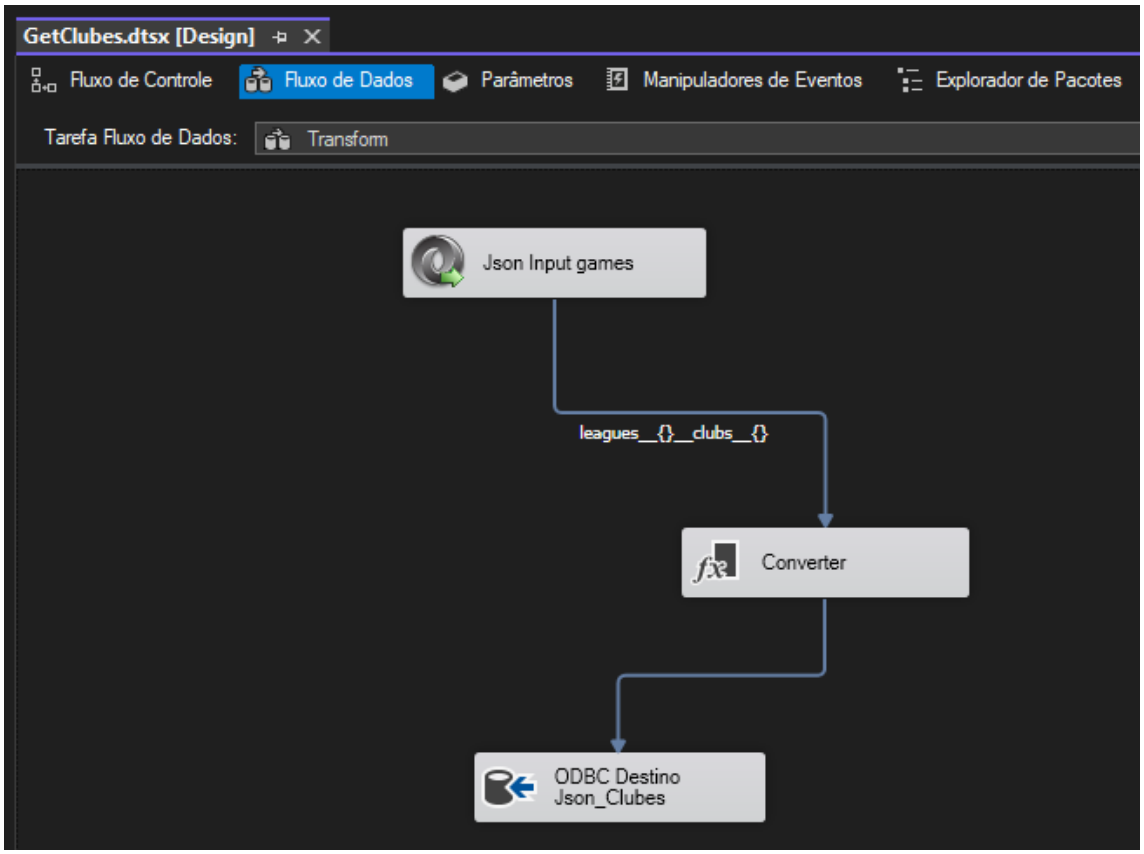

Figura 5 - Fluxo de Dados Job GetClubes

A figura anterior retrata o fluxo de dados do *package* Job GetClubes onde é realizada a tranformação dos dados. Os dados são carregados pelo o ficheiro Json, passam por uma conversão pois alguns dados são apresentados como [DT_WSTR], este formato de dados por pre-defenição não são aceites na base de dados então devem ser convertidos em [DT_STR]. Após a conversão estes são depositados na base de dados *MySql* na tabela Json_Clubes.

b. Job GetJogos

A seguinte figura retrata o fluxo de controlo do *package* Job GetJogos.

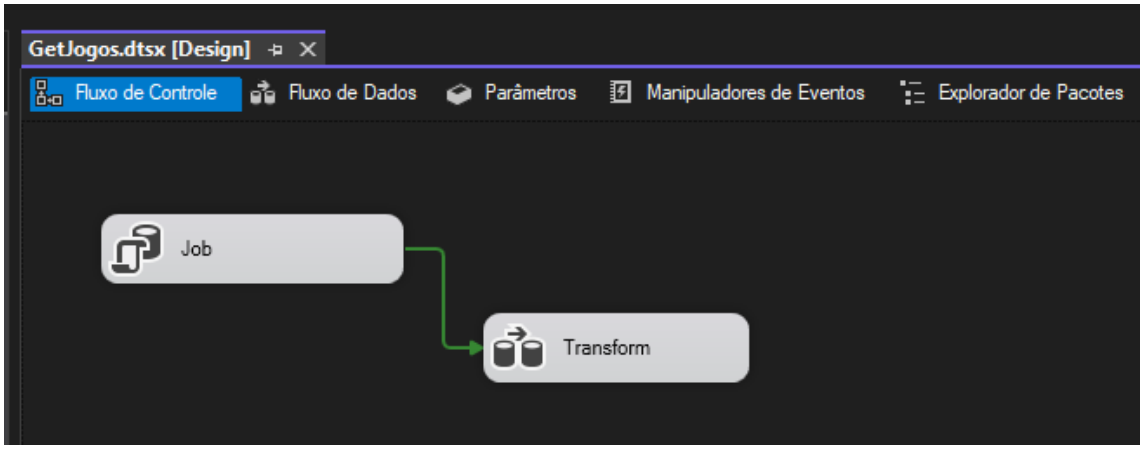

Figura 6 - Fluxo de Controle Job GetJogos

O procedimento Job no presente *package* tem como função realizar um *truncate* na tabela Json_Jogos, fazendo então um *reseat* completo na tabela. Neste tipo de estratégias é necessário não existir ruído que afete o resultado final. Após ter sucesso na sua operação este avança para o processo Transform.

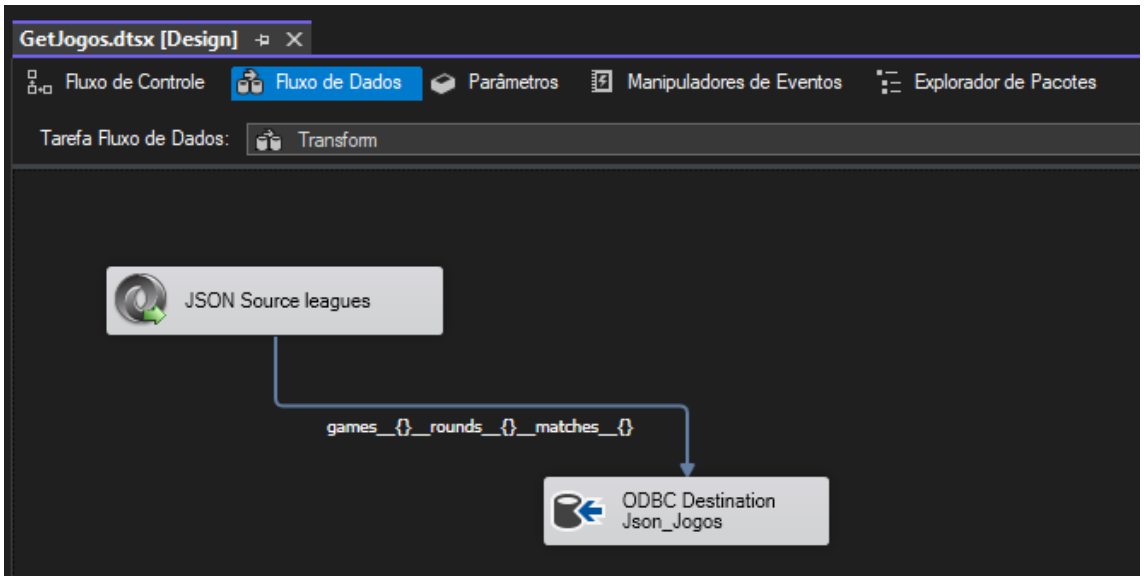

Figura 7 - Fluxo de Dados Job GetJogos

A figura anterior retrata o fluxo de dados do *package* Job GetJogos onde é realizada a tranformação dos dados. Os dados são carregados pelo o ficheiro Json, passam por uma conversão pois alguns dados são apresentados como [DT_WSTR], este formato de dados por pre-defenição não são aceites na base de dados então devem ser convertidos em [DT_STR]. Após a conversão estes são depositados na base de dados *MySql* na tabela Json_Jogos.

c. Job DimPais

A seguinte figura retrata o fluxo de controlo do *package* Job GetJogos.

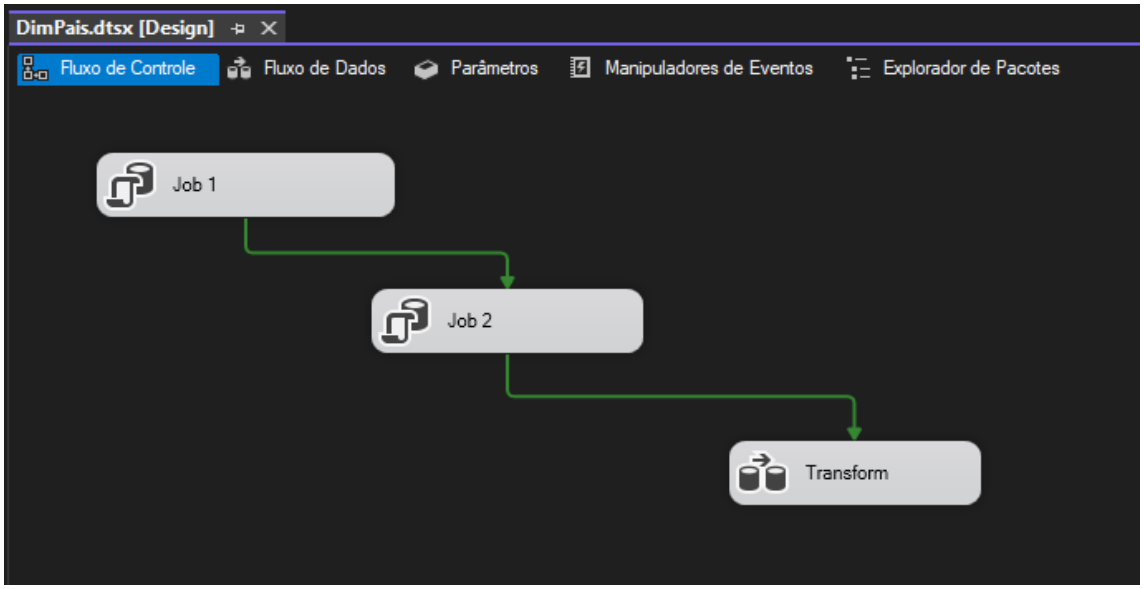

Figura 8 - Fluxo de Controle Job DimPais

O procedimento Job_1 no presente *package* tem como função realizar um *truncate* na tabela DimPais, fazendo então um *reseat* completo na tabela. Neste tipo de estratégias é necessário não existir ruído que afete o resultado final. Após ter sucesso na sua operação este avança para o processo Job_2. O processo Job_2 tem como função inserir na tabela como um valor *default,* neste caso será o valor *undefined* que ira ter como id o valor 0. Este valor serve para identificar os clubes aos quais não foram detetados a existência de um país. Após o sucesso na sua operação este avança para o processo Transform.

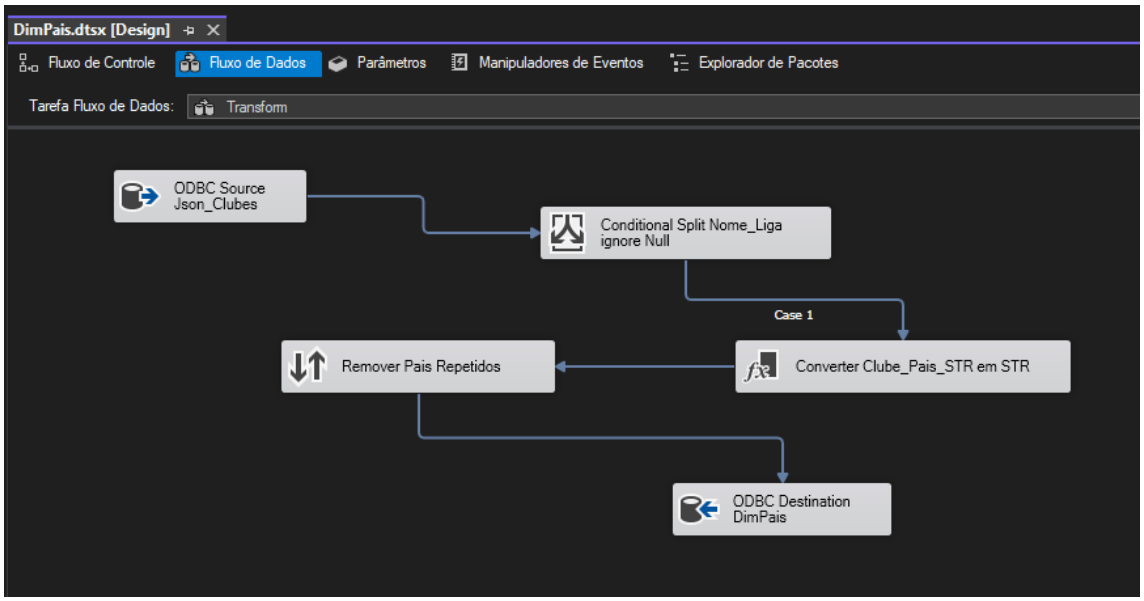

Figura 9 - Fluxo de Dados Job Dim

A figura anterior retrata o fluxo de dados do *package* Job DimPais onde é realizada a tranformação dos dados. Os dados são carregados pela tabela Json_Clubes, passam por uma verificação de existencia de null (Conditional Split Nome_Liga ignore Null) onde este usa o campo "NomeLiga" como argumento para ignorar os registos com valores a null. De seguida converte o campo "Clube_Pais" em [DT_STR], no processo seguinte (Remover Pais Repetidos) deixa unicamente um resgito por pais, estes são depositados na base de dados *MySql* na tabela DimPais.

d. Job DimCompeticoes

A seguinte figura retrata o fluxo de controlo do *package* Job DimCompeticoes.

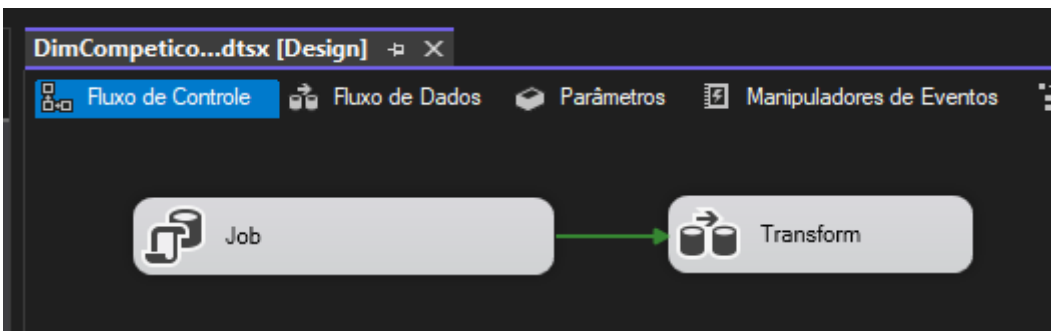

Figura 10 - Fluxo de Controle Job DimCompeticoes

O procedimento Job no presente *package* tem como função realizar um *truncate* na tabela DimCompeticoes, fazendo então um *reseat* completo na tabela. Neste tipo de estratégias é necessário não existir ruído que afete o resultado final. Após ter sucesso na sua operação este avança para o processo Transform.

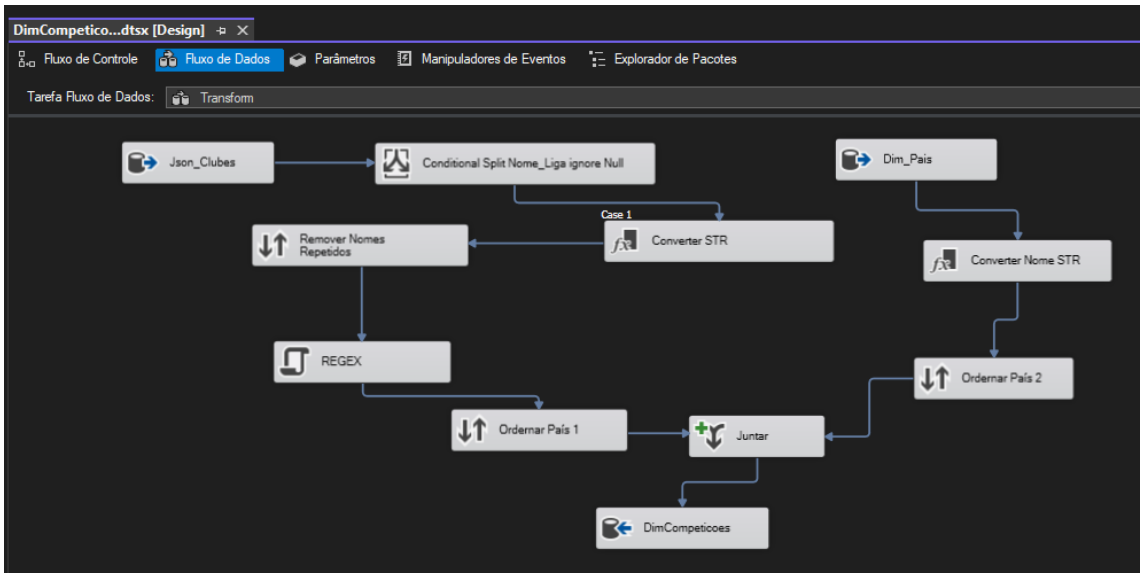

Figura 11 - Fluxo de Dados Job DimCompeticoes

A figura anterior retrata o fluxo de dados do *package* DimPais onde é realizada a tranformação dos dados. Os dados são carregados pela tabela Json_Clubes, passam por uma verificação de existencia de null (Conditional Split Nome_Liga ignore Null) onde este usa o campo "NomeLiga" como argumento para ignorar os registos com valores a null. De seguida realiza uma converção nos campos "Nome_Liga" e "Clube_Pais" em tipo [DT_STR]. Após estes processos existe a remoção de nomes repetidos, tendo assim de forma única os nomes das ligas onde de seguida este passa por uma tranformação de *regex* onde remove qualquer ruido no nome da liga, por fim é realizado um join com os valores da DimPais para indicar a que pais pertence cada liga, estes são depositados na base de dados *MySql* na tabela DimCompeticoes.

e. DimClubes

A seguinte figura retrata o fluxo de controlo do *package* Job DimClubes.

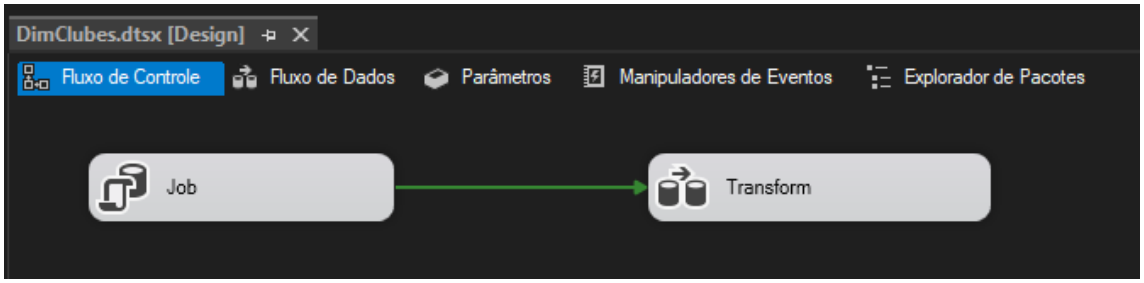

Figura 12 - Fluxo de Controle Job DimClubes

O procedimento Job no presente *package* tem como função realizar um *truncate* na tabela DimClubes, fazendo então um *reseat* completo na tabela. Neste tipo de estratégias é necessário

não existir ruído que afete o resultado final. Após ter sucesso na sua operação este avança para o processo Transform.

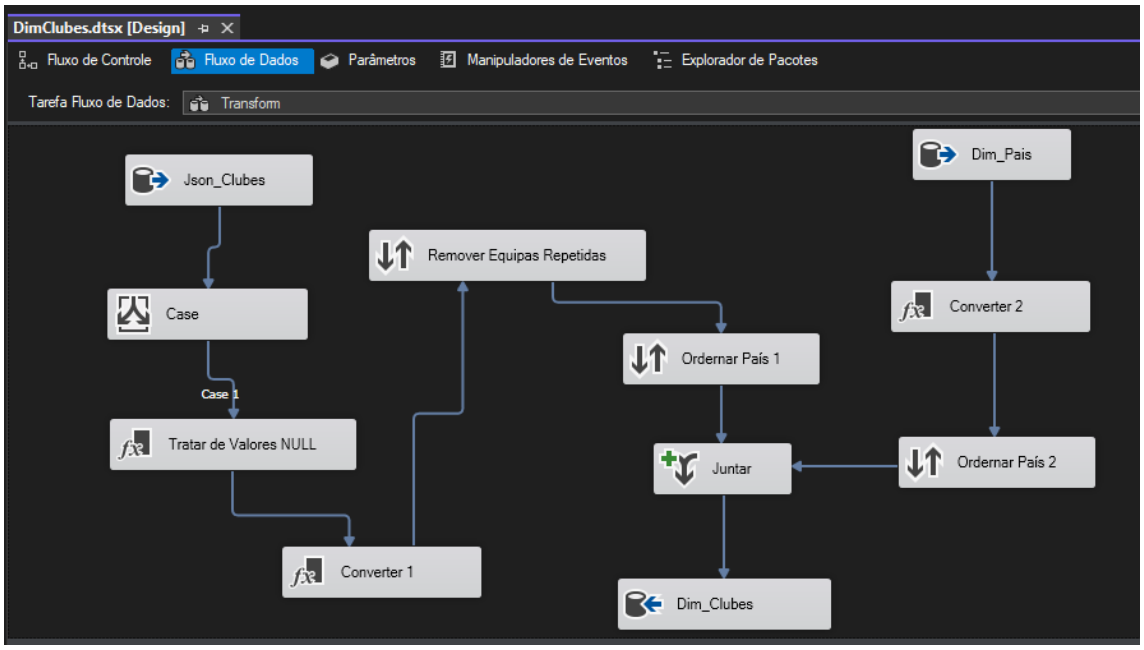

Figura 13 - Fluxo de Dados Job DimClubes

A figura anterior retrata o fluxo de dados do *package* Job DimClubes onde é realizada a transformação dos dados. Os dados são carregados pela tabela Json_Clubes, passam por uma verificação de existência de null (Case) onde este usa o campo "Clube_Nome" como argumento para ignorar os registos com valores a null. De seguida avança para o processo (Tratar de Valores NULL) que terá a função de transformar os seguintes valores:

- ISNULL(Clube_Code) ? "undefined" : (DT_STR,255,1252)Clube_Code
- ISNULL(Clube_Pais) ? "undefined" : (DT_STR,255,1252)Clube_Pais

Caso não seja indicado no ficheiro original o código do clube "Clube_Code" ou o pais do mesmo "Clube_Pais" este ira atribuir o valor "undefined". Após o sucesso desta operação o mesmo avança para a remoção de equipas repetidas (Remover Equipas Repetidas), ordena os valores pelo nome do pais e atribui/subestirui o valor original pelo ID do pais registado na DimPais, caso não encontre é atribuído o id com o valor "undefined" presente na tabela DimPais, estes dados transformados são depositados na base de dados *MySql* na tabela DimClubes.

f. DimJogos

A seguinte figura retrata o fluxo de controlo do *package* DimJogos_1.

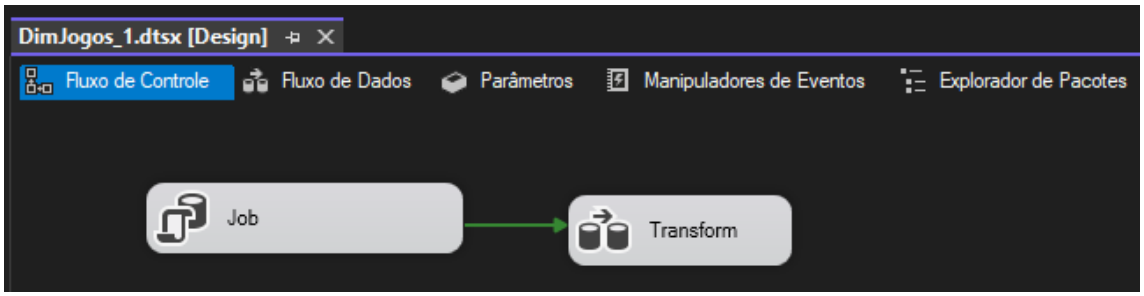

Figura 14 - Fluxo de Controle Job DimJogos

O procedimento Job no presente *package* tem como função realizar um *truncate* na tabela DimJogo, fazendo então um *reseat* completo na tabela. Neste tipo de estratégias é necessário não existir ruído que afete o resultado final. Após ter sucesso na sua operação este avança para o processo Transform.

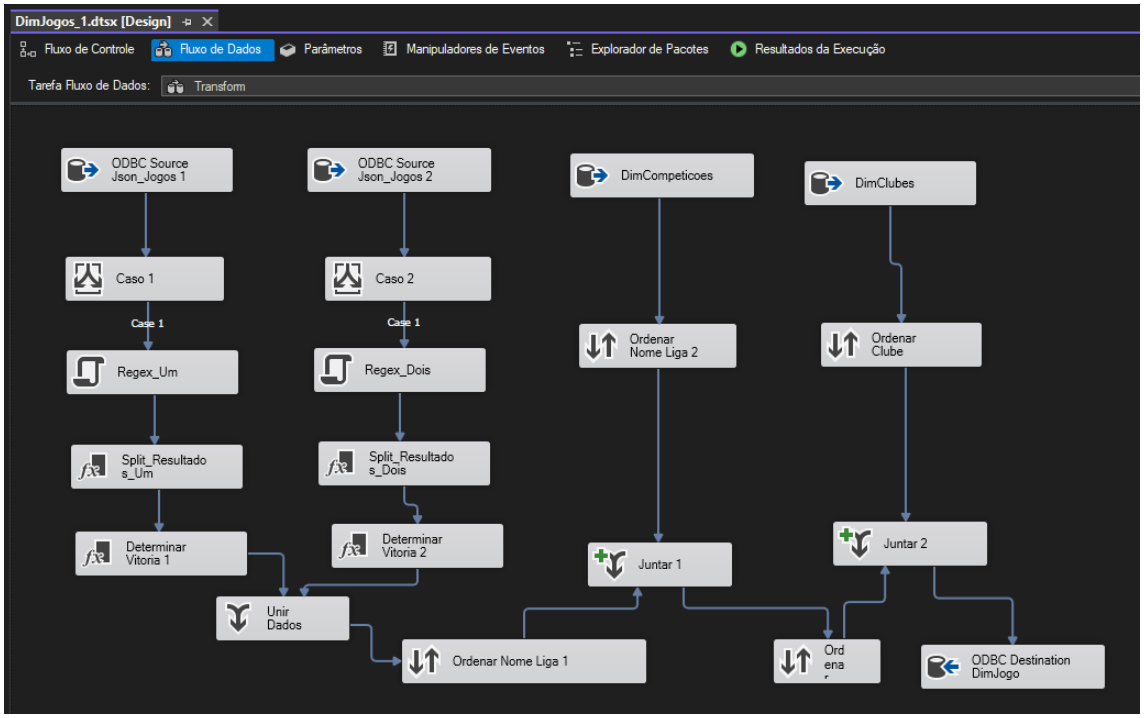

Figura 15 - Fluxo de Dados Job DimJogos

A figura anterior retrata o fluxo de dados do *package* Job DimJogos_1 onde é realizada a transformação dos dados. Os dados são carregados pela tabela Json_Jogos, este processo é realizado 2, por causa da forma como a estrutura de dados é apresentada. Segue um exemplo na seguinte figura:

NomeLiga	Jogo	Data	Equipa_Um	Equipa_Dois	Resultado	ID_VS
Premier League 2010/11	Matchday 1	2010-08-14	Bolton Wanderers FC	Fulham FC	[0, 0]	1
Premier League 2010/11	Matchday 1	2010-08-14	Wigan Athletic FC	Blackpool FC	[0, 4]	2

Figura 16 - Tabela DimJogos

Por uma questão de usabilidade e praticidade é necessário partir cada linha a meio, pois a estrutura DimJogo para criação de registo só aceita a entrada de um clube e de um resultado por registo, então como o *Microsoft SQL Server Integration Services* (MSSIS) não oferece uma forma

pratica para executar esta quebra de linha optei por chamar 2 vezes a tabela Json_Jogos e em cada chamada passava a equipa e o resultado pretendido a partir disto foi feita um caso (Caso 1/ Caso 2) em cada para verificar a existência de valores null no campo "NomeLiga", varios regex (Regex_Um/ Regex_Dois) para recolher o nome da liga, numero do jogo, a temporada em que se apresenta, e o resultado. Após o sucesso destes processos é feito um split do resultado onde é indicado o resultado de cada equipa, terminado com verificação se houve uma vitoria, empate, derrota. Por fim é feita a união dos dados, para identificar quais foram as equipas que se confrontaram foi adicionado um campo de auto incremento ao coletar os dados para a Json_Jogos para ser possível a relação entre estas linhas. Após a junção destes dados são feitos joins das tabelas DimCompeticoes e DimClubes para substitui o nome da liga e o nome do clube pelos seus respetivos ID´s, estes dados são depositados na base de dados *MySql* na tabela DimJogo.

g. Job ExeJobs

A seguinte figura retrata o fluxo de controlo do *package* Job ExeJobs.

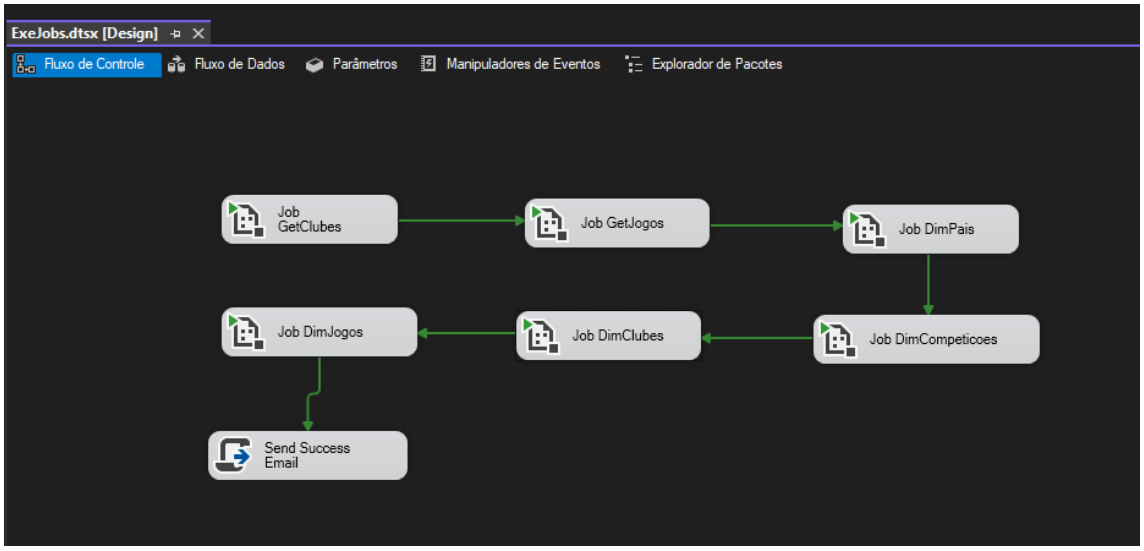

Figura 17 - Fluxo de Controle Job ExeJobs

O procedimento Task no presente *package* tem como função realizar sequencialmente todos os outros *packages* anteriormente apresentados, com sucesso de todas as execuções será enviado um email para o administrador com várias informações.

h. Power Bi

A seguinte figura retrata os dados trabalhados em Power BI.

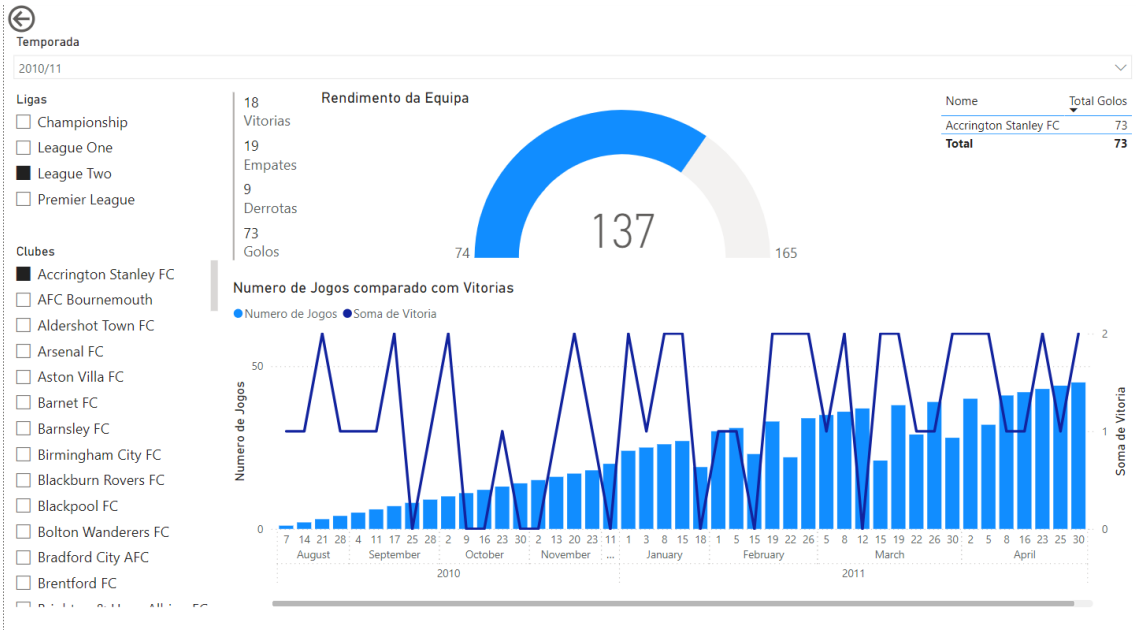

Figura 18 - Power Bi

Foram desenvolvidos gráficos e métricas para perceber o desempenho de cada equipa ao longo de casa temporada. O medidor tem como função indicar o desempenho da equipa e o Gráfico de linhas e de colunas empilhadas tem como função dar uma vista geral sobre as vitorias, empates e derrotas. Esta analise pode ser encontrada no ficheiro "Dados Analiticos.pbix".

Demostração em Vídeo

Para ajudar a ter uma melhor perceção do funcionamento, é possível visualizar através deste QrCode uma explicação extensiva do funcionamento do ETL. O vídeo explica todas os Jobs e transformações de cada *package.*

Conclusão e Trabalhos futuros

Em conclusão, este projeto de ETL desenvolvido com o SSIS cumpriu com sucesso, o objetivo de extrair dados de arquivos JSON, aplicar transformações complexas utilizando *Regex*, *joins* e condições *if*, e, finalmente, carregar os dados transformados numa base de dados *MySQL* através de *ODBC*. O processo garantiu a integração correta e eficiente dos dados, assegurando a sua qualidade e prontidão para análises. Com isso, conseguimos criar um fluxo de trabalho automatizado e otimizado, que pode ser facilmente adaptado para atender a novas necessidades de negócio e escalabilidade.

Bibliografia

Documentação SQL Server Integration Services - https://learn.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-ver16

Cozyroc - https://www.cozyroc.com/ssis/json

Power Bi - https://learn.microsoft.com/pt-pt/power-bi/