

IPCA

Escola Superior de Tecnologia



Relatório do Trabalho Prático Um Integração de Sistemas de Informação

Nuno Miguel Carvalho Araújo, nº 20078

Professor Óscar Ribeiro

2022/2023

Índice

1	Introdução	1
1.1	Motivação	1
1.2	Objetivos	1
2	Resumo do projeto	2
3	A solução implementada	3
3.1	C# com Expressões Regulares (ER)	3
3.2	Manipulação de dados com Pentaho Data Integration	6
3.3	Manipulação e apresentação de dados no node-red	8
4	Conclusão	10
5	Bibliografia	11

Índice de Figuras

1	Objetos C#	3
2	Expressão Regular	3
3	Leitura do Ficheiro CSV	4
4	Escrita do Ficheiro JSON	4
5	Conversão de CSV para JSON	5
6	Conversão de JSON para XML	6
7	Ordenação alfabética dos dados	6
8	Filtragem de dados por fundador	7
9	Aplicação de MQTT	7
10	Chips por fundador	8
11	Frequências dos chips	8
12	Inserção manual de chips	9

1 Introdução

Com este trabalho da Disciplina de Integração de Sistemas de Informação (ISI) pretende-se focar a aplicação e experimentação de ferramentas em processos de ETL (Extract, Transformation and Load), inerentes a processos de integração de sistemas de informação ao nível dos dados.

Pretende-se que sejam desenvolvidos processos de ETL que envolvam scripts próprias ou que recorram a ferramentas disponíveis como o Pentaho Kettle, Microsoft SQL Server Integration Services (MSSIS), Knime, Talend Open Studio ou outras. Ferramentas complementares como Node-RED ou HomeAssistant poderão, também, ser exploradas e integradas nos processos.

1.1 Motivação

Uma vez que os processos de negócio não param de se reconfigurar, as empresas enfrentam desafios constantes de análise e aquisição de novas soluções informáticas. A necessidade de rentabilizarem anteriores aquisições, tanto pelo investimento financeiro envolvido como pela dependência dos processos, dos dados, etc., que delas fazem parte, as empresas procuram formas capazes de analisarem mais-valias e menor impacto com as novas aquisições.

Processos que são exemplo de cenários onde processos ETL poderão desempenhar papel preponderante: auditorias a dados, processos, segurança, outros; migração e reorganização de dados; análise e processamento de dados (datamining, etc.); recomendações e previsões sobre estados com processamento recorrente a big data;

No contexto emergente das smart cities e smart environments, a evidência deste tipo de soluções é clara e crescente. A integração de soluções ditas mais inteligentes, que reagem a eventos, em processos existentes ou mesmo legados, são desafios constantes.

1.2 Objetivos

Os objetivos do desenvolvimento deste trabalho são os seguintes:

- Consolidar conceitos associados à Integração de Sistemas de Informação usando Dados;
- Analisar e especificar cenários de aplicação de processos de ETL;
- Explorar ferramentas de suporte a processos de ETL;
- Explorar novas Tecnologias, Frameworks ou Paradigmas;
- Potenciar a experiência no desenvolvimento de software;
- Facilitar a assimilação do conteúdo da Unidade Curricular.

2 Resumo do projeto

De forma a implementar os conceitos lecionados ao longo da Unidade Curricular de Integração de Sistemas de Informação, cumprindo com os objetivos propostos, decidi fazer a manipulação e apresentação de resultados de um *dataset* público.

Para começar, optei por escolher um *dataset* cujo tema fosse do meu agrado, de modo a tornar o desenvolvimento do projeto mais apelativo. Desta forma, escolhi um ficheiro no formato *CSV* que contém informação à cerca da performance de diferentes chips de CPU (processadores) e GPU (placas gráficas).

Em seguida, utilizei o C#, no Visual Studio, para abrir o *dataset*, fazer a sua leitura e o respetivo tratamento dos dados com suporte a uma Expressão Regular criada para o efeito. Após o tratamento de dados, foi desenvolvido um algoritmo para a criação e escrita de um ficheiro *JSON* com os dados pertinentes.

Posteriormente, utilizei o *Pentaho Data Integration* para executar as seguintes funções:

- Conversão do ficheiro *JSON* (input) em *XML* (output);
- Filtragem de dados do ficheiro *JSON* e armazenamento na respetiva base de dados;
- Captação de dados através do *mosquitto publisher*.

Por fim, utilizei o *node-red*, com o auxílio do *mosquitto subscriber* para tratamento e apresentação de dados através da dashboard. Desenvolvi, também, uma interface que permite ao utilizador a inserção e o armazenamento de chips na base de dados.

3 A solução implementada

3.1 C# com Expressões Regulares (ER)

Inicialmente criei um projeto em C#, no Visual Studio, com o intuito de abrir o *dataset*, fazer a sua leitura e tratar os dados, com o suporte de uma Expressão Regular criada para o efeito.

Posto isto, comecei por criar um objeto denominado de "*Vendor*", com o objetivo de armazenar os chips separados por vendedor. Em seguida, criei o objeto "*Chip*" com a intenção de armazenar todos os dados característicos de cada chip. Por fim, a cada vendedor está associada uma lista de "*Chips*".

```
8 referências
public class Vendor
{
    5 referências
    public string Name { get; set; }
    2 referências
    public List<Chip> Chips { get; set; }
}
5 referências
public class Chip
{
    1 referência
    public string Product { get; set; }
    1 referência
    public string Type { get; set; }
    1 referência
    public DateTime ReleaseDate { get; set; }
    2 referências
    public decimal ProcessSize { get; set; }
    2 referências
    public decimal TDP { get; set; }
    2 referências
    public decimal DieSize { get; set; }
    2 referências
    public decimal Transistors { get; set; }
    2 referências
    public decimal Frequency { get; set; }
    1 referência
    public string Foundry { get; set; }
    2 referências
    public decimal FP16 { get; set; }
    2 referências
    public decimal FP32 { get; set; }
    2 referências
    public decimal FP64 { get; set; }
}
```

Figura 1: Objetos C#

Em seguida, criei uma Expressão Regular (ER) capaz de interpretar cada linha do ficheiro *CSV*. De salientar que algumas das linhas não cumpriam os requisitos da ER, sendo, desta forma, descartadas, pois o interesse é armazenar apenas as linhas que possam ser convertidas para o objeto criado anteriormente.

```
@"[0-9]+,[^,]*,(GPU|CPU),[0-9]{4}-[0-9]{2}-[0-9]{2}([0-9]*[.]?[0-9]*)+,[^,]*,[^,]*([0-9]*[.]?[0-9]*)+"
```

Figura 2: Expressão Regular

Posteriormente, criei uma função capaz de fazer a leitura do ficheiro linha por linha. Enquanto o ficheiro é lido, é verificado se a linha cumpre com a Expressão Regular e, em caso afirmativo, as informações pertinentes são armazenadas nos respetivos campos do objeto. Como mencionado anteriormente, a linha é descartada caso não corresponda com a Expressão Regular. De realçar que foi necessário especificar a cultura como inglesa, pois os números decimais estavam separados por “.” e isso estava a criar conflito na leitura do ficheiro.

```
while ((line = reader.ReadLine()) != null)
{
    Match match = regex.Match(line);
    if (match.Success)
    {
        Vendor vendor = new Vendor();
        Chip chip = new Chip();
        string[] campos = match.Value.Split(',');

        chip.Product = campos[1];
        chip.Type = campos[2];
        chip.ReleaseDate = DateTime.Parse(campos[3]);
        if (decimal.TryParse(campos[4], style, culture, out decimal result4)) chip.ProcessSize = result4;
        else chip.ProcessSize = 0;
        if (decimal.TryParse(campos[5], style, culture, out decimal result5)) chip.TDP = result5;
        else chip.TDP = 0;
        if (decimal.TryParse(campos[6], style, culture, out decimal result6)) chip.DieSize = result6;
        else chip.DieSize = 0;
        if (decimal.TryParse(campos[7], style, culture, out decimal result7)) chip.Transistors = result7;
        else chip.Transistors = 0;
        if (decimal.TryParse(campos[8], style, culture, out decimal result8)) chip.Frequency = result8;
        else chip.Frequency = 0;
        chip.Foundry = campos[9];
        vendor.Name = campos[10];
        if (decimal.TryParse(campos[11], style, culture, out decimal result11)) chip.FP16 = result11;
        else chip.FP16 = 0;
        if (decimal.TryParse(campos[12], style, culture, out decimal result12)) chip.FP32 = result12;
        else chip.FP32 = 0;
        if (decimal.TryParse(campos[13], style, culture, out decimal result13)) chip.FP64 = result13;
        else chip.FP64 = 0;

        if (Vendors.Any(x => x.Name == vendor.Name))
        {
            int position = Vendors.FindIndex(p => p.Name == vendor.Name);
            Vendors[position].Chips.Add(chip);
        }
        else
        {
            List<Chip> Chips = new List<Chip>();
            Chips.Add(chip);
            vendor.Chips = Chips;
            Vendors.Add(vendor);
        }
    }
}
```

Figura 3: Leitura do Ficheiro CSV

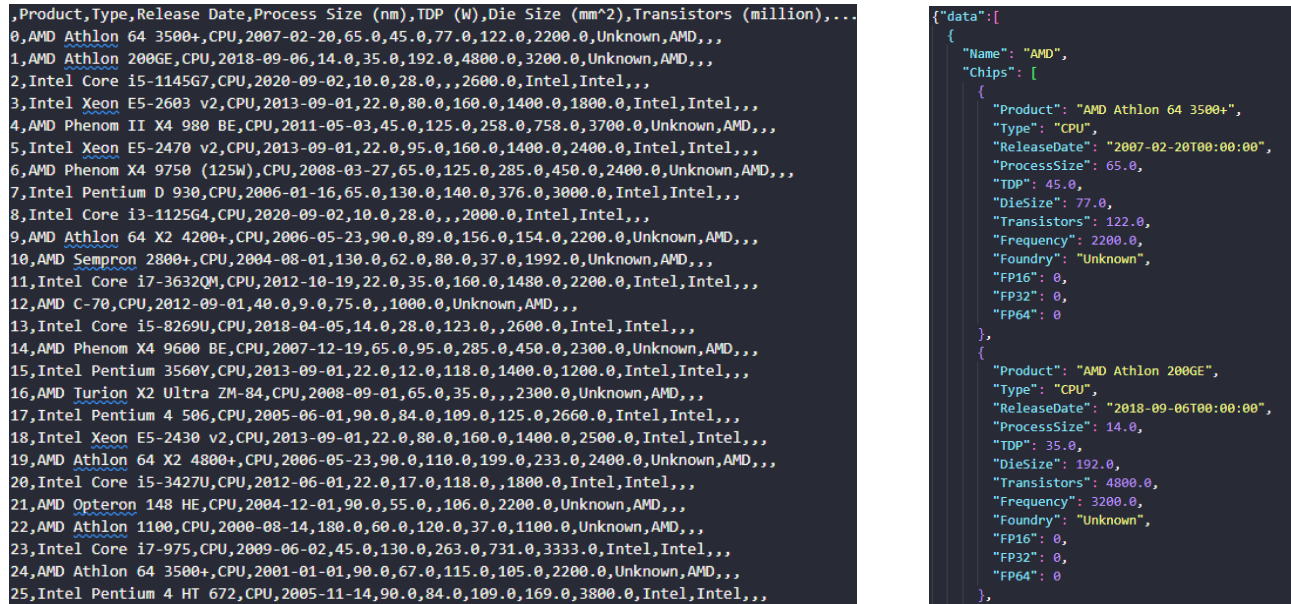
E, para concluir as manipulações em C#, criei um algoritmo responsável pela exportação dos dados para um ficheiro *JSON*. Para isso, utilizei o método de serialização disponível no IDE.

```
static void writeJSON(List<Vendor> Vendors)
{
    string file = @"../../Kettle/chip_dataset.json";

    var opt = new JsonSerializerOptions() { WriteIndented = true };
    string jsonString = JsonSerializer.Serialize<IList<Vendor>>(Vendors, opt);
    try{ File.WriteAllText(file, $"{\"data\": \" + jsonString + \"\"}); }
    catch { Console.WriteLine(\"\\nPath not found\"); }
    Console.WriteLine(\"\\nJSON file successfully written!\\n\");
}
```

Figura 4: Escrita do Ficheiro JSON

Na imagem seguinte está exposto o resultado da conversão entre o ficheiro de entrada em *CSV* e o ficheiro de saída em *JSON*.



```
,Product,Type,Release Date,Process Size (nm),TDP (W),Die Size (mm^2),Transistors (million),...
0,AMD Athlon 64 3500+,CPU,2007-02-20,65.0,45.0,77.0,122.0,2200.0,Unknown,AMD,,,
1,AMD Athlon 200GE,CPU,2018-09-06,14.0,35.0,192.0,4800.0,3200.0,Unknown,AMD,,,
2,Intel Core i5-1145G7,CPU,2020-09-02,10.0,28.0,,,2600.0,Intel,Intel,,,
3,Intel Xeon E5-2603 v2,CPU,2013-09-01,22.0,80.0,160.0,1400.0,1800.0,Intel,Intel,,,
4,AMD Phenom II X4 980 BE,CPU,2011-05-03,45.0,125.0,258.0,758.0,3700.0,Unknown,AMD,,,
5,Intel Xeon E5-2470 v2,CPU,2013-09-01,22.0,95.0,160.0,1400.0,2400.0,Intel,Intel,,,
6,AMD Phenom X4 9750 (125W),CPU,2008-03-27,65.0,125.0,285.0,450.0,2400.0,Unknown,AMD,,,
7,Intel Pentium D 930,CPU,2006-01-16,65.0,130.0,140.0,376.0,3000.0,Intel,Intel,,,
8,Intel Core i3-1125G4,CPU,2020-09-02,10.0,28.0,,,2000.0,Intel,Intel,,,
9,AMD Athlon 64 X2 4200+,CPU,2006-05-23,90.0,89.0,156.0,154.0,2200.0,Unknown,AMD,,,
10,AMD Sempron 2800+,CPU,2004-08-01,130.0,62.0,80.0,37.0,1992.0,Unknown,AMD,,,
11,Intel Core i7-3632QM,CPU,2012-10-19,22.0,35.0,160.0,1480.0,2200.0,Intel,Intel,,,
12,AMD C-70,CPU,2012-09-01,40.0,9.0,75.0,,1000.0,Unknown,AMD,,,
13,Intel Core i5-8269U,CPU,2018-04-05,14.0,28.0,123.0,,2600.0,Intel,Intel,,,
14,AMD Phenom X4 9600 BE,CPU,2007-12-19,65.0,95.0,285.0,450.0,2300.0,Unknown,AMD,,,
15,Intel Pentium 3560Y,CPU,2013-09-01,22.0,12.0,118.0,1400.0,1200.0,Intel,Intel,,,
16,AMD Turion X2 Ultra ZM-84,CPU,2008-09-01,65.0,35.0,,2300.0,Unknown,AMD,,,
17,Intel Pentium 4 506,CPU,2005-06-01,90.0,84.0,109.0,125.0,2660.0,Intel,Intel,,,
18,Intel Xeon E5-2430 v2,CPU,2013-09-01,22.0,80.0,160.0,1400.0,2500.0,Intel,Intel,,,
19,AMD Athlon 64 X2 4800+,CPU,2006-05-23,90.0,110.0,199.0,233.0,2400.0,Unknown,AMD,,,
20,Intel Core i5-3427U,CPU,2012-06-01,22.0,17.0,118.0,,1800.0,Intel,Intel,,,
21,AMD Opteron 148 HE,CPU,2004-12-01,90.0,55.0,,106.0,2200.0,Unknown,AMD,,,
22,AMD Athlon 1100,CPU,2000-08-14,180.0,60.0,120.0,37.0,1100.0,Unknown,AMD,,,
23,Intel Core i7-975,CPU,2009-06-02,45.0,130.0,263.0,731.0,3333.0,Intel,Intel,,,
24,AMD Athlon 64 3500+,CPU,2001-01-01,90.0,67.0,115.0,105.0,2200.0,Unknown,AMD,,,
25,Intel Pentium 4 HT 672,CPU,2005-11-14,90.0,84.0,109.0,169.0,3800.0,Intel,Intel,,,
{"data":{
{
  "Name": "AMD",
  "Chips": [
    {
      "Product": "AMD Athlon 64 3500+",
      "Type": "CPU",
      "ReleaseDate": "2007-02-20T00:00:00",
      "ProcessSize": 65.0,
      "TDP": 45.0,
      "DieSize": 77.0,
      "Transistors": 122.0,
      "Frequency": 2200.0,
      "Foundry": "Unknown",
      "FP16": 0,
      "FP32": 0,
      "FP64": 0
    },
    {
      "Product": "AMD Athlon 200GE",
      "Type": "CPU",
      "ReleaseDate": "2018-09-06T00:00:00",
      "ProcessSize": 14.0,
      "TDP": 35.0,
      "DieSize": 192.0,
      "Transistors": 4800.0,
      "Frequency": 3200.0,
      "Foundry": "Unknown",
      "FP16": 0,
      "FP32": 0,
      "FP64": 0
    }
  ]
}
```

Figura 5: Conversão de CSV para JSON

3.2 Manipulação de dados com Pentaho Data Integration

Na parte de manipulação de dados com recurso ao software *Pentaho Data Integration*, comecei por pegar no ficheiro *JSON* resultante da manipulação em C# anteriormente retratada e utiliza-lo como *input* no kettle.

A primeira transformação a ser feita foi a conversão de *JSON* para *XML*, para isso executei uma conversão direta, ligando um *XML output* ao *JSON input*.

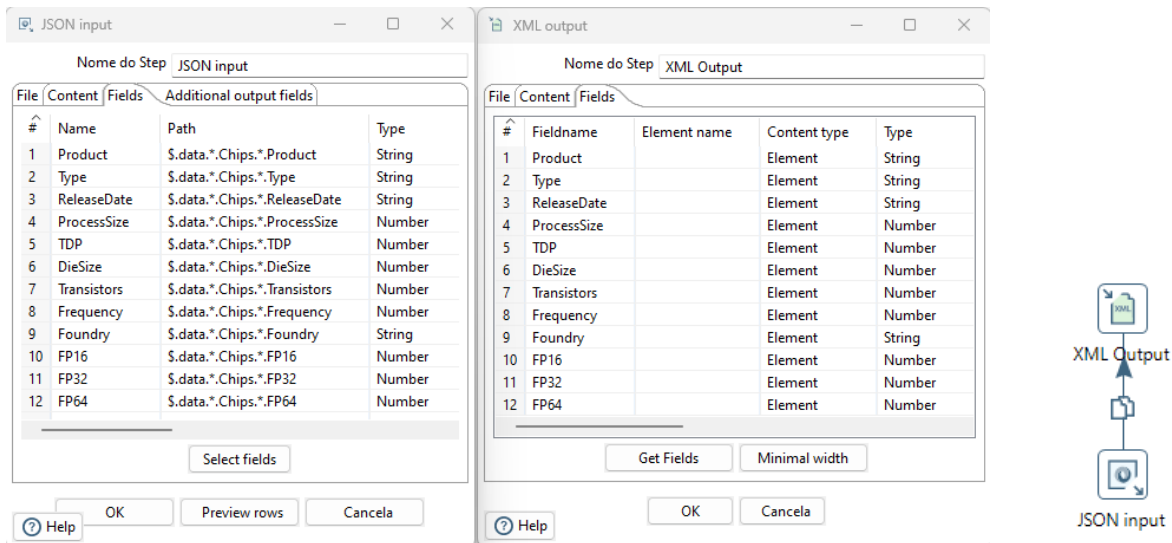


Figura 6: Conversão de JSON para XML

Em seguida, criei um método responsável por ordenar alfabeticamente todos os elementos contidos no ficheiro de entrada.

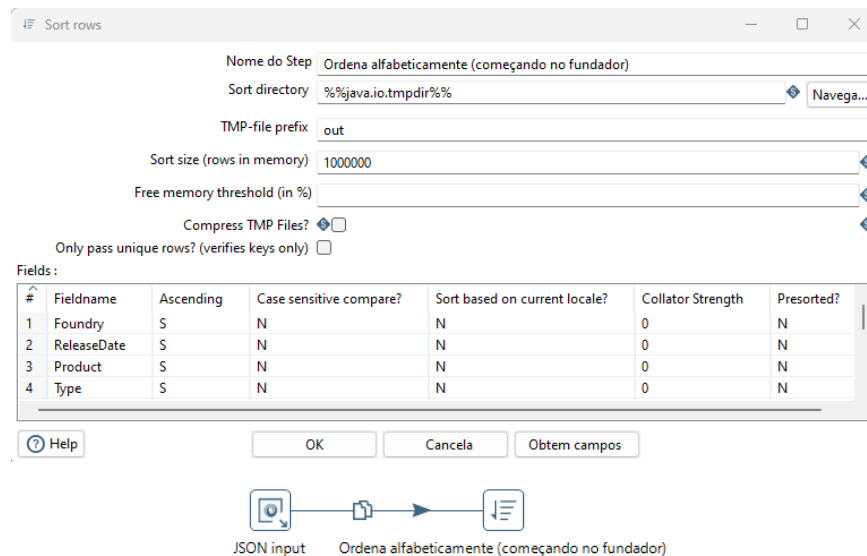


Figura 7: Ordenação alfabética dos dados

Após a ordenação alfabética dos dados, procedi à filtragem dos chips pelo campo *Foundry* e fiz o armazenamento na base de dados respectiva ao fundador em questão.

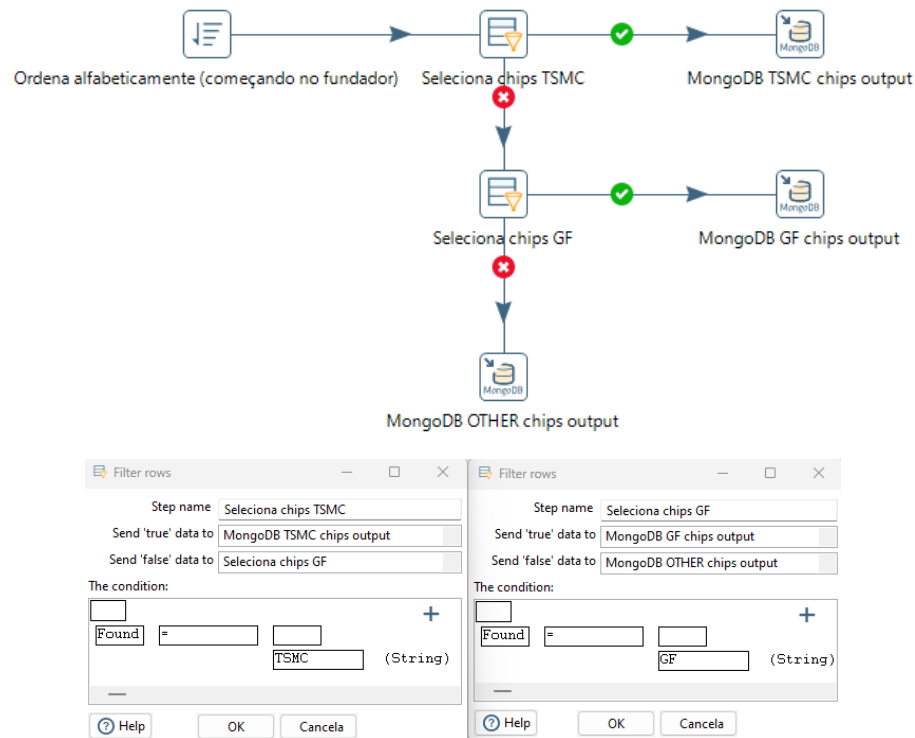


Figura 8: Filtragem de dados por fundador

Por fim, acrescentei um *MQTT publisher*, com o objetivo de intercetar os dados enviados para a base de dados e, juntamente com um *MQTT subscriber*, obter esses dados no *node-red*.

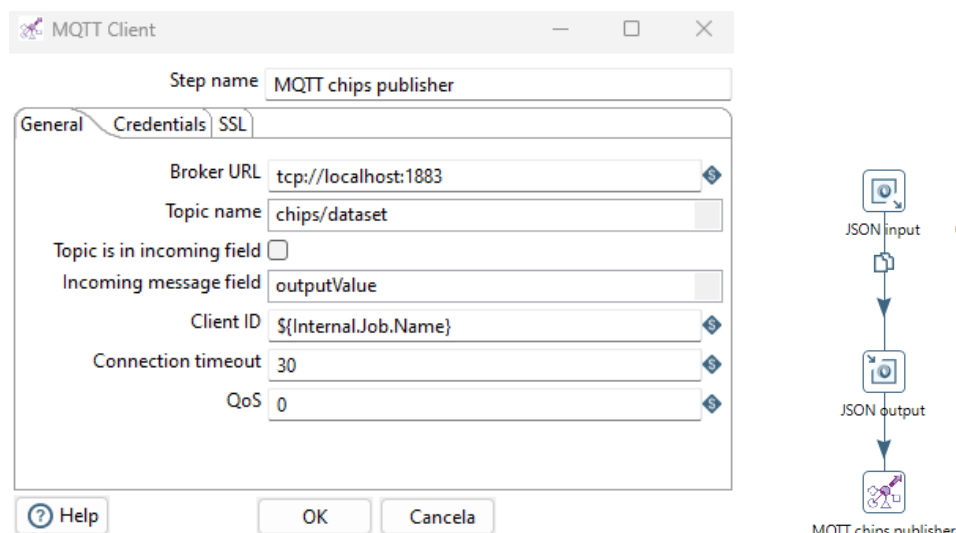


Figura 9: Aplicação de MQTT

3.3 Manipulação e apresentação de dados no node-red

A ultima ferramenta explorada foi o node-red e a sua dashboard.

Comecei por inserir um *MQTT subscriber* para obter os dados enviados com o *MQTT publisher* através do kettle.

Com esses dados, decidi criar duas funções distintas para apresentação de resultados ao utilizador.

A primeira função faz a contagem dos chips pelo campo *Foundry* e apresenta um gráfico de barras com essa informação.

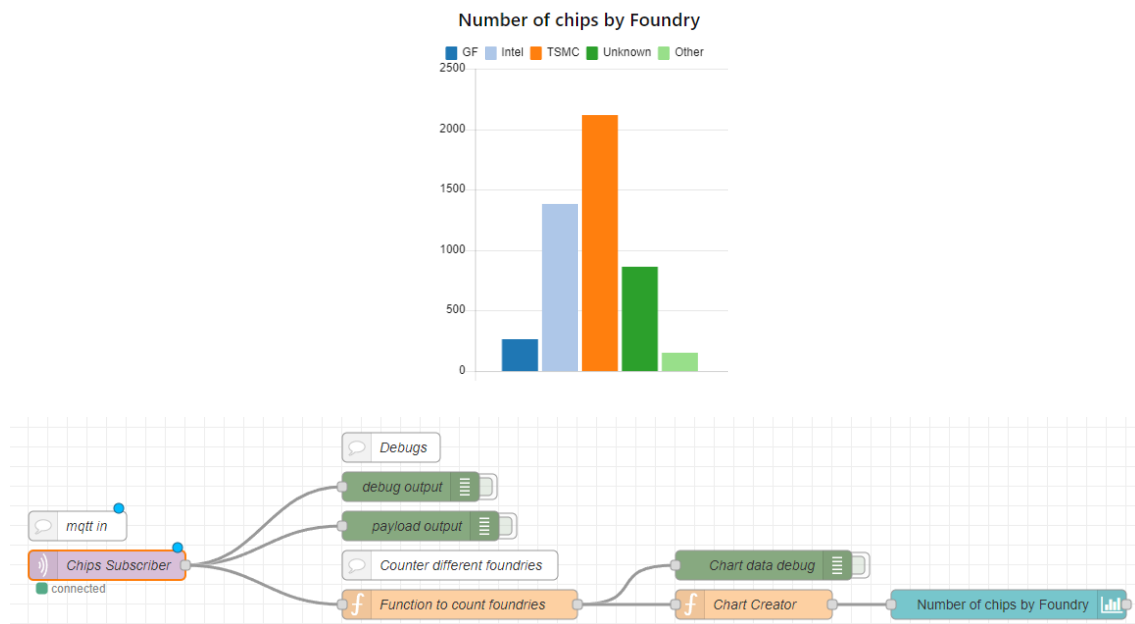


Figura 10: Chips por fundador

A segunda função deteta qual a maior, a menor e a média de frequências dos chips.

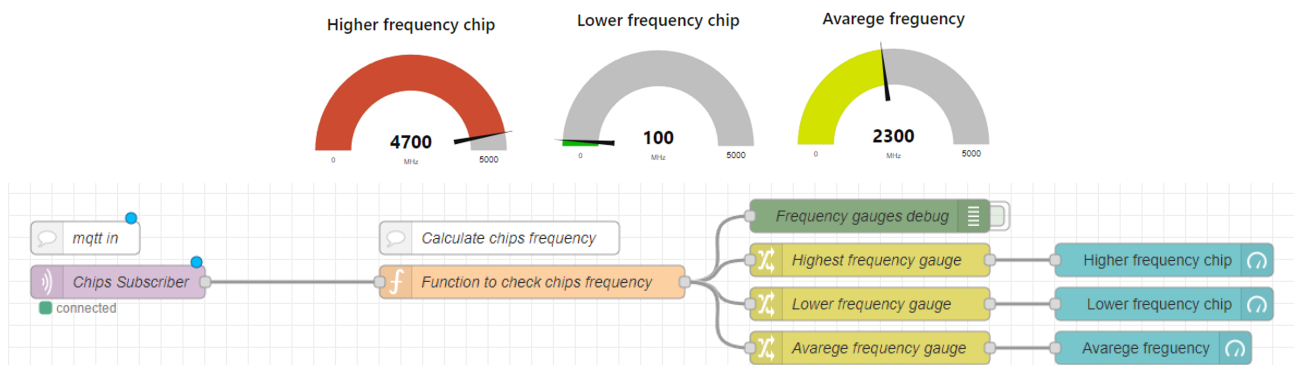


Figura 11: Frequências dos chips

Por fim, criei um interface que permite ao utilizador inserir chips manualmente, através da dashboard do node-red. Esses chips são armazenados diretamente na base de dados, mais concretamente na *collection* **ChipsUser**. De salientar que estes chips não constam no gráfico de barras anteriormente mencionado, pois o objetivo desse gráfico é mostrar apenas os dados originais de *input*.

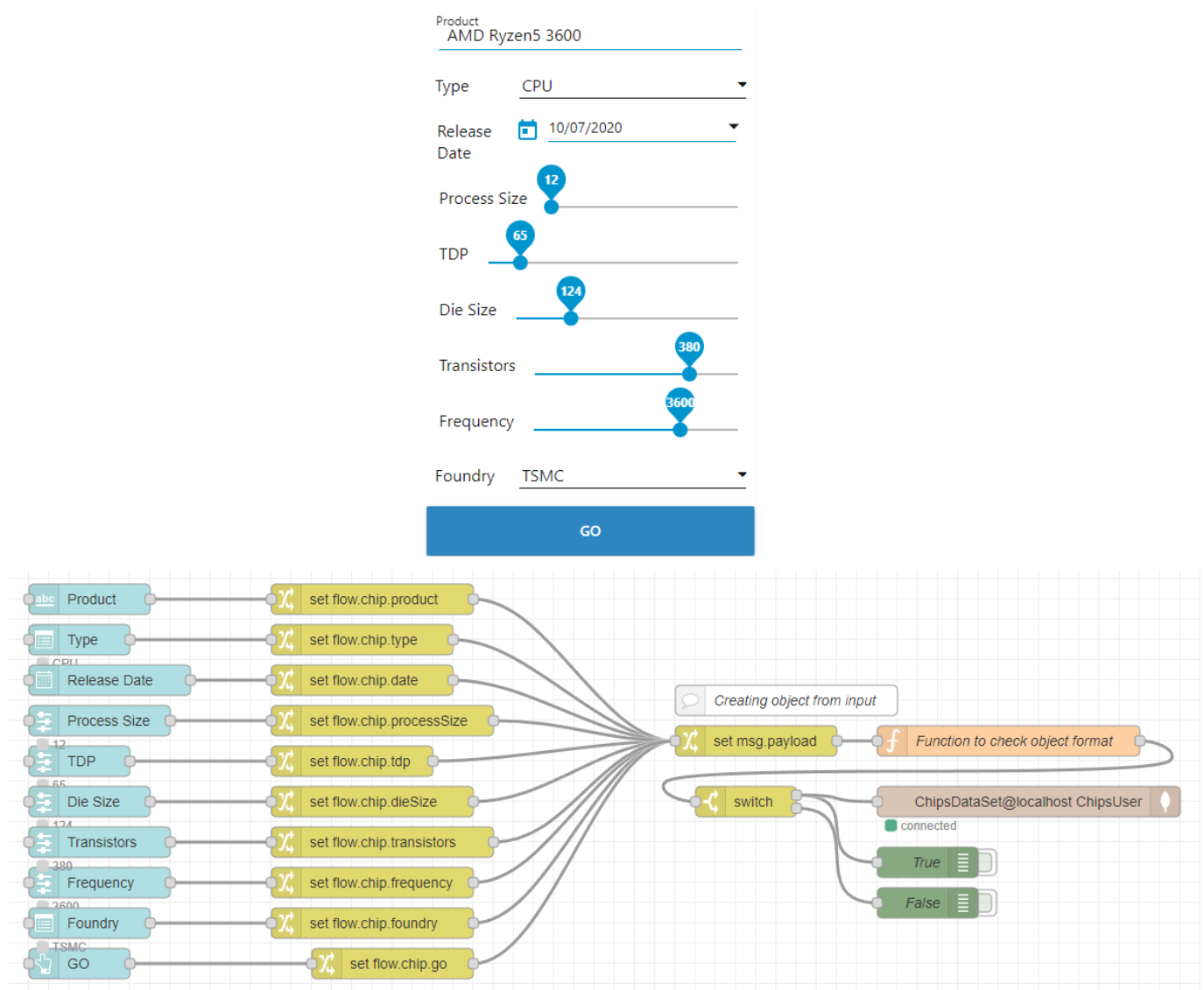


Figura 12: Inserção manual de chips

4 Conclusão

Na parte da leitura e tratamento de dados em C#, o maior obstáculo foi a identificação da expressão regular a utilizar. De salientar, também, o problema com os números decimais separados por ".", pois o algoritmo parecia bem implementado, no entanto, o output não estava correto e demorei algum tempo a identificar o problema.

Em relação à manipulação dos dados com o software *Pentaho Data Integration*, a maior dificuldade foi na correta formatação do *path* do ficheiro *JSON input*.

Quanto à manipulação e apresentação de dados no node-red, ocorreu a normal complexidade no desenvolvimento das funções que implementei. No entanto, o maior problema foi na criação da interface para inserção manual dos chips, pois senti dificuldades na junção dos dados vindos de campos separados.

De uma forma geral, concluo que a estrutura do trabalho implementada foi útil para a compreensão e interiorização dos conteúdos lecionados, até ao momento, na unidade curricular de Integração de Sistemas de Informação.

5 Bibliografia

- <https://www.kaggle.com/datasets/michaelbryantds/cpu-and-gpu-product-data>
- <https://regex101.com/>
- <https://flows.nodered.org/>