Universidade do Minho
Escola de Engenharia
Departamento de Informática

Carlos Miguel Lopes Sá Barbosa

Meta Data Migrator

Dezembro de 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Carlos Miguel Lopes Sá Barbosa

Meta Data Migrator

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
José Carlos Ramalho

Dezembro de 2021

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

AGRADECIMENTOS

Neste capítulo irei agradecer a toda a gente que contribuiu e que me acompanhou nesta jornada:

- Ao meu orientador, o professor José Carlos Ramalho, pelos conselhos que me deu durante o desenvolvimento deste projeto, bem como as lições de vida;
- À minha família, pelo amor e apoio que me têm dado desde o momento que nasci até agora;
- À minha namorada e aos meus amigos, por estarem sempre do meu lado e por me terem ajudado nos momentos mais difíceis;
- A toda a gente que conheci no meu percurso académico, porque de uma maneira ou de outra, aprendi sempre algo novo.

RESUMO

Atualmente existem empresas que possuem informação bastante valiosa, sendo necessário que haja uma boa gestão dos seus dados. Os sistemas de armazenamento têm um ciclo de vida, o que faz com que seja necessária a sua atualização periódica. Sendo assim, o processo de migração tornou-se fulcral para a atualização dos repositórios das empresas.

Com o passar dos anos, houve a necessidade de automatizar o processo de migração de modo a torná-lo mais rápido e confiável. A partir desta necessidade foi proposto o desenvolvimento uma ferramenta que, a partir de um mapeamento especificado por um utilizador, seja capaz de criar um conversor que realize uma migração.

Para ajudar ao desenvolvimento da ferramenta, foram propostos subobjetivos. Utilizando o objetivo principal como referência, foram efetuados vários estudos sobre boas práticas para a realização de um processo de migração, bem como abordagens existentes.

Após os estudos efetuados, desenvolveu-se a ferramenta de acordo com os subobjetivos propostos. De seguida, o trabalho realizado foi validado com um caso de estudo real.

Na conclusão, reflete-se acerca do trabalho realizado, das modificações necessárias e de possíveis adições.

PALAVRAS-CHAVE Migração, Especificação, Transformação, Mapeamento, *Web*, *CSV*

ABSTRACT

Currently there are companies that possess very valuable information, which makes it necessary to possess a very efficient management of their data. The storage systems have a life cycle, making it is necessary to update the systems from time to time. Therefore, the migration process became important for the update of the companies' repositories.

Over the years, there was a necessity to automate the migration process in order to make it faster and more reliable. From this need it was proposed to develop a tool that, from a mapping specified by an user, is capable of creating a converter that performs a migration.

To help the development of the tool, sub-objectives were proposed. Using the main objetive as a reference, several studies were carried out in terms of the best practices for carrying out a migration process, as well as existing approaches.

After the studies, the tool was developed in accordance with the proposed sub-objectives. Then, the tool was validated with a real case study.

In the conclusion, it reflects on the work done until this point, the necessary modifications and possible additions.

KEYWORDS Migration, Specification, Transformation, Mapping, *Web*, CSV

CONTEÚDO

LISTA DE FIGURAS

1

INTRODUÇÃO

Neste capítulo será apresentado o contexto, onde se explica a necessidade da realização de migrações. De seguida segue-se a motivação, onde se justifica o porquê de automatizar o processo de migração. Depois serão apresentados os objetivos, bem como a metodologia que se utilizará. Por fim, será exposta a estrutura da dissertação.

1.1 CONTEXTO

Desde o início da humanidade, o Homem tentou arranjar maneiras de armazenar informação, desde desenhar símbolos em paredes, até escrever livros. Atualmente, com o avanço da tecnologia, diversas maneiras de guardar informação foram substituídas, tornando-se assim obsoletas. Sendo assim, foram criadas novas maneiras mais práticas de guardar informação.

Uma destas maneiras que surgiu foi armazenar informação em ficheiros, os quais podem ser guardados em vários tipos de dispositivos (computadores, telemóveis, servidores, entre outros).

Com isto, a migração de dados passou a ser uma tarefa cada vez mais necessária devido ao crescimento dos sistemas de informação e à evolução tecnológica que não para, tornando muitos sistemas obsoletos. Logo, é necessária a migração de um tipo de dados para outro tipo mais utilizado.

A migração é uma tarefa complexa, devido ao facto de estar a lidar com vários tipos de dados diferentes e ser necessário converter para o formato desejado. Normalmente gastam-se bastantes recursos no projeto em que a migração estiver inserida, devido ao facto de ser necessário um estudo intensivo das estruturas de dados que estão a ser utilizadas e das estruturas de dados que vão ser implementadas.

1.2 MOTIVAÇÃO

A informação tem vindo a tornar-se cada vez mais importante, sendo essencial fazer uma boa gestão dela, permitindo assim melhorar a organização e da realização de tarefas. Esta informação tem vindo a acumular-se nos repositórios das empresas. Estes têm um ciclo de vida de, aproximadamente, cinco anos, havendo assim a necessidade de fazer a atualização dos sistemas.

O processo de migração inicialmente era manual, o que o tornava moroso. Tendo em conta esta situação, bem como a necessidade de se realizar o processo mais regularmente, decidiu-se automatizar para o tornar mais rápido. Outras vantagens relevantes da automatização são a redução de erros devido ao *input* conter formatos inválidos, a redução do erro humano e a redução de custos associados.

O grau de automatização tem vindo a aumentar com os últimos anos, permitindo ao utilizador ter acesso a um vasto leque de ferramentas que simplificam bastante o processo de migração, sendo atualmente essenciais para um bom migrador de dados.

1.3 OBJETIVOS

Esta dissertação tem como propósito criar uma ferramenta que, a partir de um mapeamento especificado por um dado utilizador, seja capaz de gerar um conversor que realize uma migração.

Para se cumprir este objetivo, foram criados os seguintes subobjetivos:

- Desenvolvimento de uma Linguagem que permita ao utilizador especificar mapeamentos;
- Criação de um *Parser* para a Linguagem;
- Enriquecimento do *Parser* com ações semânticas;
- Criação de uma aplicação *Web* que integre todas as ferramentas desenvolvidas, e as disponibilize ao utilizador numa *interface* amigável.

1.4 METODOLOGIA

Para a realização desta dissertação, será utilizada a seguinte metodologia:

- Estudo teórico acerca de boas práticas para a realização de um processo de migração;
- Estudo teórico acerca de abordagens existentes para processos de migração;
- Desenvolvimento de um migrador;
- Desenvolvimento e implementação de uma aplicação *Web*.

Para além disso, para se manter o ritmo de trabalho, serão realizadas reuniões semanais com o orientador.

1.5 ESTRUTURA DO DOCUMENTO

Neste documento apresenta-se o estado da arte, onde se descreve um processo de migração que obedece a boas práticas, bem como se explica algumas abordagens existentes.

Após o estado da arte, é explicada a abordagem tomada e as decisões que foram tomadas para realizar o projeto.

De seguida, o migrador é explicado a partir dos subobjetivos apresentados anteriormente, desde a criação da linguagem (onde se explicará a sintaxe para a especificação das transformações), a criação do *parser* dessa linguagem (onde se apresentará uma framework que possibilita gerar parsers em *JavaScript* (JS)), a incorporação das ações semânticas e por fim a aplicação *Web* (onde se apresentará as tecnologias que foram utilizadas para desenvolver o *backend* e *frontend*).

Para além disso, é apresentado o caso de estudo, onde se apresenta o objetivo da migração, bem como os resultados obtidos.

Por fim, termina-se o documento com a conclusão, onde se conclui acerca do trabalho realizado, bem como algumas reflexões acerca do trabalho futuro e da importância do projeto.

2

ESTADO DA ARTE

Este capítulo apresenta as boas práticas para a realização de um processo de migração, bem como algumas abordagens semelhantes já existentes.

2.1 CICLO DE VIDA DE UM PROCESSO DE MIGRAÇÃO

Atualmente existem empresas que possuem repositórios digitais que se encontram operacionais há muitos anos. Muitos desses repositórios não acompanharam a evolução tecnológica, o que faz com que tenham menos armazenamento ou performance em relação a sistemas mais atualizados. Para além disso, o custo para os manter é elevado, pois é fulcral recorrer aos poucos serviços disponíveis que fornecem o suporte necessário para manter o sistema atual.

Tendo em conta a situação anterior, muitas empresas optam pela realização de migração de dados para um sistema mais atualizado. No entanto, como existe muita informação a ser gerida, é necessário ter o devido cuidado, para que não ocorram contratempos, onde se destacam os seguintes:

- A extração da informação é bastante complexa devido ao facto de existir documentação incompleta ou inexistente ou a falta da funcionalidade de exportação de dados;
- As estruturas de dados de origem e de destino são incompatíveis;
- O processo de limpeza de dados durante o processo pode causar erros;
- O desenvolvimento e automatização de processos de validação de dados podem ser muito difíceis de se desenvolver.

A partir deste pressuposto, Ferreira, Ramalho e Faria (Ferreira et al., 2013) fazem uma síntese do que deve ser um processo de migração. Este trabalho resultou da análise e fusão das melhores práticas em utilização em grandes instituições públicas e privadas europeias. Sendo assim, foi proposto que um processo de migração seja organizado nas seguintes etapas:

- Análise e consulta: Fazer uma análise dos sistemas/repositórios atuais, bem como as características da informação que está a ser armazenada e as necessidades de ambos os lados, de forma a definir a melhor estratégia para migrar e formalizar os requisitos necessários;
- Planificação e design: Seguir a definição dos requisitos da fase anterior, bem como desenvolver todas as especificações necessárias que levará à elaboração das rotinas de migração e de processos de teste;
- Desenvolvimento: Elaborar ferramentas para a migração de acordo com as especificações, bem como a elaboração das rotinas de migração e de processos de teste;
- Setup e Fase de testes: Criar um ambiente de teste usando o sistema/repositório de onde se quer migrar os dados e a ferramenta de migração que será utilizada. De modo a garantir a fiabilidade da ferramenta, deve-se fazer testes a todos os aspetos da informação;
- Execução das Migrações: Executar as rotinas de migração e migrar toda a informação para a nova infraestrutura;
- Validação: Testar e analisar a nova infraestrutura recém-populada, verificando se existem incoerências. Também começa o processo de documentação do sistema de migração criado;
- Término: É aconselhado treinar funcionários para que consigam lidar com a nova infraestrutura, bem como realizar manutenções e testes sistemáticos.

Figura 1: Esquema representativo do processo proposto

Neste trabalho, propõe-se uma ferramenta de suporte que permita a automatização de parte ou da totalidade das fases da planificação e design e de desenvolvimento.

2.2 ABORDAGENS EXISTENTES

Foram estudadas algumas abordagens que realizam a automatização do processo de migração. A seguir apresentam-se aquelas que foram consideradas mais relevantes para o trabalho desenvolvido.

2.2.1 *CSV ANNOTATION*

Uma das abordagem que foi encontrada denomina-se de *CSV-ANNOTATION* (Mahmud et al., 2018). Tem como objetivo converter ficheiros *CSV* em tabelas anotadas, de modo a que os ficheiros *CSV* estejam preparados para serem utilizados em aplicações *Web*.

Nesta abordagem foi criado um algoritmo para realizar o processamento do ficheiro. No entanto, este possui três sub-algoritmos para realizar a anotação da informação, porque realiza a anotação em termos de filas, de colunas e de células.

CSV File Parsing

O algoritmo que permite o processamento recebe um ficheiro *CSV* e um ficheiro em *JSON* com metadados onde é especificado o processo de migração. O ficheiro com metadados possui informação acerca dos dados que são importantes para as anotações.

Realiza-se um pré-processamento do ficheiro, onde ocorre uma limpeza dos dados. Por fim, utilizam-se os sub-algoritmos para a obtenção das anotações. Os sub-algoritmos ocorrem da seguinte forma:

- *Column Annotation Algorithm*: Neste algoritmo cria-se uma lista a partir dos cabeçalhos das colunas e, a partir de cada coluna, obtém-se os elementos desta. Por fim, realizam-se as anotações para cada elemento;
- *Row Annotation Algorithm*: Neste algoritmo cria-se uma lista a representar cada linha e obtém-se os elementos de cada uma delas. Por fim, realizam-se as anotações para cada elemento;
- *Cell Annotation Algorithm*: Neste algoritmo cria-se uma lista, começando na primeira linha, primeira coluna até à última linha, última coluna, realizando-se anotações para cada célula.

2.2.2 *Building a generic csv writer/reader using reflection*

Outra abordagem encontra-se no caso do *Building a generic csv writer/reader using reflection* (Aykanat, 2018). Nesta abordagem utiliza-se objetos em *Java* (JAV) para representar e guardar informação de ficheiros *CSV*.

O seguinte ficheiro será o caso de estudo para esta implementação, guardando dados de vários indivíduos, que contem as seguintes informações:

- O ID do indivíduo;
- O primeiro nome do indivíduo;
- O apelido do indivíduo.

```
id,name,lastname,
1,murat,aykanat,
2,john,smith,
```

Figura 2: Conteúdo do ficheiro exemplificativo

Cada objeto corresponde a um indivíduo. Os dados do indivíduo serão representados da seguinte forma:

- O ID será o *integer* Id;
- O primeiro nome será a *string* Name;
- O apelido será a *string* LastName.

Para se realizar o processamento dos dados assume-se que cada linha representa um indivíduo, logo a partir dessa linha, ao separar a string por vírgulas obtém-se a informação pretendida. Para voltar a inserir no ficheiro *CSV*, basta juntar a informação toda numa string e separá-la por vírgulas (para separar por campos) e por newlines (para separar as entradas). Sendo assim, criaram-se 2 funções:

- A função ToCSV, que transforma a informação de um objeto em string;
- A função AssignValuesFromCsv, que transforma a informação do ficheiro num objeto.

Posto isto, existe maneira de reverter cada ação que seja realizada, como se pode mostrar no esquema seguinte:

```
ficheiro CSV -> AssignValuesFromCsv() -> Objeto
Objeto -> toCSV() -> ficheiro CSV
```

2.2.3 *Tomassetti*

A abordagem de *Tomassetti* (Tomassetti, 2018) consiste no facto de haver a necessidade de lidar com muita informação e, dependendo das situações, pode levar a contratempos, tais como arranjar forma de fazer o processamento da informação sem qualquer perda.

Por isso, é proposta uma metodologia para desenvolver um serviço capaz de migrar para qualquer formato desejado. Neste caso, vai ser demonstrado um caso: um serviço que migre de *CSV* e que seja capaz de devolver em *JSON*.

Para este caso vai-se fazer uso de uma tecnologia chamada de *ANTLR* (Parr, 1989), que serve para gerar parsers que processam texto estruturado, utilizando notação de BNF-estendida para desenvolver gramáticas livres de contexto.

Usando o *ANTLR*, pode-se realizar processamento de texto para obter as informações pretendidas. Para interpretar cada produção das gramáticas, o *ANTLR* permite definir os ações semânticas para cada produção, usando *visitors*. A partir deles, consegue-se atribuir a informação a uma estrutura de dados, neste caso uma classe. Para se obter a informação, existe uma *API* que permite realizar pedidos HTTP aos objetos guardados, havendo assim o seguinte conjunto de rotas:

- *GET* para consultar informação (para consultar a lista toda e para consultar uma entrada específica);
- *POST* para inserir informação;
- *DELETE* para eliminar informação.

Todas estas rotas devolvem a informação num objeto em *JSON*.

2.2.4 *Migrate Source CSV*

O *Drupal* (DRU) é uma *framework open-source* desenvolvida em PHP (PHP) que permite gerir o conteúdo de um website, de forma a evitar a modificação de estruturas e componentes que se encontram implementadas.

Esta *framework* contém inúmeros módulos criados pela comunidade, tais como um módulo para o *Google Analytics* e o *Pathauto Module*, mas o módulo que vai ser exposto nesta dissertação será o *Migrate Source CSV*, que se encontra disponível a partir do *Drupal 8*.

Este módulo permite migrar ficheiros *CSV* para uma estrutura que o *Drupal* consiga interpretar. Para realizar uma migração, recorre-se a um ficheiro *YAML* para especificar as transformações. O processo de migração divide-se em 3 fases: a obtenção da informação, a manipulação dos dados e a submissão dos dados.

Obtenção da informação

Nesta fase do processo o utilizador fornece os dados que serão necessários para a migração.

Dentro do ficheiro *YAML*, utiliza-se o atributo source para identificar esta fase. Este atributo consiste num dicionário onde se especifica todos os dados necessários, entre os quais o tipo do ficheiro de entrada (utiliza-se o atributo *plugin*), a sua localização (utiliza-se o atributo *path*) e uma lista contendo a informação sobre as colunas que serão utilizadas (utiliza-se o atributo *column_names*).

A representação de cada coluna na lista descrita realiza-se da seguinte forma: cria-se um par chave-valor, onde a chave é o número da posição da coluna dentro do ficheiro de entrada (a primeira coluna considera-se que se encontra na posição zero) e o valor é outro par chave-valor, onde a chave é o nome que o programa irá utilizar e o valor é um nome *user friendly*.

De seguida encontra-se um excerto de como o source se pode encontrar:

```
source:
    plugin: csv
    path: public://csv/people.csv
column_names:
    0:
        first_name : First Name
    1:
        last_name : Last Name
```

Manipulação dos dados

Nesta fase do processo realiza-se o mapeamento da informação especificada na fase anterior para a estrutura final.

Dentro do ficheiro *YAML*, utiliza-se o atributo process para identificar esta fase. Este atributo possui uma lista de pares chave-valor. A chave é o nome do atributo que será inserido na estrutura e o valor pode ser uma de várias operações disponíveis: - um mapeamento simples sem a ocorrência de ações especiais; - um mapeamento simples com a ocorrência de ações especiais (por exemplo, formatação de datas); - concatenação de colunas, podendo ser utilizado um delimitador.

De seguida encontra-se um excerto de como o process se pode encontrar:

```
process:
    title:
        plugin: concat
        source:
        - first_name
        - last_name
        delimiter: ' '
    field_first_name: first_name
    field_last_name: last_name
```

Submissão dos dados

Nesta fase do processo indica-se a estrutura para onde os dados serão adicionados.

Dentro do ficheiro *YAML*, utiliza-se o atributo destination para identificar esta fase. Utiliza-se o atributo plugin para indicar a estrutura.

De seguida encontra-se um excerto de como o destination se pode encontrar:

```
destination:
    plugin: entity:node
```

2.3 CONCLUSÃO DO CAPÍTULO

Neste capítulo foi apresentado o ciclo de vida de um processo de migração, onde se enunciaram os problemas que se podem encontrar durante o processo, bem como as boas práticas a serem realizadas. Para além disso, exposeram-se algumas abordagens encontradas. A abordagem que se assemelha mais ao pretendido é a do (DRU), porque possui versatilidade em termos do número de colunas do ficheiro de entrada a serem utilizadas nos mapeamentos. De seguida, vai ser apresentado o trabalho realizado.

3

ANÁLISE E IMPLEMENTAÇÃO

Este capítulo apresenta a abordagem utilizada para desenvolver o migrador, bem como as decisões que foram tomadas para realizar o mesmo.

Posteriormente vai ser apresentada uma proposta de linguagem para especificação de transformações, bem como o funcionamento do migrador.

A seguir vai ser exposta a framework *PEG.js* , que será utilizada para auxiliar o desenvolvimento do migrador, bem como a gramática desenvolvida para fazer processamento da especificação. De seguida serão explicadas as ações semânticas inerentes às regras implementadas na gramática.

Por fim, vai ser exibida a aplicação *Web*, onde se descreve como foi desenvolvido o *Frontend* e o *Backend*.

3.1 ABORDAGEM E DECISÕES TOMADAS

O migrador foi desenvolvido com o intuito de fornecer ao utilizador uma maior versatilidade em termos de manipulação de dados a partir de campos de ficheiros *CSV*. Foram definidos os seguintes requisitos para o migrador:

- Flexibilidade em termos do número de colunas a utilizar;
- Flexibilidade em termos do número de colunas a gerar;
- Opção de limpeza de dados;
- Aplicação de funções de transformação nos dados.

O migrador possibilita a geração de dados em cinco formatos de saída:

- CSV - também conhecido por *Comma-separated values* (Mahmud et al., 2018), é um tipo de ficheiro que separa a informação utilizando vírgulas (ou pontos e vírgulas). Este tipo de ficheiro contém entradas (que são separadas por uma mudança de linha) e em cada uma delas possui várias colunas, onde cada coluna representa uma dada informação.
- JSON - também conhecido por *JavaScript Object Notation* (Bourhis et al., 2020) é um formato de informação que utiliza pares *chave-valor* para armazenar dados. Estes dados são constituídos por *data types* presentes em Javascript.
- XML - também conhecido por *eXtensible Markup Language* (Brahmia et al., 2020), é uma linguagem de *markup* que permite especificar os elementos de marcação sem limitações e guardar informação hierarquicamente.
- HTML - também conhecido por *Hypertext Markup Language* (Tabarés, 2021) é uma linguagem de *markup* que permite gerar páginas Web;
- SQL - também conhecido por *Structured Query Language* (Yunus et al., 2018), é uma linguagem que permite especificar e organizar bases de dados.

A escolha destes formatos é justificada pela Resolução do Conselho de Ministros n.º 2/2018. Esta Resolução aprovou o Regulamento Nacional de Interoperabilidade Digital (RNID), introduzida pela Lei n.º 36/2011, de 21 de junho, que veio a adotar normas abertas nos sistemas informáticos do Estado. Dentro dessas normas, adotaram-se formatações de ficheiros para determinadas situações, destacando as seguintes:

- JSON - Linguagem para descrição de documentos e formatação de dados, para interpretação não-humana, no âmbito de aplicações e sistemas acedidos por dispositivos móveis;
- XML - Linguagem para descrição de documentos e formatação de dados, para interpretação não-humana;
- HTML - Linguagem para descrição de documentos para apresentação nativa em browsers;
- SQL - Interação com SGBD (Sistema de Gestão de Bases de Dados).

Para além da Resolução mencionada, a escolha também é justificada pela Lei n. º26/2016, capítulo 1, nos seguintes artigos:

- No artigo 10º é estipulado o seguinte: 'A informação (...) deve ser disponibilizada em formato aberto e em termos que permitam o acesso aos conteúdos de forma não condicionada, privilegiando-se a disponibilização em formatos legíveis por máquina, que permitam o seu ulterior tratamento automatizado.';
- No artigo 3º, o formato aberto é considerado 'um formato de ficheiro disponibilizado ao público e reutilizável independentemente da plataforma utilizada, nos termos do regime jurídico que regula a adoção de normas abertas para a informação em suporte digital na Administração Pública' e o formato legível por máquina é considerado 'um formato de ficheiro estruturado de modo a ser possível, por meio de aplicações de software, nele identificar, reconhecer e extrair dados específicos, incluindo declarações de facto, bem como a sua estrutura interna'.

3.2 MIGRADOR

Esta secção apresenta o migrador, onde se descreve o módulo desenvolvido com o intuito de processar o ficheiro com a especificação das transformações, bem como a aplicação Web desenvolvida com o objetivo de servir de local para realizar as migrações.

3.2.1 *Módulo*

Para criar o migrador foi necessário desenvolver um módulo capaz de processar como a migração se vai realizar. Este módulo irá interpretar um programa, descrito numa linguagem criada para o propósito, que possui transformações especificadas pelo utilizador. Se tudo correr bem, o resultado do programa será devolvido.

Esta linguagem desenvolvida faz parte de um conjunto de linguagens denominado de *Domain Specific Languages* (DSLs) (Negm et al., 2019), que são desenvolvidas para resolver um dado problema para um dado contexto, não havendo a necessidade de se criar uma *General Purpose Language* (GPL).

A linguagem é classificada de *text-based* (Gómez et al., 2020), porque a linguagem é representada por um conjunto de regras (gramática) e o programa como o texto que se rege por essas regras. Para além disso, é considerada uma *internal DSL* devido ao facto de ser sido desenvolvida a partir de uma linguagem já existente (neste caso, foi desenvolvida a partir de *JavaScript*, mais especificamente o formato *JSON*).

Para auxiliar à explicação da linguagem, vai-se recorrer a um ficheiro *CSV*, que se irá denominar *alunos.csv*, que tem o seguinte conteúdo:

```
id,nome,apelido,notas
1,carlos,barbosa,12#15#13
2,adolfo,dias,13#19#9
3,martim,manhã,14#14#12
```

Figura 3: Conteúdo do ficheiro *alunos.csv*

Linguagem

A linguagem para a especificação das transformações tem a forma de um dicionário, possuindo uma lista de pares chave-valor:

- fileName: o nome do ficheiro que vai servir de entrada;

```
{
    "fileName" : "alunos.csv",
    ...
}
```

- type: o tipo de saída que será gerado, que pode ser *CSV*,*JSON*, *XML*, *HTML* ou *SQL*;

```
{
    "type" : "CSV",
    ...
}
```

- headers: flag que indica se o ficheiro tem cabeçalhos ou não. Caso não se introduza este par, assume-se que não possui cabeçalhos;

```
{
    "headers" : false,
    ...
}
```

- transformations: lista de transformações que serão realizadas para a migração. Se a lista for vazia, significa que se irão migrar todos os campos sem quaisquer transformações;

```
{
    "transformations" : [],
    ...
}
```

- root (opcional): campo apenas necessário para o XML caso se queira dar um nome à raíz. Caso não se introduza este par, a raíz será designada por *root*;

```
{
    "root" : "class",
    ...
}
```

- row (opcional): campo apenas necessário para o XML caso se queira dar um nome a cada entrada. Caso não se introduza este par, as entradas serão designadas por *row*;

```
{
    "row" : "student",
    ...
}
```

- tableName (opcional): campo apenas necessário para o SQL caso se queira dar um nome à tabela. Caso não se introduza este par, a tabela será designada por *new_table*);

```
{
    "tableName" : "students",
    ...
}
```

- delimiter (opcional): delimitador dos campos de cada entrada. Caso não se introduza este par, o delimitador fica a vírgula (,);

```
{
    "delimiter" : ";",
    ...
}
```

- xmlOutput (opcional): campo apenas necessário para o XML caso se queira adicionar elementos aos elementos que derivam de campos;

```
{
    "xmlOutput" : false,
    ...
}
```

- outputName (opcional): nome do ficheiro resultante. Caso não se introduza este par, o nome do ficheiro será *output*.

```
{
    "outputName" : "file_output",
    ...
}
```

Em termos de transformações é possível fazer as seguintes operações:

- Criar um novo campo a partir de um campo do ficheiro de entrada;
- Criar um novo campo a partir de vários campos do ficheiro de entrada (utilizando um separador);
- Criar vários campos a partir de um único campo do ficheiro de entrada (utilizando um separador).

Dentro destas operações podem-se aplicar ainda mais 2 operações:

- Substituir caracteres especificados a partir de uma expressão regular;
- Calcular o resultado de uma função e utilizar o seu valor no campo em causa.

Para cada tipo de operação é necessário fornecer alguns parâmetros e respeitar a seguinte sintaxe:

- Para criar um novo campo a partir de um campo do ficheiro de entrada são necessários dois parâmetros: o nome do campo ou a posição do campo (dependendo do tipo de ficheiro), e o nome do novo campo. A sintaxe é a seguinte para ambos os casos:
  - **{"coluna_nova": {"column": "coluna_velha"}}** : obteve-se a coluna 'coluna_velha' a partir do ficheiro de entrada para gerar a coluna "coluna_nova"no ficheiro de saída;
  - **{"coluna_nova": {"column": "0"}}** : obteve-se a coluna da primeira posição a partir do ficheiro de entrada para gerar a coluna 'coluna_nova' no ficheiro de saída.

  Exemplo: Pretende-se migrar a coluna *nome* do ficheiro *alunos.csv* para o novo ficheiro, mudando o nome para *name*. A especificação da transformação ficará da seguinte forma:

```
{
    "transformations" : [
        {
            "name" : {
                "column" : "nome"
            }
        }
    ],
    ...
}
```

- Para criar um novo campo a partir de vários campos do ficheiro de entrada são necessários três parâmetros: uma lista com o nome do campo ou a posição do campo (dependendo do tipo de ficheiro), o nome do novo campo e o separador que se utiliza para juntar os vários campos. A sintaxe é a seguinte para ambos os casos:
  - **{"coluna_nova": {"columns": "[coluna_velha_0, coluna_velha_1]","separator":"#"}}** : obtiveram-se as colunas 'coluna_velha_0' e 'coluna_velha_1' a partir do ficheiro de entrada, ligando-as com a expressão '#' para gerar a coluna 'coluna_nova' no ficheiro de saída;
  - **{"coluna_nova": {"columns": "[0,1]","separator ":"#"}}** : obtiveram-se a primeira e segunda coluna a partir do ficheiro de entrada, ligando-as com a expressão '#' para gerar a coluna 'coluna_nova' no ficheiro de saída.

  Exemplo: Pretende-se criar a coluna *full_name* a partir da junção das colunas *nome* e *apelido* do ficheiro *alunos.csv*, sendo que o conteúdo de ambas as colunas estará separado por um espaço em branco. A especificação da transformação ficará da seguinte forma:

```
{
    "transformations" : [
        {
            "full_name" : {
                "columns" : "[nome,apelido]",
                "separator" : " "
            }
        }
    ],
    ...
}
```

- Para criar vários campos a partir de um único campo do ficheiro de entrada são necessários três parâmetros: nome do campo ou a posição do campo (dependendo do tipo de ficheiro), uma lista com os nomes dos novos campos e o separador que se utiliza para separar em vários campos. A sintaxe é a seguinte para ambos os casos:
  - **{"[coluna_nova_0,coluna_nova_1]": {"column": "coluna_velha","separator": "#"}}** : obteve-se a coluna 'coluna_velha' a partir do ficheiro de entrada, separando a mesma numa lista de dados a partir da expressão '#', para gerar as colunas 'coluna_nova_0' e 'coluna_nova_1' no ficheiro de saída;
  - **{"[coluna_nova_0,coluna_nova_1]": {"column": "0", "separator": "#"}}** : obteve-se a primeira coluna a partir do ficheiro de entrada, separando a mesma numa lista de dados a partir da expressão '#', para gerar as colunas 'coluna_nova_0' e 'coluna_nova_1' no ficheiro de saída.

  Exemplo: Pretende-se criar três colunas que se irão denominar de *first_test*, *second_test* e *third_test* a partir do campo *notas*. Como os dados estão separados por cardinal (#), vai-se utilizar esse caracter como separador. A especificação da transformação ficará da seguinte forma:

```
{
    "transformations" : [
        {
            "[first_test,second_test,third_test]" : {
                "column" : "notas",
                "separator" : "#"
            }
        }
    ],
    ...
}
```

- Para substituir caracteres, utilizam-se dois parâmetros: o 'replace', indicando uma expressão regular para o padrão que se quer substituir, e o 'replaceWith' que possui o conjunto de caracteres que serão utilizados para substituir. A sintaxe é a seguinte para ambos os casos:
  - **{"coluna_nova": {"column": "coluna_velha","replace": "[\t]+", "replaceWith":}}** : obteve-se a coluna "coluna_velha"do ficheiro de entrada, substituiu-se tudo que obedece à expressão regular '[\t]+' por ' ', para gerar a coluna 'coluna_nova' no ficheiro de saída;
  - **{"coluna_nova": {"column": "0","replace": "[\t]+", "replaceWith":}}** : obteve-se a primeira coluna do ficheiro de entrada, substituiu-se tudo que obedece à expressão regular '[\t]+' por ' ', para gerar a coluna 'coluna_nova' no ficheiro de saída;
- Para utilizar uma função, usa-se o parâmetro 'function'. A função só pode possuir um único argumento, que será o próprio campo. Utiliza-se a seguinte sintaxe para ambos os casos:
  - **{"coluna_nova": {"column": "coluna_velha", "function":"function func (str){return str.toUpperCase();}}}** : obteve-se a coluna 'coluna_velha' do ficheiro de entrada, aplicou-se uma função que devolve a string original convertida em maiúsculas, para gerar a coluna 'coluna_nova' no ficheiro de saída;
  - **{"coluna_nova": {"column": "0", "function":"function func (str){ return str.toUpperCase();}}}** : obteve-se a primeira coluna do ficheiro de entrada, aplicou-se uma função que devolve a string original convertida em maiúsculas, para gerar a coluna 'coluna_nova' no ficheiro de saída.

Arquitetura

O migrador é constituído por 3 módulos:

- o pré-processamento da informação;
- a conversão dos dados;
- o pós-processamento da informação, dependendo do output desejado.

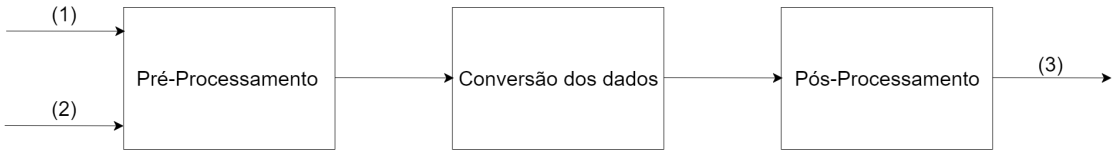

Figura 4: Esquema representativo do funcionamento do módulo (1 - ficheiro CSV ; 2 - Ficheiro com a especificação das transformações ; 3 - Output gerado)

Pré-Processamento

Nesta fase, é processado o ficheiro com a especificação das transformações utilizando PEG.js , onde se obterá toda a informação acerca da migração a partir da linguagem criada.

Após a obtenção da informação pretendida, o ficheiro que se pretende migrar será processado, utilizando uma biblioteca chamada de CSV.js (Adaltas, 2010), gerando uma representação intermédia constituída por uma lista de listas de Strings. Será a partir desta representação intermédia que os conversores de dados irão realizar a migração.

PEG.js

Para processar a linguagem desenvolvida, o migrador recorre ao *parser* gerado do PEG.js (Majda, 2010), que é uma framework em Javascript que permite ao utilizador gerar *parsers* que possibilitam o processamento de informação.

O parser é constituído por um conjunto de regras, onde cada regra possui o seu nome e a expressão de *parsing*. Esta possui um padrão que se quer identificar (utilizando expressões regulares) e a ação a ser realizada (código *JavaScript*).

De seguida vai ser apresentado um exemplo de uma gramática que faz processamento de texto para realizar um somatório:

```
calc = n:number + c:calc {return n+c;}
       / n:number {return n;}
number = d:[0-9]+ { return parseInt(d.join(""), 10); }
```

Neste exemplo pode-se verificar que existem 2 regras: o *calc* e o *number*. O *number* representa qualquer número que se encontre no somatório e o *calc* representa cada conta de adição que será realizada. Como exemplo prático vai ser utilizado o seguinte somatório:

```
1+2+3+4+5
```

A partir deste exemplo prático, o parser irá calcular o valor da seguinte forma:

```
1+(2+(3+(4+(5))))
```

e irá realizar o somatório da direita para a esquerda, onde se obterá o valor *15*.

Para além do que foi mostrado, também existe o *initializer*. O *initializer* é uma área delimitada por chavetas ('{' e '}') que possui código *Javascript* que será executado antes de se realizar qualquer tipo de processamento. Nessa área é possível adicionar funções e variáveis que podem ser utilizadas em qualquer altura durante o processamento.

No seguinte exemplo, pode-se verificar a regra *sum* que utiliza a função *print_result* para imprimir o resultado que a regra *calc* devolver:

```
{
    function print_result(n){
        console.log(n)
    }
}
sum = calc {print_result(calc);}
calc = n:number + c:calc {return n+c;}
        / n:number {return n;}
number = d:[0-9]+ { return parseInt(d.join(""), 10); }
```

Gramática

Como foi explicado anteriormente, a linguagem foi desenvolvida a partir de *JSON*, onde a especificação das transformações têm a forma de um dicionário. Utilizando o PEG.js, chegou-se à seguinte gramática para lidar com a linguagem descrita:

- A regra *list* define um objeto onde se encontra todos os pares *chave-valor*:

```
1  list = _ "{" _ val (_ "," _ val)* _ "}" _
```

- A regra *val* define todos os pares *chave-valor* existentes:

```
2  val = "\"fileName\"" _ ":" _ '\"' filename '\"'
3
4      / "\"type\"" _ ":" _ '\"' type '\"'
5
6      / "\"headers\"" _ ":" _  bool
7
8      / "\"transformations\"" _ ":" _ '[' _ (dsl)? _ ']'
9
10     / "\"root\"" _ ":" _ "\"" word "\""
11
12     / "\"row\"" _ ":" _ "\"" word "\""
13
14     / "\"tableName\"" _ ":" _ "\"" word "\""
15
16     / "\"xmlOutput\"" _ ":" _ bool
17
18     / "\"outputName\"" _ ":" _ "\"" word "\""
19
20     / "\"delimiter\"" _ ":" _ delimiter
```

- A regra *bool* define os valores de um boleano, ou seja, verdadeiro (*true*) ou falso (*false*);

```
21 bool = "true"
22      / "false"
```

- A regra *filename* indica como os nomes de ficheiros devem ser definidos;

```
23 filename = word_file '.' word_file
```

- A regra *type* define todos os tipos de output permitidos;

```
24 type = "CSV"
25      / "XML"
26      / "JSON"
27      / "HTML"
28      / "SQL"
```

- A regra *dsl* define o conteúdo da lista das transformações;

```
29 dsl = dslpair (_ ',' _ dslpair)*
```

- A regra *dslpair* define o par *chave-valor* representativo da transformação;

```
30 dslpair = '{' _ '\"' word '\"' _ ":" _ "{" _
31 transformation _ "}" _ '}'
32         / '{' _ '\"[' word ((',' word)+)?
33 ']\"' _ ":" _ "{" _
34 transformation_separate _ "}" _ '}'
```

- A regra *transformation_separate* define a transformação onde se utiliza uma coluna para gerar várias colunas;

```
35 transformation_separate= "\"column\"" _ ":" _ "\"" word
36 "\"" _ "," _ "\"separator\"" _ ":" _ "\"" exp "\"" (_
37 "," _ "\"replace\"" _ ":" _ "\"" exp "\"" _  "," _
38 "\"replaceWith\"" _ ":" _ "\"" exp "\"")? (_ "," _
39 "\"function\"" _ ":" _ "\"" exp "\"")?
```

- A regra *transformation* define os restantes tipos de transformações, ou seja, os que não requerem a geração de várias colunas;

```
40 transformation= "\"columns\"" _ ":" _ "\"[" join_list
41 "]\"" _ "," _ "\"separator\"" _ ":" _ "\"" exp "\""
42 (_ "," _ "\"replace\"" _ ":" _ "\"" exp "\"" _  "," _
43 "\"replaceWith\"" _ ":" _ "\"" exp "\"")? (_ "," _
44 "\"function\"" _ ":" _ "\"" exp "\"")?
45
46                / "\"column\"" _ ":" _ "\"" word "\""
47 (_ "," _ "\"replace\"" _ ":" _ "\"" exp "\"" _  ","
48 _ "\"replaceWith\"" _ ":" _ "\"" exp "\"")? (_ ","
49 _ "\"function\"" _ ":" _ "\"" exp "\"")?
```

- A regra *join_list* define a lista de colunas a gerar, caso se pretenda a geração de múltiplas colunas a partir de uma;

```
50 join_list = word (',' word)*
```

- A regra *delimiter* define o delimitador;

```
51 delimiter = '\"' d_elem '\"'
```

- A regra *d_elem* define a expressão para o delimitador;

```
52 d_elem = [^\"]+
```

- A regra *word_file* define a expressão para o nome e a extensão do ficheiro;

```
53 word_file = [^.,\"]+
```

- A regra *word* define todo o tipo de palavras;

```
54 word = [^,\"\[\]]+
```

- A regra *exp* define a expressão para *regex*;

```
55 exp = [^\"]*
```

- A regra _ define todos os espaços vazios que se encontram no ficheiro;

```
56 _ "whitespace" = [ \t\n\r]*
```

Incorporação das Ações Semânticas

Neste caso específico, o *initializer* possui uma variável e uma função relevantes: uma variável entitulada de *res*, que é uma estrutura de dados que contém todos os dados inerentes à realização da migração, bem como uma função chamada de *parseFile*, que faz pré-processamento do ficheiro de entrada, realiza o processo de migração e devolve o resultado final.

De seguida serão enunciadas as ações semânticas para cada regra:

- *list* : Executa a função *parseFile*, após a obtenção de todos os dados necessários para a realização do processo de migração;
- *val* : Obtem os dados para cada tipo de campo, inserindo na variável *res*;
- *bool* : Devolve booleanos;
- *filename* : Devolve a *string* com o nome do ficheiro
- *type* : Devolve o tipo de ficheiro para o qual se irá realizar a migração;
- *dsl* : Não possui ação semântica
- *dslpair* : Cria uma estrutura auxiliar para adicionar a transformação à lista de transformações;
- *transformation_separate* : Devolve a informação necessária para a estrutura auxiliar que representará a especificação da transformação;
- *transformation* : Devolve a informação necessária para a estrutura auxiliar que representará a especificação da transformação;
- *join_list* : Devolve uma lista de colunas a gerar, caso se pretenda a geração de múltiplas colunas a partir de uma;
- *delimiter* : Devolve o delimitador;
- *d_elem* : Devolve a expressão do delimitador;
- *word_file* : Devolve uma *string* que esteja contida ou no nome ou na extensão do ficheiro de entrada;
- *word* : Devolve uma palavra (sem ser do *word_file*;
- *exp* : Devolve um conjunto de caracteres;
- _ : Não possui ação semântica associada.

Conversão dos Dados

Consoante o tipo de resultado pretendido, existem duas possibilidades para a realização da conversão:

- Se o resultado pretendido for CSV, HTML ou SQL, os dados serão tratados como uma lista de listas, pois o resultado final será sempre representado num formato tabular;
- Se o resultado pretendido for JSON ou XML, os dados serão tratados como uma lista de objetos, pois o resultado final será bastante dependente da chave. Por exemplo, para JSON haverá 'chave' : 'valor' em cada objeto, para XML haverá <chave>valor</chave> para cada entrada.

Dependendo do resultado, ainda existe uma função própria para o tipo de ficheiro que o utilizador pretende gerar, devolvendo assim o resultado de acordo com as preferências do utilizador.

Pós-Processamento

Dependendo do tipo de output, a informação obtida poderá sofrer um pós-processamento. No caso de JSON, o resultado das funções que o utilizador utilizar durante a migração não se restringirá apenas em strings, mas em todos os tipos válidos em JSON:

- Number (números inteiros, número fraccionários ex: 1, 1.9, . . .);
- String (ex: 'hello world');
- Array (lista de objetos ex:[1,2,3]);
- Object (exemplo:{'name' : ' John', 'surname' : 'Smith'});
- Boolean (ex: true, false);
- null.

No caso do XML, existe a necessidade de haver um pós-processamento para se verificar se existem certos caracteres especiais, que são os seguintes: <, >, &, ' e o ". Se ocorrer alguns destes caracteres, devem-se proteger da seguinte forma:

- '<' substitui-se por '<'
- '>' substitui-se por '>';
- '&' substitui-se por '&';
- "' substitui-se por ''';
- "" substitui-se por '"'.

No caso do SQL, calcula-se o tamanho máximo que cada campo pode ter, verificando o comprimento de todas as instâncias desse campo.

3.2.2 *Aplicação Web*

Para se poder utilizar o módulo num ambiente mais acessível para os utilizadores, desenvolveu-se uma aplicação *Web*, que possui 2 servidores (um para o *Frontend* e outro para o *Backend*). O *Frontend* permite enviar os ficheiros necessários para ocorrer a migração, através de pedidos POST com os dados do ficheiro de entrada e com os dados da especificação das transformações, que por sua vez o *Backend* envia as respostas os pedidos efetuados.

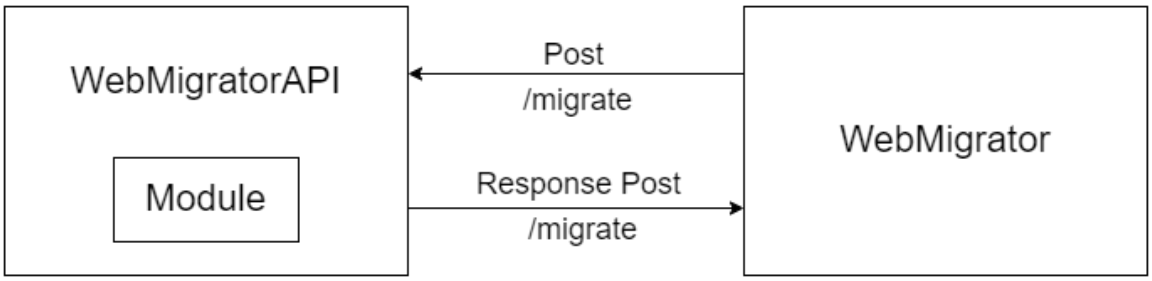

Figura 5: Esquema representativo da dinâmica entre o Módulo (*Module*), o Backend (*WebMigratorAPI*) e o Frontend (*WebMigrator*)

Backend

O *Backend* desenvolveu-se a partir de um projeto em Express.js (Holowaychuk, 2010), que é uma framework que possibilita a geração de servidores *Web*, permitindo a gestão dos pedidos *HTTP*. Este servidor vai lidar com um único pedido: um pedido *POST* para a rota */migrate*, que recebe a informação do ficheiro de entrada e a especificação das transformações.

Ao receber esta informação, será enviado para o módulo desenvolvido (que está incluído no próprio backend), que por sua vez devolve o resultado pretendido para o servidor, para ser enviado como resposta.

Frontend

O *Frontend* desenvolveu-se a partir de um projeto em VueJS (You, 2014), que é uma framework que possibilita a geração de aplicações *Web*.

A aplicação desenvolvida possui duas páginas: uma onde ocorrem as migrações e outra dedicada à aprendizagem da linguagem que permite a especificação das transformações.

Na página onde ocorrem as migrações existe uma área dedicada ao ficheiro de entrada e uma área dedicada ao ficheiro com a especificação das transformações, onde se pode escolher entre importar um ficheiro ou criar um ficheiro de raíz.

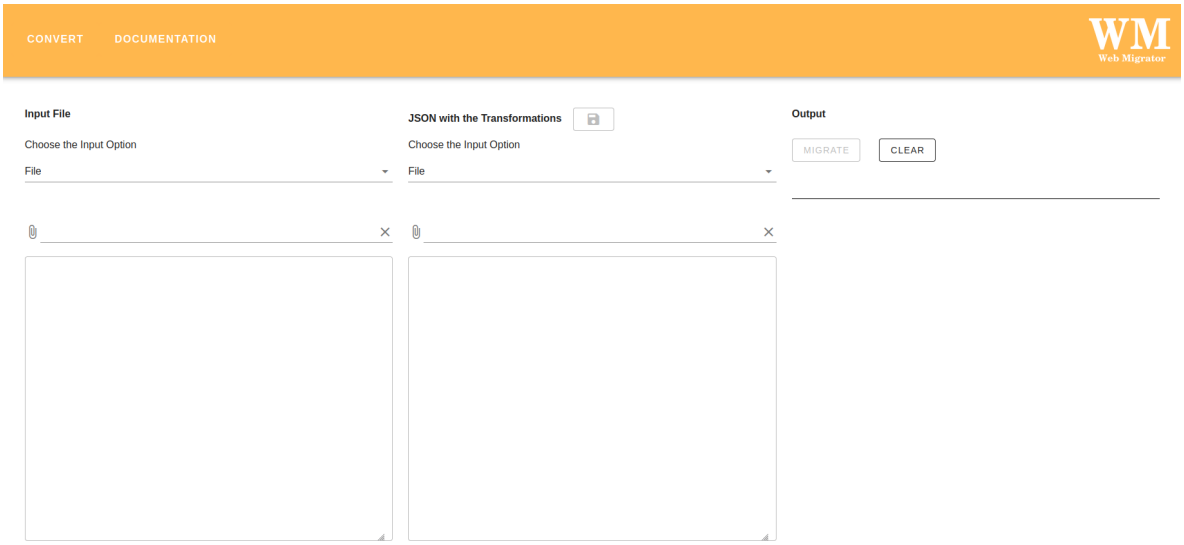

Figura 6: Aplicação Web na secção de Migração

Para começar uma migração, importa-se/cria-se o ficheiro de entrada e importa-se/cria-se o ficheiro com a especificação das transformações.

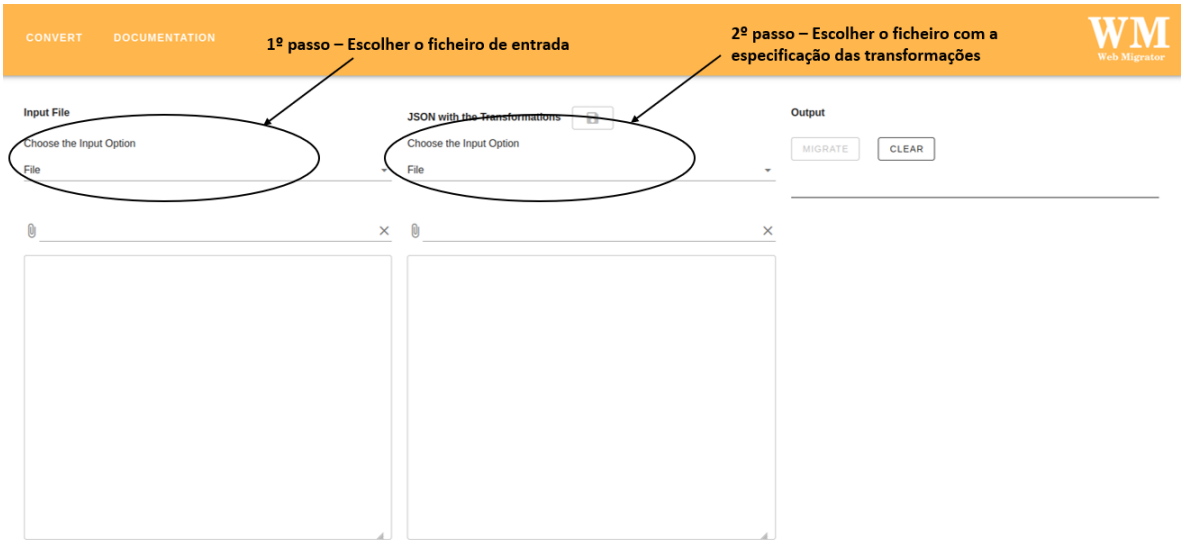

Figura 7: Passos a realizar para uma migração (parte 1)

Após a importação, submetem-se os dados ao clicar no botão para realizar a migração, onde podem ocorrer 2 situações:

- Aparece uma mensagem de erro que explica onde e como ocorreu o erro. Os tipos de erros que podem acontecer são os seguintes:
  - o ficheiro com a especificação da transformação possui erros semânticos;
  - o nome do ficheiro de entrada está incorreto;
  - as funções de Javascript possuem erros de sintaxe.
- Ocorre a migração.

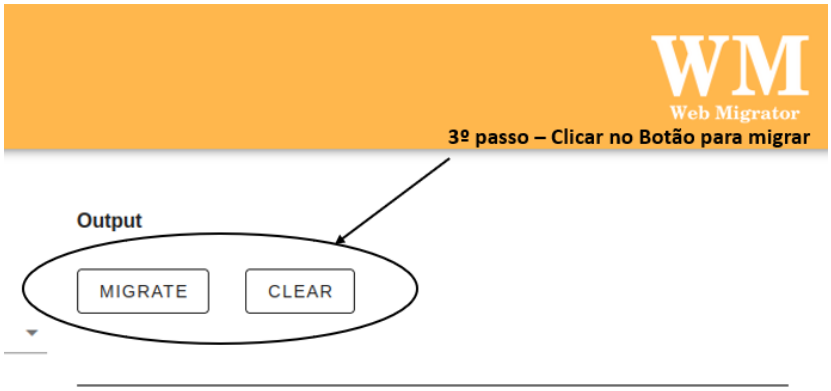

Figura 8: Passos a realizar para uma migração (parte 2)

Em termos do resultado obtido, é possível fazer download do ficheiro pretendido ou copiar a informação para o clipboard.

Figura 9: Passos a realizar para uma migração (parte 3)

Funcionamento e Opções

Para inicializar o servidor do backend, deve-se aceder à pasta 'webMigratorAPI' e correr o comando *npm start* na bash. Para inicializar o servidor do frontend, deve-se aceder à pasta 'webMigrator' e correr o comando *npm run serve* na bash. Neste momento, as portas são as seguintes:

- Backend: porta 3000 (a porta pode ser alterada caso o ficheiro 'www' que se encontra na pasta 'bin' seja acedido. Na linha 15 encontra-se a variável port, que indica a porta que o servidor utiliza. Basta substituir o valor que se encontra (3000) pela porta pretendida;
- Frontend: porta 8080 (a porta pode ser alterada no ficheiro 'package.json', na chave script, que por sua vez possui a chave serve. Dentro desta chave pode-se alterar o comando de inicialização do servidor para *vue-cli-service serve –port new_port*, onde new_port é a porta pretendida).

3.3 CONCLUSÃO DO CAPÍTULO

Neste capítulo, foram apresentadas e justificadas as decisões tomadas, bem como as partes constituintes do migrador: a linguagem,o *parser* da linguagem e a aplicação *Web*. De seguida, será apresentado o caso de estudo, que servirá para validar o trabalho realizado.

4

CASO DE ESTUDO

Neste capítulo encontra-se o caso de estudo, que servirá para testar e validar o trabalho desenvolvido.

Nas secções seguintes apresentam-se os vários problemas que este caso levanta e as respetivas soluções, utilizando a ferramenta desenvolvida.

4.1 CONTEXTO

Com o intuito de disponibilizar dados recolhidos da **Administração Pública** (AP) ao cidadão comum, a **Direção Geral do Livro, dos Arquivos e das Bibliotecas** (DGLAB) tem vindo a promover iniciativas, tais como, por exemplo, a construção e disponibilização da **Lista Consolidada** (LC).

A Lista Consolidada (Lourenço et al., 2020) consiste numa estrutura hierárquica de classes que representam funções, subfunções e processos de negócio executados pela Adminstração Pública. A Lista resulta de uma iniciativa que tem como propósito disponibilizar informação relativa à classificação e avaliação da informação pública, que serve de guia para normalizar o desenvolvimento de planos de classificação e tabelas de seleção das entidades que realizam funções de Estado.

A DGLAB decidiu criar um projeto entitulado de "M51 - CLAV - Arquivo digital: Plataforma modular de classificação e avaliação da informação pública", que se encontra incluído na Medida 51 do **Programa de simplificação administrativa Simplex +**, que se denomina "Arquivo Digital". Este projeto tem como objetivo fornecer serviços e produtos para a Administração Pública, bem como para as empresas e o cidadão. Um desses serviços é operacionalizar a Lista Consolidada: automatizar a criação de instrumentos e automatizar os processos de avaliação.

4.2 OBJETIVO

A informação que está contida na Lista Consolidada encontra-se em 19 folhas de cálculo, resultantes de uma década de diversos projetos, onde colaboraram várias entidades e centenas de pessoas da Administração Pública na recolha de informação.

Cada folha possui 25 campos e o número de entradas varia entre 15 e 79. Como existe uma quantidade decente de dados, se ocorrer o processo de migração de dados de forma manual, a tarefa torna-se lenta e existe uma grande probabilidade de ocorrer erros. Logo, é necessário um processo automático que permita corrigir ambos os problemas.

O objetivo que se pretende alcançar é migrar estas folhas de cálculo, que se encontram em *CSV*, para outros tipos de ficheiro. Especificamente, para este caso de estudo, optou-se por migrar para *JSON* e *XML*.

4.3 PROCEDIMENTO

Para realizar a migração, foi necessário saber identificar quais eram os campos a utilizar do ficheiro de input, bem como o output que se queria gerar a partir destes.

De seguida, serão apresentados todos os campos migrados, bem como uma breve síntese do que o campo é, que campos do ficheiro original foram utilizados, se foram realizadas funções de limpeza e se aplicaram funções de transformação nos dados:

- Nível
  - Campo(s) Utilizado(s): Código
  - Descrição: Nível da classe (1º e 2º nível representam funções da Administração Pública, 3º e 4º nível representam processos de negócio)
  - Funções Aplicadas : Cálculo do número de *substrings* do parâmetro Código. Estas são geradas a partir da separação do caracter ponto ('.')

  Exemplo de *output*:

```
input: 100.10.001
output: {
    "nivel" : 3
}
```

- Código
  - Campo(s) Utilizado(s): Código
  - Descrição: Identificador da classe
  - Funções Aplicadas: Remoção de mudanças de linha

  Exemplo de *output*:

```
input: 100.10.001
output: {
    "codigo" : "100.10.001"
}
```

- Status
  - Campo(s) Utilizado(s): Estado
  - Descrição: Estado da classe (se se encontra ativa, inativa, etc.)
  - Funções Aplicadas: Substituição do campo original por uma letra:
    - a *string* vazia é substituída pela letra 'A';
    - a *string* 'Em harmonização' é substituída pela letra 'H';
    - a *string* 'Inativa' é substituída pela letra 'I'.

  Exemplo de *output*:

```
input:
output: {
    "status" : "A"
}
```

- Título
  - Campo(s) Utilizado(s): Título
  - Descrição: Título da classe
  - Funções Aplicadas: Remoção de mudanças de linha

  Exemplo de *output*:

```
input: Produção e comunicação de atos legislativos
output: {
    "titulo" : "Produção e comunicação de atos
    legislativos"
}
```

- Descrição
  - Campo(s) Utilizado(s): Descrição
  - Descrição: Texto descritivo da classe
  - Funções Aplicadas: Remoção de mudanças de linha

input: Elaboração ou participação na elaboração de atos legislativos (...) entidades competentes e promulgação.
output: {
    "descricao" : "Elaboração ou participação na elaboração de atos legislativos (...) entidades competentes e promulgação."
}

- Tipo de Processo
  - Campo(s) Utilizado(s): Tipo de processo
  - Descrição: Tipo de processo (se é um processo de conservação ou processo de eliminação)
  - Funções Aplicadas: N/A

Exemplo de *output*:

input: PC
output: {
    "tipoProc" : "PC"
}

- Processo Transversal
  - Campo(s) Utilizado(s): Processo transversal (S/N)
  - Descrição: Transversabilidade do Processo (se o processo intervém como participante de outros processos ou não)
  - Funções Aplicadas: N/A

Exemplo de *output*:

input: S
output: {
    "procTrans" : "S"
}

- Notas de Aplicação
  - Campo(s) Utilizado(s): Notas de aplicação
  - Descrição: Notas textuais que descrevem em que casos deve ser utilizada esta classe
  - Funções Aplicadas: Produção de uma lista de *substrings* a partir do campo original (*string*), utilizando como separador o caracter "**#**". Após este passo, é efetuada a criação de uma estrutura para cada *substring* da lista produzida. A estrutura utilizada não varia consoante o conteúdo de cada *substring*.

Exemplo de *output*:

```
input: Iniciativa legislativa por parte de cidadãos#
       Leis#
       (...)
       Transposição de diretivas da União Europeia#
output: {
    "notasAp": [
         {
             "nota": "Iniciativa legislativa por
             parte de cidadãos"
         },
         {
             "nota": "Leis"
         },
         (...)
         {
             "nota": "Transposição de diretivas da
             União Europeia"
         }
    ]
}
```

- Exemplos de Notas de Aplicação
  - Campo(s) Utilizado(s): Exemplos de NA
  - Descrição: Exemplos onde esta classe se aplica
  - Funções Aplicadas: Produção de uma lista de *substrings* a partir do campo original (*string*), utilizando como separador o caracter "**#**". Após este passo, é efetuada a criação de uma estrutura para cada *substring* da lista produzida. A estrutura utilizada não varia consoante o conteúdo de cada *substring*.

Exemplo de *output*:

```
input: Código da Estrada#
       Código Civil#
       Regime de acesso à informação administrativa e
ambiental e de reutilização dos documentos administrativos#
output: {
    "exemplosNotasAp": [
         {
             "exemplo": "Código da Estrada"
         },
         {
             "exemplo": "Código Civil"
         },
         {
             "exemplo": "Regime de acesso à informação
             administrativa e ambiental e de reutilização
             dos documentos administrativos"
         }
    ]
}
```

- Notas de Exclusão
  - Campo(s) Utilizado(s): Notas de exclusão
  - Descrição: Notas textuais que descrevem em que casos não deve ser usada esta classe
  - Funções Aplicadas: Produção de uma lista de *substrings* a partir do campo original (*string*), utilizando como separador o caracter "**#**". Após este passo, é efetuada a criação de uma estrutura para cada *substring* da lista produzida. A estrutura utilizada não varia consoante o conteúdo de cada *substring*.

Exemplo de *output*:

```
input: Os procedimentos administrativos, (...) Definição e
avaliação de políticas#
O impulso legiferante, (...) Processamento de petições,
reclamações e sugestões#
output: {
    "notasEx": [
        {
            "nota": "Os procedimentos administrativos, (...)
        Definição e avaliação de políticas"
        },
        {
            "nota": "O impulso legiferante, (...)
        Processamento de petições, reclamações e sugestões"
        }
    ]
}
```

- Dono do processo
  - Campo(s) Utilizado(s): Dono do processo
  - Descrição: Entidade que fez parte do processo de negócio (presente a partir do nível 3)
  - Funções Aplicadas: Produção de uma lista de *substrings* a partir do campo original (*string*), utilizando como separador o caracter "**#**". Após este passo, é efetuada a criação de uma estrutura para cada *substring* da lista produzida. Esta possui os seguintes elementos: a *sigla*, o *idDono* e o *idTipo*.

    Existe uma variável auxiliar cujo conteúdo é um dicionário, que indica o tipo de dono a partir de qualquer *Dono do Processo*. Esta informação foi obtida a partir da análise das folhas de cálculo.

    A *sigla* é a *string* representativa do *Dono do processo*. O *idDono* é uma concatenação da abreviação do tipo do dono (obtida a partir da variável mencionada) com a sigla. O *idTipo* é o tipo do dono por extenso.

Exemplo de *output*:

```
input: AR#
       ALRAA#
       ALRAM#
       Gov#
output: {
    "donos": [
         {
             "sigla": "AR",
             "idDono": "ent_AR",
             "idTipo": "Entidade"
         },
         {
             "sigla": "ALRAA",
             "idDono": "ent_ALRAA",
             "idTipo": "Entidade"
         },



         {
             "sigla": "ALRAM",
             "idDono": "ent_ALRAM",
             "idTipo": "Entidade"
         },
         {
             "sigla": "Gov",
             "idDono": "tip_Gov",
             "idTipo": "TipologiaEntidade"
         }
    ]
}
```

- Participantes
  - Campo(s) Utilizado(s): Participante no processo, Tipo de intervenção do participante
  - Descrição: Participantes de um processo de negócio (presente a partir do nível 3)
  - Funções Aplicadas: Neste tipo de mapeamento, são utilizados vários campos do ficheiro de entrada para gerar uma estrutura de dados. Recorreu-se ao parâmetro denominado de *function*, de modo a ser possível criar uma função para realizar o processamento de informação e devolver a estrutura de dados pretendida.

    No entanto, a implementação da função aceita apenas um parâmetro. Para dar a volta a esta limitação, optou-se por realizar a junção de cada campo envolvido (o *Participante no processo* e o *Tipo de intervenção do participante*), utilizando como separador a *substring* '**####**'. A partir desta junção, é agora possível utilizar todos os campos na função.

    Para tornar a manipulação dos dados mais simples, no início da execução da função separou-se todos os campos a partir da *substring* '**####**'. É importante salientar que nenhum ficheiro de entrada deve conter a *substring* mencionada, caso contrário pode ocorrer problemas de processamento da informação.

    1º Passo – Obtenção dos dados
    Participante no processo
    JurisAPP#
    SGPCM#
    (...)
    SGME#
    Tipo de intervenção do participante
    Assessorar#
    Assessorar#
    (...)
    Assessorar#
    2º Passo – Junção dos dados
    JurisAPP#SGPCM#(...)SGME#####Assessorar#Assessorar#(...)
    Assessorar#
    3º Passo – Separação dos dados (início da função)
    ['JurisAPP#SGPCM#(...)SGME#','Assessorar#Assessorar#(...)
    Assessorar#']

Para cada campo é aplicada a produção de uma lista de *substrings* a partir do campo original (*string*), utilizando como separador o caracter "**#**". É efetuada a criação de uma lista de estruturas com base na junção das duas listas. Cada estrutura obtida é composta por elementos que ocupam a mesma posição nas listas de cada campo. Estes elementos são fixos, sendo estes o *participLabel*, o *idTipo* e o *idParticipante*.

Existe uma variável auxiliar (denominada de *type*) cujo conteúdo é um dicionário, que indica o tipo de participante. Também existe outra variável auxiliar (denominada de *obj*) cujo conteúdo é um dicionário, que indica o papel do participante. Esta informação foi obtida a partir da análise das folhas de cálculo.

O *participLabel* é obtido a partir da variável *type*, que devolve o tipo de participante a partir do *Participante no processo*. O *idTipo* é obtido a partir da variável *obj*, que devolve o papel do participante a partir do *Tipo de relação entre processos*. O *idParticipante* é uma concatenação entre a abreviação do tipo do participante e o *Participante no processo*.

Exemplo de *output*:

```
input:
Participante no processo
JurisAPP#
SGPCM#
(...)
SGME#
Tipo de intervenção do participante
Assessorar#
Assessorar#
(...)
Assessorar#
output: {
    "participantes": [
         {
             "participLabel": "Assessor",
             "idTipo": "Entidade",
             "idParticipante": "ent_JurisAPP"
         },
         {
             "participLabel": "Assessor",
             "idTipo": "Entidade",
             "idParticipante": "ent_SGPCM"
         },
                    (...)
```

```
            {
                "participLabel": "Assessor",
                "idTipo": "Entidade",
                "idParticipante": "ent_SGME"
            }
        ]
    }
```

- Processos Relacionados
  - Campo(s) Utilizado(s): Código do processo relacionado, Tipo de relação entre processos
  - Descrição: Processos que tiveram alguma influência num dado processo de negócio (presente a partir do nível 3)
  - Funções Aplicadas: Neste tipo de mapeamento, são utilizados vários campos do ficheiro de entrada para gerar uma estrutura de dados. Recorreu-se ao parâmetro denominado de *function*, de modo a ser possível criar uma função para realizar o processamento de informação e devolver a estrutura de dados pretendida.

    No entanto, a implementação da função aceita apenas um parâmetro. Para dar a volta a esta limitação, optou-se por realizar a junção de cada campo envolvido (o *Código do processo relacionado* e o *Tipo de relação entre processos*), utilizando como separador a *substring* '**####**'. A partir desta junção, é agora possível utilizar todos os campos na função.

    Para tornar a manipulação dos dados mais simples, no início da execução da função separou-se todos os campos a partir da *substring* '**####**'. É importante salientar que nenhum ficheiro de entrada deve conter a *substring* mencionada, caso contrário pode ocorrer problemas de processamento da informação.

    1º Passo – Obtenção dos dados
    Código do processo relacionado
    100.10.200#
    100.10.400#
    (...)
    150.10.700#
    Tipo de relação entre processos
    Complementar#
    Complementar#
    (...)
    Cruzada#
    2º Passo – Junção dos dados
    100.10.200#100.10.400#(...)150.10.700#####Complementar#

Complementar#(...)Cruzada#

3º Passo – Separação dos dados (início da função)

['100.10.200#100.10.400#(...)150.10.700#','Complementar#
Complementar#(...)Cruzada#']

Para cada campo é aplicada a produção de uma lista de *substrings* a partir do campo original (*string*), utilizando como separador o caracter "**#**". É efetuada a criação de uma lista de estruturas com base na junção das duas listas. Cada estrutura obtida é composta por elementos que ocupam a mesma posição nas listas de cada campo. Estes elementos são fixos, sendo estes o *codigo* e o *idRel*.

Existe uma variável auxiliar (denominada de *obj*) cujo conteúdo é um dicionário, que indica o tipo de relação. Esta informação foi obtida a partir da análise das folhas de cálculo.

O *codigo* é a *string* representativa do *Código do processo relacionado*. O *idRel* obtem-se a partir da variável *obj*, que devolve o tipo de relação a partir do *Tipo de relação entre processos*.

Exemplo de *output*:

input:
Código do processo relacionado
100.10.200#
100.10.400#
(...)
150.10.700#
Tipo de relação entre processos
Complementar#
Complementar#
(...)
Cruzada#
output: {

```
    "processosRelacionados": [
        {
            "codigo": "100.10.200",
            "idRel": "eComplementarDe"
        },
```

```
            {
                "codigo": "100.10.400",
                "idRel": "eComplementarDe"
            },
                    (...)
            {
                "codigo": "150.10.700",
                "idRel": "eCruzadoCom"
            }
        ]
}
```

- PCA
  - Campo(s) Utilizado(s): Justificação PCA, Nota ao PCA, Prazo de conservação administrativa, Forma de contagem do PCA
  - Descrição: Período para o qual os documentos se podem manter no sistema (presente a partir do nível 3)
  - Funções Aplicadas: Neste tipo de mapeamento, são utilizados vários campos do ficheiro de entrada para gerar uma estrutura de dados. Recorreu-se ao parâmetro denominado de *function*, de modo a ser possível criar uma função para realizar o processamento de informação e devolver a estrutura de dados pretendida.

    No entanto, a implementação da função aceita apenas um parâmetro. Para dar a volta a esta limitação, optou-se por realizar a junção de cada campo envolvido (a *Justificação PCA*, a *Nota ao PCA*, o *Prazo de conservação administrativa* e a *Forma de contagem do PCA*), utilizando como separador a *substring* '**####**'. A partir desta junção, é agora possível utilizar todos os campos na função.

    Para tornar a manipulação dos dados mais simples, no início da execução da função separou-se todos os campos a partir da *substring* '**####**'. É importante salientar que nenhum ficheiro de entrada deve conter a *substring* mencionada, caso contrário pode ocorrer problemas de processamento da informação.

    ```
    1º Passo - Obtenção dos dados
    Justificação PCA
    #Critério gestionário:
    Prazo para imputação (...) tempo do mandato de maior
    duração: 5 anos (Presidente da República).
    ```

Nota ao PCA

Prazo de conservação administrativa
5
Forma de contagem do PCA
Data de conclusão do procedimento

2º Passo – Junção dos dados
#Critério gestionário:Prazo para imputação (...)
tempo do mandato de maior duração: 5 anos
(Presidente da República).########5####Data de
conclusão do procedimento
3º Passo – Separação dos dados (início da função)
[`#Critério gestionário:Prazo para imputação (...)
tempo do mandato de maior duração: 5 anos
(Presidente da República).',' `,'5','Data de
conclusão do processo']

É efetuada a criação da estrutura que possui como elementos a *justificacao*, as *notas*, os *valores* e a *formaContagem*.

O elemento denominado de *valores* corresponde à *string* representativa do *Prazo de conservação administrativa*, que corresponde ao valor do prazo, representado em anos.

A *formaContagem* é a *string representativa* da *Forma de contagem do PCA*, que corresponde à instrução relativa à ação / momento que origina a contagem.

O elemento denominado de *notas* corresponde à *string* representativa da *Nota ao PCA*, que corresponde a notas associadas ao prazo.

A *justificacao* é uma estrutura de dados que é obtida a partir do processamento do campo *Justificação PCA*. Esta constitui uma lista de critérios que servem para justificar o prazo. Para separar cada entrada dessa lista, é aplicada a produção de uma lista de *substrings* a partir do campo original, utilizando como separador o caracter '**#**'. De seguida, são criados os seguintes elementos para cada entrada:

* o elemento *tipoId* é obtido a partir da correspondência de uma expressão regular que deteta critérios que servem para justificar o prazo, a partir da *keyword* 'Critério';
* o elemento *legislacao* é obtido a partir da aplicação de uma expressão regular que deteta referências a legislações que se encontram entre parêntesis retos. Estas legislações estão associadas ao critério utilizado;

* o elemento *processos* é obtido a partir da aplicação de uma expressão regular que deteta referências processos que contêm o seguinte formato: **[0-9]{3}.[0-9]{2}.[0-9]{3}** . Estes processos estão associados ao critério utilizado;
* o elemento *conteudo* corresponde à *string* representativa sem o critério, ou seja, corresponde a um texto descritivo associado ao critério utilizado.

Exemplo de *output*:

```
input:
Justificação PCA
#Critério gestionário:
Prazo para imputação (...) tempo do mandato de maior
duração: 5 anos (Presidente da República).
Nota ao PCA

Prazo de conservação administrativa
5
Forma de contagem do PCA
Data de conclusão do procedimento
output: {
    "pca": {
        "justificacao": [
            {
                "tipoId": "CriterioGestionário",
                "legislacao": [],
                "processos": [],
                "conteudo": "Prazo para imputação (...)
                o tempo do mandato de maior duração:
                5 anos (Presidente da República)."
            }
        ],
        "notas": "",
        "valores": "5",
        "formaContagem": "Data de conclusão do procedimento"
    }
}
```

- DF
  - Campo(s) Utilizado(s): Justificação DF, Notas, Destino final
  - Descrição: Ação a realizar após o término do prazo (presente a partir do nível 3)
  - Funções Aplicadas: Neste tipo de mapeamento, são utilizados vários campos do ficheiro de entrada para gerar uma estrutura de dados. Recorreu-se ao parâmetro denominado de *function*, de modo a ser possível criar uma função para realizar o processamento de informação e devolver a estrutura de dados pretendida.

    No entanto, a implementação da função aceita apenas um parâmetro. Para dar a volta a esta limitação, optou-se por realizar a junção de cada campo envolvido (a *Justificação DF*, as *Notas* e o *Destino Final*), utilizando como separador a *substring* '**####**'. A partir desta junção, é agora possível utilizar todos os campos na função.

    Para tornar a manipulação dos dados mais simples, no início da execução da função separou-se todos os campos a partir da *substring* '**####**'. É importante salientar que nenhum ficheiro de entrada deve conter a *substring* mencionada, caso contrário pode ocorrer problemas de processamento da informação.

```
1º Passo - Obtenção dos dados
Justificação DF
#Critério de complementaridade informacional:
É complementar dos
PN 100.10.200 - Produção e comunicação de atos
regulamentares gerais;
PN 100.10.400 - Produção e comunicação de atos
regulamentares locais;
(...)
PN 100.20.200 - Produção e comunicação de instruções para
aplicação de diplomas jurídico-normativos.
Notas

Destino Final
C
2º Passo - Junção dos dados
'#Critério de complementaridade informacional: É complementar dos
PN 100.10.200 - Produção e comunicação de
atos regulamentares gerais;
PN 100.10.400 - Produção e comunicação de atos
regulamentares locais;
(...)
```

```
PN 100.20.200 – Produção e comunicação de instruções para
aplicação de diplomas jurídico-normativos.########C
```

3º Passo – Separação dos dados (início da função)

```
['#Critério de complementaridade informacional: É complementar dos
PN 100.10.200 – Produção e comunicação de
atos regulamentares gerais;
PN 100.10.400 – Produção e comunicação de atos
regulamentares locais;
(...)
PN 100.20.200 – Produção e comunicação de instruções para
aplicação de diplomas jurídico-normativos.','','C']
```

É efetuada a criação da estrutura que possui como elementos a *justificacao*, a *nota*, o *valor*.

O elemento denominado de *valor* corresponde à *string* representativa do *Destino Final*, que corresponde ao destino do documento após terminar o prazo estabelecido no *Prazo de Conservação Administrativa*. Possui 3 possíveis valores: conservação permanente ("C"), conservação parcial por amostragem ("CP") ou eliminação ("E").

O elemento denominado de *nota* corresponde à *string* representativa da *Notas*, que corresponde a notas associadas ao destino final.

A *justificacao* é uma estrutura de dados que é obtida a partir do processamento do campo *Justificação DF*. Esta constitui uma lista de critérios que servem para justificar o prazo. Para separar cada entrada dessa lista, é aplicada a produção de uma lista de *substrings* a partir do campo original, utilizando como separador o caracter '**#**'. De seguida, são criados os seguintes elementos para cada entrada:

* o elemento *tipoId* é obtido a partir da correspondência de uma expressão regular que deteta critérios que servem para justificar o destino, a partir da *keyword* 'Critério';
* o elemento *legislacao* é obtido a partir da aplicação de uma expressão regular que deteta referências a legislações que se encontram entre parêntesis retos. Estas legislações estão associadas ao critério utilizado;
* o elemento *processos* é obtido a partir da aplicação de uma expressão regular que deteta referências processos que contêm o seguinte formato: **[0-9]{3}.[0-9]{2}.[0-9]{3}** . Estes processos estão associados ao critério utilizado;
* o elemento *conteudo* corresponde à *string* representativa sem o critério, ou seja, corresponde a um texto descritivo associado ao critério utilizado.

Exemplo de *output*:

```
input:
Justificação DF
#Critério de complementaridade informacional:
É complementar dos
PN 100.10.200 – Produção e comunicação de atos regulamentares gerais;
PN 100.10.400 – Produção e comunicação de atos regulamentares locais;
(...)
PN 100.20.200 – Produção e comunicação de instruções para aplicação
de diplomas jurídico-normativos.
Notas

Destino Final
C
output: {
    "df": {
         "justificacao": [
             {
                 "tipoId": "CriterioComplementaridadeInformacional",
                 "legislacao": [],
                 "processos": [
                     {
                         "procId": "c100.10.200"
                     },
                     {
                         "procId": "c100.10.400"
                     },(...)
                     {
                         "procId": "c100.20.200"
                     }
                 ],
                 "conteudo": "É complementar dos PN 100.10.200 –
                 Produção e comunicação de atos regulamentares
                 gerais; PN 100.10.400 – Produção e comunicação
                 de atos regulamentares locais;(...); PN
                 100.20.200 – Produção e comunicação de
                 instruções para aplicação de diplomas
                 jurídico-normativos."
```

```
            }
        ],
        "nota": null,
        "valor": "C"
        }
}
```

4.4 DETALHES DA MIGRAÇÃO E CONCLUSÕES

Como foi mencionado no *Procedimento*, realizou-se o processo de migração para *JSON* e para *XML*, que se encontram em Anexo (JSON e XML). O resultado obtido ficou bastante similar ao resultado pretendido, havendo alguns campos que não foram possíveis de serem criados devido à falta de informação dentro do próprio *CSV*.

Com a linguagem desenvolvida conseguiu-se, de forma intuitiva, expressar os mapeamentos necessários, bem como a aplicação de funções de transformação de dados, para mapeamentos envolvendo um campo de entrada e um campo de saída.

Foi encontrado um obstáculo durante o processo, que foi o processamento de vários campos para gerar uma estrutura de dados específica, onde se recorreu à aplicação de funções de transformação de dados. A função possui apenas um único argumento, mas era necessário utilizar vários argumentos. Após um estudo intensivo da estrutura e da própria ferramenta, foi possível realizar este tipo de ação, onde se pode concluir que a aplicação deste tipo de transformação pode ser melhorada.

5

CONCLUSÃO

Este último capítulo apresenta uma análise do trabalho realizado, onde será apresentado o trabalho futuro e as conclusões acerca do projeto, bem como uma reflexão acerca do trabalho desenvolvido.

5.1 TRABALHO FUTURO E CONCLUSÕES ACERCA DO PROJETO

Durante o desenvolvimento verificou-se que existem algumas mudanças que podem ser realizadas, de modo a aumentar o leque de possibilidades relativas ao processo pela mesma executada.

Umas das mudanças que pode ser realizada é a adição de mais formatos de entrada (ou seja, em vez de aceitar apenas *CSV*, pode incluir *JSON*,*XML*,*SQL*, entre outros), bem como mais formatos de saída (por exemplo, *Turtle*).

Outra mudança que está planeada para o futuro é a reestruturação das funções de transformação nos casos que se aplicam a vários campos de entrada.

Por fim, está planeada a simplificação da *interface* gráfica, de modo a que a realização da especificação na aplicação seja o mais *user-friendy* com um editor orientado à sintaxe.

Posto isto, verificou-se que os subobjetivos propostos na Motivação foram cumpridos, bem como o objetivo principal. Para além disso, a ferramenta foi posta à prova com um caso de estudo, onde o resultado final ficou bastante similar ao resultado pretendido, faltando alguns campos que, devido à natureza do migrador, não eram possíveis de implementar. Sendo assim, o trabalho feito até à data está validado.

5.2 REFLEXÃO ACERCA DO VALOR DO PROJETO NO DESENVOLVIMENTO PESSOAL E PROFISSIONAL

Este projeto foi um teste aos conhecimentos adquiridos, servindo de preparação para o futuro e para o mercado de trabalho.

Serviu para pôr os conhecimentos obtidos na licenciatura e do mestrado em prova, nomeadamente o desenvolvimento de *parsers* e gramáticas, bem como o desenvolvimento de aplicações *Web*. Também serviu para adquirir novos conhecimentos, nomeadamente em termos de boas práticas para a realização de um processo de migração. Permitiu também desenvolver competências em termos de gestão de tempo, comunicação e de resolução de problemas, obrigando a estar aberto a novas maneiras de resolver situações.

BIBLIOGRAFIA

Drupal Steward. URL https://www.drupal.org/home.

Java. URL https://www.java.com/en/.

JavaScript. URL https://developer.mozilla.org/pt-BR/docs/Web/JavaScript.

PHP. URL https://www.php.net/.

Adaltas. CSV.js, 2010. URL https://csv.js.org/.

Murat Aykanat. Building a Generic CSV Writer/Reader Using Reflection, 2018. URL https://www.pluralsight.com/guides/building-a-generic-csv-writer-reader-using-reflection.

Pierre Bourhis, Juan L. Reutter, and Domagoj Vrgoč. JSON: Data model and query languages. *Information Systems*, 2020. ISSN 03064379. doi: 10.1016/j.is.2019.101478.

Zouhaier Brahmia, Hind Hamrouni, and Rafik Bouaziz. XML data manipulation in conventional and temporal XML databases: A survey, 2020. ISSN 15740137.

Miguel Ferreira, José Carlos Ramalho, and Luís Faria. Guidelines for legacy repository migration. In *Archiving 2013, Washington DC, USA*, 2013. URL http://hdl.handle.net/1822/23978. Accessed: 2021-04-20.

Omar S. Gómez, Raul Rosero Miranda, and Karen Verdín. Crudyleaf: A dsl for generating spring boot rest apis from entity crud operations. *Cybernetics and Information Technologies*, 20:3–14, 08 2020. doi: 10.2478/cait-2020-0024.

TJ Holowaychuk. Express.js, 2010. URL https://expressjs.com/.

Alexandra Lourenço, José Carlos Ramalho, Maria Rita Gago, and Pedro Penteado. Plataforma CLAV: contributo para a disponibilização de dados abertos da Administração Pública em Portugal 1. *Cadernos BAD*, 2020. ISSN 1645-2895.

S. M.Hasan Mahmud, Md Altab Hossin, Hosney Jahan, Sheak Rashed Haider Noori, and Touhid Bhuiyan. CSV-ANNOTATE: Generate annotated tables from CSV file. In *2018 International Conference on Artificial Intelligence and Big Data, ICAIBD 2018*, 2018. ISBN 9781538669877. doi: 10.1109/ICAIBD.2018.8396169.

David Majda. PEG.js - Parser Generator for Javascript, 2010. URL https://pegjs.org/.

Eman Negm, Soha Makady, and Akram Salah. Survey on domain specific languages implementation aspects. *International Journal of Advanced Computer Science and Applications*, 10(11), 2019. doi: 10.14569/IJACSA.2019.0101183. URL http://dx.doi.org/10.14569/IJACSA.2019.0101183.

Terence Parr. ANTLR, 1989. URL https://www.antlr.org/.

Raúl Tabarés. HTML5 and the evolution of HTML; tracing the origins of digital platforms. *Technology in Society*, 2021. ISSN 0160791X. doi: 10.1016/j.techsoc.2021.101529.

Gabriele Tomassetti. So Much Data, So Many Formats: a Conversion Service, 2018. URL https://tomassetti.me/so-much-data-so-many-formats/.

Evan You. Vue.js, 2014. URL https://vuejs.org/.

Mohd Amin Mohd Yunus, Muhammad Zainulariff Brohan, Nazri Mohd Nawi, Ely Salwana Mat Surin, Nurhakimah Azwani Md Najib, and Chan Wei Liang. Review of SQL injection: Problems and prevention. *International Journal on Informatics Visualization*, 2018. ISSN 25499904. doi: 10.30630/joiv.2.3-2.144.

Parte I

ANEXOS

A

ESPECIFICAÇÃO DESENVOLVIDA PARA JSON

```
{
    "type": "JSON",
    "fileName": "200-utf8.csv",
    "transformations": [
        {
            "nivel" : {
                "column" : "Código",
                "function" : "function level(s){
                    let res = s.split('.').filter(x => x !== '');
                    return res.length
                }"
            }
        },
        {
            "codigo": {
                "column": "Código",
                "replace" : "\n",
                "replaceWith" : ""
            }
        },
```

```
        {
            "status": {
                "column" : "Estado",
                "function" : "function stat(e){
                    let obj={
                        '':'A',
                        'Em harmonização':'H',
                        'Inativo':'I'
                    };
                    return !(obj[`${e}`]) ? 'A' : obj[`${e}`];
                }"
            }
        },
        {
            "titulo": {
                "column": "Título",
                "replace" : "\n",
                "replaceWith" : ""
            }
        },
        {
            "descricao": {
                "column": "Descrição",
                "replace" : "\n",
                "replaceWith" : ""
            }
        },
        {
            "tipoProc": {
                "column": "Tipo de processo"
            }
        },
        {
            "procTrans": {
                "column": "Processo transversal (S/N)"
            }
        },
```

```
58          {
59              "notasAp":{
60                  "column" : "Notas de aplicação",
61                  "function" : "function nap(input){
62                      input = input.replace(/#/g,'')
63                          .split(/\\n/g);
64                      return input.map(x => {
65                          return {nota:x}}
66                          ).filter(x => x.nota !== '')
67                      }"
68              }
69          },
70          {
71              "exemplosNotasAp": {
72                  "column" : "Exemplos de NA",
73                  "function" : "function ena (input){
74                      let res = [];
75                      input.replace(/#/g,'')
76                          .split(/\\n/g)
77                          .filter(x => x !== '')
78                          .map(x => res.push(
79                              {'exemplo' : x}
80                          ));
81                      return res;
82                  }"
83              }
84          },
85          {
86              "notasEx": {
87                  "column" : "Notas de exclusão",
88                  "function" : "function nex(input){
89                      input = input.replace(/#/g,'')
90                          .split(/\\n/g);
91                      return input.map(x => {return {nota:x}})
92                          .filter(x => x.nota !== '')
93                      }"
94              }
95          },
```

```
        {
            "donos" : {
                "column" : "Dono do processo",
                "function" : "function dono(input){
                    input = input.replace(/\\n/g,'')
                        .replace(/( |\\.)+/g,'_').split(/#/g)
                        .filter(x => x !== '');

                    let obj = {
                    'ALRAA':'ent_ALRAA','ALRAM':'ent_ALRAM',
                    'AR':'ent_AR','Gov':'tip_Gov','STA':'ent_STA',
                    'STJ':'ent_STJ','TC':'ent_TC',
                    'TContas':'ent_TContas','GovReg-A':'ent_GovReg-A',
                    'GovReg-M':'ent_GovReg-M','AL':'tip_AL',
                    'CCDR':'tip_CCDR','ACT':'ent_ACT','AP':'tip_AP',
                    'ASAE':'ent_ASAE','CASES':'ent_CASES',
                    'CGA':'ent_CGA','CITE':'ent_CITE',
                    'CNPDPCJ':'ent_CNPDPCJ','CP':'ent_CP',
                    'CPL':'ent_CPL','CRL':'ent_CRL',
                    'DGADR':'ent_DGADR','DGAE':'ent_DGAE',
                    'DGAV':'ent_DGAV','DGC':'ent_DGC',
                    'DGERT':'ent_DGERT','DGPM':'ent_DGPM',
                    'DGRM':'ent_DGRM','DGSS':'ent_DGSS',
                    'DRAP':'tip_DRAP','GAMA':'ent_GAMA',
                    'GEE':'ent_GEE','GEP':'ent_GEP','GPP':'ent_GPP',
                    'Grupo_IP':'ent_Grupo_IP','ICNF':'ent_ICNF',
                    'IEFP':'ent_IEFP','IES':'tip_IES',
                    'IFAP':'ent_IFAP','IGAMAOT':'ent_IGAMAOT',
                    'IGFCSS':'ent_IGFCSS','IGFSS':'ent_IGFSS',
                    'IGMTSSS':'ent_IGMTSSS','II':'ent_II',
                    'IMT':'ent_IMT','INFARMED':'ent_INFARMED',
                    'INIAV':'ent_INIAV','INR':'ent_INR',
                    'ISS':'ent_ISS','IVDP':'ent_IVDP','IVV':'ent_IVV',
                    'OIT':'ent_OIT','PDR2020':'ent_PDR2020',
                    'POISE':'ent_POISE','SGME':'ent_SGME',
                    'SGMTSSS':'ent_SGMTSSS','IPQ':'ent_IPQ',
                    'CM':'ent_CM','RA-A':'ent_RA-A','RA-M':'ent_RA-M',
                    'CES':'ent_CES','DGAEP':'ent_DGAEP',
```

```
                'DGACCP':'ent_DGACCP','SGMNE':'ent_SGMNE',
                'GEPAC':'ent_GEPAC','DGTF':'ent_DGTF',
                'APA':'ent_APA','FA':'tip_FA','FSS':'tip_FSS',
                'SEF':'ent_SEF','SIRP':'ent_SIRP',
                'DGEEC':'ent_DGEEC','DGEG':'ent_DGEG',
                'DGPJ':'ent_DGPJ','INE':'ent_INE',
                'DGAEu':'ent_DGAEu','DGPE':'ent_DGPE',
                'PCM':'ent_PCM','ER':'tip_ER','Emb':'tip_Emb',
                'Camões':'ent_Camões','AICEP':'ent_AICEP',
                'DGLAB':'ent_DGLAB','IAPMEI':'ent_IAPMEI',
                'INA':'ent_INA','Trb':'tip_Trb',
                'IGFEJ':'ent_IGFEJ','SPE':'tip_SPE',
                'IP':'tip_IP','DGRSP':'ent_DGRSP','AT':'ent_AT',
                'ESPS':'tip_ESPS','EAARPNC':'tip_EAARPNC',
                'SGMF':'ent_SGMF','DGEstE':'ent_DGEstE',
                'ANA':'ent_ANA','Marinha':'ent_Marinha',
                'IGCP':'ent_IGCP','CMVM':'ent_CMVM',
                'IRN':'ent_IRN','DGAL':'ent_DGAL',
                'ADSE':'ent_ADSE','IASFA':'ent_IASFA',
                'ISSA':'ent_ISSA','ISSM':'ent_ISSM',
                'BNP':'ent_BNP','ERC':'ent_ERC','Cns':'tip_Cns',
                'SGMAI':'ent_SGMAI','ANEPC':'ent_ANEPC',
                'ANSR':'ent_ANSR','IPST':'ent_IPST',
                'SPMS':'ent_SPMS','ACSS':'ent_ACSS',
                'DGAJ':'ent_DGAJ','INMLCF':'ent_INMLCF',
                'SSAP':'ent_SSAP','IGAC':'ent_IGAC',
                'IPDJ':'ent_IPDJ','ICA':'ent_ICA','AMN':'ent_AMN',
                'CP-MC':'ent_CP-MC','DGPC':'ent_DGPC',
                'PSP':'ent_PSP','INPI':'ent_INPI',
                'ISAN_Portugal':'ent_ISAN_Portugal',
                'APEL':'ent_APEL','CISAC':'ent_CISAC',
                'CEGER':'ent_CEGER','DNS_PT':'ent_DNS_PT',
                'ANQEP':'ent_ANQEP','DGES':'ent_DGES',
                'TP':'ent_TP','DGS':'ent_DGS','DGAM':'ent_DGAM',
                'GNR':'ent_GNR','CEIC':'ent_CEIC',
                'ANACOM':'ent_ANACOM','ERS':'ent_ERS',
                'INEM':'ent_INEM','IMPIC':'ent_IMPIC',
                'DGE':'ent_DGE','OP':'tip_OP','CNOC':'ent_CNOC',
```

```
172                     'CCPFC':'ent_CCPFC','DGRDN':'ent_DGRDN',
173                     'SGPCM':'ent_SGPCM','GNS':'ent_GNS',
174                     'IPAC':'ent_IPAC','PM':'ent_PM','SIS':'ent_SIS',
175                     'A3ES':'ent_A3ES','ESPAP':'ent_ESPAP',
176                     'ARS':'tip_ARS','DGT':'ent_DGT','ASF':'ent_ASF',
177                     'EAI':'tip_EAI','ECS':'tip_ECS','ECM':'tip_ECM',
178                     'ECV':'tip_ECV','ENMC':'ent_ENMC',
179                     'ANAC':'ent_ANAC','GPIAAF':'ent_GPIAAF',
180                     'PJ':'ent_PJ','AAN':'ent_AAN','FAP':'ent_FAP',
181                     'EMGFA':'ent_EMGFA','CNCS':'ent_CNCS',
182                     'ANMP':'ent_ANMP','IPMA':'ent_IPMA',
183                     'PMar':'ent_PMar','AAC':'tip_AAC','MP':'ent_MP',
184                     'TIC':'tip_TIC','CPES':'ent_CPES',
185                     'OPC':'tip_OPC','JP':'tip_JP','CA':'tip_CA',
186                     'TR':'tip_TR','BNI':'ent_BNI','BNA':'ent_BNA',
187                     'CN':'tip_CN','TComarca':'tip_TComarca',
188                     'TEP':'tip_TEP','OSAE':'ent_OSAE',
189                     'EstE':'tip_EstE','CPVC':'ent_CPVC',
190                     'APAV':'ent_APAV','CPCJ':'tip_CPCJ',
191                     'SCML':'ent_SCML','TFM':'tip_TFM',
192                     'IPSS':'tip_IPSS','HH':'tip_HH',
193                     'UPCS':'tip_UPCS','INSA':'ent_INSA',
194                     'SMT':'tip_SMT','AS':'tip_AS','EPAE':'ent_EPAE',
195                     'ACES':'tip_ACES','IST':'ent_IST','UL':'tip_UL',
196                     'ECP':'tip_ECP','UI':'tip_UI','EGAP':'tip_EGAP',
197                     'PR':'ent_PR','CNE':'ent_CNE'};
198                     let res = [];
199                     for(let i=0;i<input.length;i++){
200                         let x = input[i];
201                         res.push({
202                             sigla: x,
203                             idDono: obj[x],
204                             idTipo: `${obj[x].split('_')[0] === 'ent' ?
205                             'Entidade' : 'TipologiaEntidade'}`
206                         }
207                     )};
208                     return res;
209                 }"
```

```
            }
        },
        {
            "participantes": {
                "columns": "[Participante no processo,
                Tipo de intervenção do participante]",
                "separator" : "####",
                "function" : "function c(input){
                    let i = input.split('####');
                    let s1=i[0].replace(/\\n/g,'')
                        .split('#').filter(x => x !== '');
                    let s2=i[1].replace(/\\n/g,'')
                        .split('#').filter(x => x !== '');
                    let type = {
                        'ent':['ACT','ASAE','CASES','CGA','CITE',
                        'CNPDPCJ','CP','CPL','CRL','DGADR','DGAE',
                        'DGAV','DGC','DGERT','DGPM','DGRM','DGSS',
                        'GAMA','GEE','GEP','GPP','ICNF','IEFP',
                        'IFAP','IGAMAOT','IGFCSS','IGFSS','IGMTSSS',
                        'II','IMT','INCM','INIAV','INR','ISS','IVDP',
                        'IVV','JurisAPP','OIT','PDR2020','POISE',
                        'PR','SGME','SGMTSSS','SGPCM','DGS','INFARMED',
                        'Grupo IP','AMT','DGTF','DGO','PCM','DGLAB',
                        'DGAEP','DGAL','TContas','APA','SEF','SIRP',
                        'AR','DGACCP','DGAEu','DGPE','SGMNE','ANEPC',
                        'ANPC','Camões','DGPDN','INMLCF','PJ','CRESAP',
                        'INA','CSM','CNPSSS','AT','MP','IPMA','ANACOM',
                        'CADA','CNPD','DGPC','IAPMEI','IH','LNEG',
                        'ESPAP','IGFEJ','IRN','MF','IGCP','MJ','GNR',
                        'PSP','ACSS','ANSR','DGMARE','DGRSP','DGE',
                        'DGT','IAMA','IGAI','IST','DGAM','CNPRP','IPQ',
                        'DGEG','ERSAR','AMN','TP','PM','ECHA','INEM',
                        'ANMP','SEAL','GPIAAF','PGR','AAN','ANS','FAP',
                        'Marinha','PMar','SGMAI','EMGFA','ASF','DGAJ',
                        'DGPJ','RA-A','RA-M','INPI','SGMJ','SGMF',
                        'DGRDN','SSAP','DGES','SICAD','ERS','IGAS',
                        'SPMS','BdP','CNE','ALRAA','ALRAM','CM'
                        ],
```

```
248                         'tip':['AP','CCDR','DRAP','CT','ONS','Trb',
249                         'AL','ACE','Gov','SG','FA','FSS','ER','IES',
250                         'TFM','SPE','ACES','HH','EstE','Cns','AAGR',
251                         'ARS','CH','EGAG','OPC','PMun','TEP','IPSS',
252                         'UPCS','CPCJ','SMT','EDA','UI','UL','AF','AM',
253                         'AADR','AV']};
254                         let label = {
255                             Iniciar : 'Iniciador',
256                             Assessorar: 'Assessor',
257                             Comunicar:'Comunicador',
258                             Apreciar:'Apreciador',
259                             Executar:'Executor'
260                         };
261                         let r = [];
262                         if(s2.length ===0){
263                             return r
264                         };
265                         for(let i=0;i<s1.length;i++)
266                             let key = Object.keys(type)
267                                 .find(key => type[key].includes(s1[i]));
268                             let idTipo = key === 'ent' ?
269                                 'Entidade' : 'TipologiaEntidade';
270                             r.push({
271                                 participLabel:label[s2[i]],
272                                 idTipo:idTipo,
273                                 idParticipante:`${key}_${s1[i]
274                                     .replace(/\\s/g,'_')}`
275                             });
276                         };
277                     return r;
278                 }"
279             }
280         },
```

```
281         {
282             "processosRelacionados": {
283                 "columns": "[Código do processo relacionado,
284                 Tipo de relação entre processos]",
285                 "separator" : "####",
286                 "function" : "function c(input){
287                     let inp = input.split('####');
288                     inp = inp.map(x => {
289                         return x.replace(/\\n/g,'')
290                             .replace(/\\s+/g,' ').split('#')
291                             .map(y => y.replace(/\\s+/g,' ').trim())
292                             .filter(y => y !== '')
293                         });
294                     let obj = {
295                         'Sucessão (sucessor)' : 'eSucessorDe',
296                         'Sucessão (antecessor)' : 'eAntecessorDe',
297                         'Síntese (sintetizado)' : 'eSintetizadoPor',
298                         'Cruzada' : 'eCruzadoCom',
299                         'Suplemento para' : 'eSuplementoPara',
300                         'Síntese (sintetiza)' : 'eSinteseDe',
301                         'Complementar' : 'eComplementarDe',
302                         'Sucessão(antecessor)' : 'eAntecessorDe',
303                         'Suplemento de' : 'eSuplementoDe'
304                     };
305                     let r = [];
306                     for(let i=0;i<inp[0].length;i++){
307                         r.push({
308                             codigo:inp[0][i],
309                             idRel:obj[inp[1][i]]
310                         });
311                     };
312                     return r;
313                 }"
314             }
315         },
```

```
316         {
317             "pca": {
318                 "columns": "[Justificação PCA,Nota ao PCA,
319                 Prazo de conservação administrativa,
320                 Forma de contagem do PCA]",
321                 "separator" : "####",
322                 "function": "function pca(inp){
323                     const isNull = (value) => typeof value === 'object'
324                         && !value;
325                     const obj = {
326                         'Critério gestionário':
327                         'CriterioJustificacaoGestionario',
328                         'Critério legal':'CriterioJustificacaoLegal',
329                         'Critério de utilidade administrativa':
330                         'CriterioJustificacaoUtilidadeAdministrativa'
331                     };
332                     inp = inp.split('####');
333                     let input = inp[0];
334                     input = input.split('#');
335                     input.splice(0,1);
336                     let res = {};
337                     res['justificacao']= [] ;
338                     for(let i=0;i<input.length;i++){
339                         let ind = input[i];
340                         let typeTest = ind
341                             .match(/^Critério\\s*(de)?\\s*([^:]+)\\:/);
342                         let type = (!isNull(typeTest)
343                         && typeTest.length > 0) ?
344                         typeTest[0].replace(/:/g,'') : '';
345                         let diplomaTest = ind.match(/\\[([^\\]]*)\\]/g);
346                         let diploma = isNull(diplomaTest) ?
347                             [] : diplomaTest
348                                 .map(x => {
349                                     return {
350                                         'legId' :
351                                             x.replace(/[\\[\\]]/g,'')
352                                     }
353                                 }
```

```
354                                 );
355                             let processesTest = ind.
356                                 match(/[0-9]{3}.[0-9]{2}.[0-9]{3}/g);
357                             let processes = isNull(processesTest) ?
358                                 [] : processesTest.map(x=>{
359                                     return {
360                                         'procId' : `c${x}`
361                                         }
362                                     }
363                                 );
364                             let descriptionTest = ind
365                                 .match(/^[^:]*:\\n([\\s\\S]*)/);
366                             let description = isNull(descriptionTest) ?
367                             '' : descriptionTest[1] ;
368                             if(type.lenght > 0 || diploma.length > 0 ||
369                             processes.length > 0 || description.length > 0){
370                                 res['justificacao'].push(
371                                     {
372                                         tipoId : `${obj[`${type}`]
373                                         ? obj[`${type}`] : ''}`,
374                                         legislacao : diploma,
375                                         processos : processes,
376                                         conteudo : description
377                                             .replace(/\n/g,' ')
378                                     }
379                                 )};
380                             }
381                             if(res['justificacao'].length === 0)
382                                 delete res['justificacao'];
383                             res['notas'] = inp[1].replace(/\n/g,' ');
384                             res['valores'] = inp[2]
385                                 .replace(/[^0-9]+/g,'###');
386                             if(inp[3].length > 0)
387                                 res['formaContagem'] = inp[3]
388                                     .replace(/\n/g,' ');
389                             return res;
390                         }"
391                 }
```

```
392         },
393         {
394             "df": {
395                 "columns": "[Justificação DF,Notas,Destino final]",
396                 "separator" : "####",
397                 "function": "function df(inp) {
398                     const isNull = (value) => typeof value === 'object'
399                         && !value;
400                     inp = inp.split('####');
401                     let input = inp[0];
402                     input = input.split('#');
403                     const obj = {
404                         'Critério de complementaridade informacional':
405                         'CriterioJustificacaoComplementaridadeInfo',
406                         'Critério de densidade informacional':
407                         'CriterioJustificacaoDensidadeInfo',
408                         'Critério legal':'CriterioJustificacaoLegal'
409                     };
410                     input.splice(0, 1);
411                     let res = {};
412                     res['justificacao'] = [];
413                     for (let i = 0; i < input.length; i++) {
414                         let ind = input[i];
415                         let typeTest = ind
416                             .match(/^Critério\\s*(de)?\\s*([^:]+)\\:/);
417                         let type = (!isNull(typeTest)
418                         && typeTest.length > 0) ?
419                             typeTest[0].replace(/:/g,'') : '';
420                         let diplomaTest = ind.match(/\\[([^\\]]*)\\]/g);
421                         let diploma = isNull(diplomaTest) ? [] :
422                             diplomaTest.map(x => { return {
423                                 'legId': x.replace(/[\\[\\]]/g, '')
424                                 }
425                             });
426                         let processesTest = ind
427                             .match(/[0-9]{3}.[0-9]{2}.[0-9]{3}/g);
428                         let processes = isNull(processesTest) ? [] :
429                         processesTest.map(x => {
```

```
                                return {
                                    'procId': `c${x}`
                                    }
                            });
                            let descriptionTest = ind
                                .match(/^[^:]*:\\n([\\s\\S]*)/);
                            let description = isNull(descriptionTest) ?
                                '' : descriptionTest[1];
                            if (type.lenght > 0 || diploma.length > 0 ||
                            processes.length > 0 || description.length > 0) {
                                res['justificacao'].push({
                                    tipoId: `${obj[`${type}`] ?
                                        obj[`${type}`] : ''}`,
                                    legislacao: diploma,
                                    processos: processes,
                                    conteudo: description
                                    .replace(/\n/g,' ')
                                    })
                                };
                            }
                            res['nota'] = inp[1].length > 0 ?
                            inp[1].replace(/\n/g,' ') : null;
                            res['valor'] = inp[2]
                                .replace(/[^A-Za-z]+/g,'');
                            return res;
                        }"
                }
            }
    ],
    "outputName": "new_200_utf8",
    "headers": true,
    "delimiter": ";"
}
```

B

ESPECIFICAÇÃO DESENVOLVIDA PARA XML

```
{
    "type": "XML",
    "fileName": "200-utf8.csv",
    "transformations": [
        {
            "nivel" : {
                "column" : "Código",
                "function" : "function level(s){
                    let res = s.split('.').filter(x => x !== '');
                    return res.length
                }"
            }
        },
        {
            "codigo": {
                "column": "Código",
                "replace" : "\n",
                "replaceWith" : ""
            }
        },
```

```
        {
            "status": {
                "column" : "Estado",
                "function" : "function stat(e){
                    let obj={
                        '':'A',
                        'Em harmonização':'H',
                        'Inativo':'I'
                    };
                    return !(obj[`${e}`]) ? 'A' : obj[`${e}`];
                }"
            }
        },
        {
            "titulo": {
                "column": "Título",
                "replace" : "\n",
                "replaceWith" : ""
            }
        },
        {
            "descricao": {
                "column": "Descrição",
                "replace" : "\n",
                "replaceWith" : ""
            }
        },
        {
            "tipoProc": {
                "column": "Tipo de processo"
            }
        },
        {
            "procTrans": {
                "column": "Processo transversal (S/N)"
            }
        },
```

```
58          {
59              "notasAp":{
60                  "column" : "Notas de aplicação",
61                  "function" : "function nap(input){
62                      let res = [];
63                      input = input.replace(/#/g,'')
64                          .split(/\\n/g);
65                      res = input.map(x => {return {nota:x}})
66                          .filter(x => x.nota !== '');
67                      return res.length === 0 ?
68                          '' : res.map(x => {return
69                              `<item><nota>${x.nota}</nota></item>`
70                      }).join('')
71                  }"
72              }
73          },
74          {
75              "exemplosNotasAp": {
76                  "column" : "Exemplos de NA",
77                  "function" : "function ena (input){
78                      let res = [];
79                      input.replace(/#/g,'')
80                          .split(/\\n/g).filter(x => x !== '')
81                          .map(x => res.push({'exemplo' : x}));
82                      return res.length === 0 ? '' : res.map(x => {return
83                          `<item><exemplo>${x.exemplo}</exemplo></item>`
84                      }).join('')
85                  }"
86              }
87          },
```

```
88          {
89              "notasEx": {
90                  "column" : "Notas de exclusão",
91                  "function" : "function nex(input){
92                      input = input.replace(/#/g,'')
93                          .split(/\\n/g);
94                      let res = input.map(x => {return {nota:x}})
95                          .filter(x => x.nota !== '');
96                      return res.length === 0 ? '' : res.map(x => {return
97                          `<item><nota>${x.nota}</nota></item>`
98                      }).join('')
99                  }"
100             }
101         },
102         {
103             "donos" : {
104                 "column" : "Dono do processo",
105                 "function" : "function dono(input){
106                     input = input.replace(/\\n/g,'')
107                         .replace(/( |\\.)+/g,'_')
108                         .split(/#/g).filter(x => x !== '');
109                     let obj = {'ALRAA':'ent_ALRAA','ALRAM':'ent_ALRAM',
110                     'AR':'ent_AR','Gov':'tip_Gov','STA':'ent_STA',
111                     'STJ':'ent_STJ','TC':'ent_TC',
112                     'TContas':'ent_TContas','GovReg-A':'ent_GovReg-A',
113                     'GovReg-M':'ent_GovReg-M','AL':'tip_AL',
114                     'CCDR':'tip_CCDR','ACT':'ent_ACT','AP':'tip_AP',
115                     'ASAE':'ent_ASAE','CASES':'ent_CASES',
116                     'CGA':'ent_CGA','CITE':'ent_CITE',
117                     'CNPDPCJ':'ent_CNPDPCJ','CP':'ent_CP',
118                     'CPL':'ent_CPL','CRL':'ent_CRL',
119                     'DGADR':'ent_DGADR','DGAE':'ent_DGAE',
120                     'DGAV':'ent_DGAV','DGC':'ent_DGC',
121                     'DGERT':'ent_DGERT','DGPM':'ent_DGPM',
122                     'DGRM':'ent_DGRM','DGSS':'ent_DGSS',
123                     'DRAP':'tip_DRAP','GAMA':'ent_GAMA',
124                     'GEE':'ent_GEE','GEP':'ent_GEP','GPP':'ent_GPP',
125                     'Grupo_IP':'ent_Grupo_IP','ICNF':'ent_ICNF',
```

```
126                     'IEFP':'ent_IEFP','IES':'tip_IES',
127                     'IFAP':'ent_IFAP','IGAMAOT':'ent_IGAMAOT',
128                     'IGFCSS':'ent_IGFCSS','IGFSS':'ent_IGFSS',
129                     'IGMTSSS':'ent_IGMTSSS','II':'ent_II',
130                     'IMT':'ent_IMT','INFARMED':'ent_INFARMED',
131                     'INIAV':'ent_INIAV','INR':'ent_INR',
132                     'ISS':'ent_ISS','IVDP':'ent_IVDP','IVV':'ent_IVV',
133                     'OIT':'ent_OIT','PDR2020':'ent_PDR2020',
134                     'POISE':'ent_POISE','SGME':'ent_SGME',
135                     'SGMTSSS':'ent_SGMTSSS','IPQ':'ent_IPQ',
136                     'CM':'ent_CM','RA-A':'ent_RA-A','RA-M':'ent_RA-M',
137                     'CES':'ent_CES','DGAEP':'ent_DGAEP',
138                     'DGACCP':'ent_DGACCP','SGMNE':'ent_SGMNE',
139                     'GEPAC':'ent_GEPAC','DGTF':'ent_DGTF',
140                     'APA':'ent_APA','FA':'tip_FA','FSS':'tip_FSS',
141                     'SEF':'ent_SEF','SIRP':'ent_SIRP',
142                     'DGEEC':'ent_DGEEC','DGEG':'ent_DGEG',
143                     'DGPJ':'ent_DGPJ','INE':'ent_INE',
144                     'DGAEu':'ent_DGAEu','DGPE':'ent_DGPE',
145                     'PCM':'ent_PCM','ER':'tip_ER','Emb':'tip_Emb',
146                     'Camões':'ent_Camões','AICEP':'ent_AICEP',
147                     'DGLAB':'ent_DGLAB','IAPMEI':'ent_IAPMEI',
148                     'INA':'ent_INA','Trb':'tip_Trb',
149                     'IGFEJ':'ent_IGFEJ','SPE':'tip_SPE',
150                     'IP':'tip_IP','DGRSP':'ent_DGRSP','AT':'ent_AT',
151                     'ESPS':'tip_ESPS','EAARPNC':'tip_EAARPNC',
152                     'SGMF':'ent_SGMF','DGEstE':'ent_DGEstE',
153                     'ANA':'ent_ANA','Marinha':'ent_Marinha',
154                     'IGCP':'ent_IGCP','CMVM':'ent_CMVM',
155                     'IRN':'ent_IRN','DGAL':'ent_DGAL',
156                     'ADSE':'ent_ADSE','IASFA':'ent_IASFA',
157                     'ISSA':'ent_ISSA','ISSM':'ent_ISSM',
158                     'BNP':'ent_BNP','ERC':'ent_ERC','Cns':'tip_Cns',
159                     'SGMAI':'ent_SGMAI','ANEPC':'ent_ANEPC',
160                     'ANSR':'ent_ANSR','IPST':'ent_IPST',
161                     'SPMS':'ent_SPMS','ACSS':'ent_ACSS',
162                     'DGAJ':'ent_DGAJ','INMLCF':'ent_INMLCF',
163                     'SSAP':'ent_SSAP','IGAC':'ent_IGAC',
```

```
164                 'IPDJ':'ent_IPDJ','ICA':'ent_ICA','AMN':'ent_AMN',
165                 'CP-MC':'ent_CP-MC','DGPC':'ent_DGPC',
166                 'PSP':'ent_PSP','INPI':'ent_INPI',
167                 'ISAN_Portugal':'ent_ISAN_Portugal',
168                 'APEL':'ent_APEL','CISAC':'ent_CISAC',
169                 'CEGER':'ent_CEGER','DNS_PT':'ent_DNS_PT',
170                 'ANQEP':'ent_ANQEP','DGES':'ent_DGES',
171                 'TP':'ent_TP','DGS':'ent_DGS','DGAM':'ent_DGAM',
172                 'GNR':'ent_GNR','CEIC':'ent_CEIC',
173                 'ANACOM':'ent_ANACOM','ERS':'ent_ERS',
174                 'INEM':'ent_INEM','IMPIC':'ent_IMPIC',
175                 'DGE':'ent_DGE','OP':'tip_OP','CNOC':'ent_CNOC',
176                 'CCPFC':'ent_CCPFC','DGRDN':'ent_DGRDN',
177                 'SGPCM':'ent_SGPCM','GNS':'ent_GNS',
178                 'IPAC':'ent_IPAC','PM':'ent_PM','SIS':'ent_SIS',
179                 'A3ES':'ent_A3ES','ESPAP':'ent_ESPAP',
180                 'ARS':'tip_ARS','DGT':'ent_DGT','ASF':'ent_ASF',
181                 'EAI':'tip_EAI','ECS':'tip_ECS','ECM':'tip_ECM',
182                 'ECV':'tip_ECV','ENMC':'ent_ENMC',
183                 'ANAC':'ent_ANAC','GPIAAF':'ent_GPIAAF',
184                 'PJ':'ent_PJ','AAN':'ent_AAN','FAP':'ent_FAP',
185                 'EMGFA':'ent_EMGFA','CNCS':'ent_CNCS',
186                 'ANMP':'ent_ANMP','IPMA':'ent_IPMA',
187                 'PMar':'ent_PMar','AAC':'tip_AAC','MP':'ent_MP',
188                 'TIC':'tip_TIC','CPES':'ent_CPES',
189                 'OPC':'tip_OPC','JP':'tip_JP','CA':'tip_CA',
190                 'TR':'tip_TR','BNI':'ent_BNI','BNA':'ent_BNA',
191                 'CN':'tip_CN','TComarca':'tip_TComarca',
192                 'TEP':'tip_TEP','OSAE':'ent_OSAE',
193                 'EstE':'tip_EstE','CPVC':'ent_CPVC',
194                 'APAV':'ent_APAV','CPCJ':'tip_CPCJ',
195                 'SCML':'ent_SCML','TFM':'tip_TFM',
196                 'IPSS':'tip_IPSS','HH':'tip_HH',
197                 'UPCS':'tip_UPCS','INSA':'ent_INSA',
198                 'SMT':'tip_SMT','AS':'tip_AS','EPAE':'ent_EPAE',
199                 'ACES':'tip_ACES','IST':'ent_IST','UL':'tip_UL',
200                 'ECP':'tip_ECP','UI':'tip_UI','EGAP':'tip_EGAP',
201                 'PR':'ent_PR','CNE':'ent_CNE'};
```

```
                    let res = [];
                    for(let i=0;i<input.length;i++){
                        let x = input[i];
                        res.push({
                            sigla: x,
                            idDono: obj[x],
                            idTipo: `${obj[x].split('_')[0] === 'ent' ?
                            'Entidade' : 'TipologiaEntidade'}`
                        })};
                        return res.length === 0 ?
                            '' : res.map(x => {return
                            `<item>
                                <sigla>${x.sigla}</sigla>
                                <idDono>${x.idDono}</idDono>
                                <idTipo>${x.idTipo}</idTipo>
                            </item>`}).join('');
                }"
            }
        },
        {
            "participantes": {
                "columns": "[Participante no processo,
                Tipo de intervenção do participante]",
                "separator" : "####",
                "function" : "function c(input){
                    let i = input.split('####');
                    let s1=i[0].replace(/\\n/g,'')
                        .split('#').filter(x => x !== '');
                    let s2=i[1].replace(/\\n/g,'')
                        .split('#').filter(x => x !== '');
                    let type = {
                        'ent':['ACT','ASAE','CASES','CGA','CITE',
                        'CNPDPCJ','CP','CPL','CRL','DGADR','DGAE',
                        'DGAV','DGC','DGERT','DGPM','DGRM','DGSS',
                        'GAMA','GEE','GEP','GPP','ICNF','IEFP',
                        'IFAP','IGAMAOT','IGFCSS','IGFSS','IGMTSSS',
                        'II','IMT','INCM','INIAV','INR','ISS','IVDP',
                        'IVV','JurisAPP','OIT','PDR2020','POISE',
```

```
'PR','SGME','SGMTSSS','SGPCM','DGS','INFARMED',
'Grupo IP','AMT','DGTF','DGO','PCM','DGLAB',
'DGAEP','DGAL','TContas','APA','SEF','SIRP',
'AR','DGACCP','DGAEu','DGPE','SGMNE','ANEPC',
'ANPC','Camões','DGPDN','INMLCF','PJ','CRESAP',
'INA','CSM','CNPSSS','AT','MP','IPMA','ANACOM',
'CADA','CNPD','DGPC','IAPMEI','IH','LNEG',
'ESPAP','IGFEJ','IRN','MF','IGCP','MJ','GNR',
'PSP','ACSS','ANSR','DGMARE','DGRSP','DGE',
'DGT','IAMA','IGAI','IST','DGAM','CNPRP','IPQ',
'DGEG','ERSAR','AMN','TP','PM','ECHA','INEM',
'ANMP','SEAL','GPIAAF','PGR','AAN','ANS','FAP',
'Marinha','PMar','SGMAI','EMGFA','ASF','DGAJ',
'DGPJ','RA-A','RA-M','INPI','SGMJ','SGMF',
'DGRDN','SSAP','DGES','SICAD','ERS','IGAS',
'SPMS','BdP','CNE','ALRAA','ALRAM','CM'
],
'tip':['AP','CCDR','DRAP','CT','ONS','Trb',
'AL','ACE','Gov','SG','FA','FSS','ER','IES',
'TFM','SPE','ACES','HH','EstE','Cns','AAGR',
'ARS','CH','EGAG','OPC','PMun','TEP','IPSS',
'UPCS','CPCJ','SMT','EDA','UI','UL','AF','AM',
'AADR','AV']};
let label = {
    Iniciar : 'Iniciador',
    Assessorar: 'Assessor',
    Comunicar:'Comunicador',
    Apreciar:'Apreciador',
    Executar:'Executor'
};
if(s2.length ===0){return ''};
let r = [];
for(let i=0;i<s1.length;i++){
    let key = Object.keys(type).find(key =>
    type[key].includes(s1[i]));
    let idTipo = key === 'ent' ? 'Entidade' :
    'TipologiaEntidade';
    r.push(`<item>
```

```
                                <particip Label>
                                    ${label[s2[i]]}
                                </particip Label>
                                <idTipo>${idTipo}</idTipo>
                                <idParticipante>
                                    ${key}_${s1[i].replace(/\\s/g,'_')}
                                </idParticipante>
                                </item>`);
                        };
                        return r.join('');
                    }"
            }
        },
        {
            "processosRelacionados": {
                "columns": "[Código do processo relacionado,
                Tipo de relação entre processos]",
                "separator" : "####",
                "function" : "function c(input){
                    let inp = input.split('####');
                    inp = inp.map(x => {return
                        x.replace(/\\n/g,'')
                        .replace(/\\s+/g,' ')
                        .split('#')
                        .map(y => y.replace(/\\s+/g,' ').trim())
                        .filter(y => y !== '')
                    });
                    let obj = {
                        'Sucessão (sucessor)' : 'eSucessorDe',
                        'Sucessão (antecessor)' : 'eAntecessorDe',
                        'Síntese (sintetizado)' : 'eSintetizadoPor',
                        'Cruzada' : 'eCruzadoCom',
                        'Suplemento para' : 'eSuplementoPara',
                        'Síntese (sintetiza)' : 'eSinteseDe',
                        'Complementar' : 'eComplementarDe',
                        'Sucessão(antecessor)' : 'eAntecessorDe',
                        'Suplemento de' : 'eSuplementoDe'
                    };
```

```
316                     let r = [];
317                     for(let i=0;i<inp[0].length;i++){
318                         r.push(`<processoRelacionado>
319                             <codigo>${inp[0][i]}</codigo>
320                             <idRel>${obj[inp[1][i]]}</idRel>
321                             </processoRelacionado>`
322                         );
323                     };
324                     return r.length === 0 ? '' : r.join(''); }"
325             }
326         },
327         {
328             "pca": {
329                 "columns": "[Justificação PCA,Nota ao PCA,
330                 Prazo de conservação administrativa,
331                 Forma de contagem do PCA]",
332                 "separator" : "####",
333                 "function": "function pca(inp){
334                     const isNull = (value) => typeof value === 'object'
335                         && !value;
336                     const obj = {
337                         'Critério gestionário':
338                         'CriterioJustificacaoGestionario',
339                         'Critério legal':'CriterioJustificacaoLegal',
340                         'Critério de utilidade administrativa':
341                         'CriterioJustificacaoUtilidadeAdministrativa'
342                     };
343                     inp = inp.split('####');
344                     let input = inp[0];
345                     input = input.split('#');
346                     input.splice(0,1);
347                     let res = {};
348                     res['justificacao']= [] ;
349                     for(let i=0;i<input.length;i++){
350                         let ind = input[i];
351                         let typeTest = ind
352                             .match(/^Critério\\s*(de)?\\s*([^:]+)\\:/);
353                         let type = (!isNull(typeTest)
```

```
354                         && typeTest.length > 0) ?
355                             typeTest[0].replace(/:/g,'') : '';
356                         let diplomaTest = ind
357                             .match(/\\[([^\\]]*)\\]/g);
358                         let diploma = isNull(diplomaTest) ? [] :
359                             diplomaTest.map(x => {return
360                             `<legId>
361                                 ${x.replace(/[\\[\\]]/g,'')}
362                             </legId>` }).join('');
363                         let processesTest = ind
364                         .match(/[0-9]{3}.[0-9]{2}.[0-9]{3}/g);
365                         let processes = isNull(processesTest) ? [] :
366                         processesTest.map(x=>{return
367                         `<procId>c${x}</procId>`}).join('');
368                         let descriptionTest = ind
369                             .match(/^[^:]*:\\n([\\s\\S]*)/);
370                         let description = isNull(descriptionTest) ? '' :
371                             descriptionTest[1];
372                         if(type.lenght > 0 || diploma.length > 0 ||
373                         processes.length > 0 || description.length > 0){
374                             res['justificacao'].push(`<item>
375                                 <tipoId>${obj[`${type}`]
376                                     ? obj[`${type}`] : ''}
377                                 </tipoId>
378                                 <legislacao>${diploma}</legislacao>
379                                 <processos>${processes}</processos>
380                                 <conteudo>${description
381                                     .replace(/\n/g,'')}
382                                 </conteudo>
383                                 </item>`) };
384                         }
385                         let re = '';
386                         if(res['justificacao'].length === 0)
387                             delete res['justificacao'];
388                         else re = re +
389                             `<justificacao>
390                                 ${res['justificacao'].join('')}
391                             </justificacao>`;
```

```
                        res['notas'] = inp[1].replace(/\n/g,' ');
                        res['valores'] = inp[2]
                            .replace(/[^0-9]+/g,'###');
                        re = re +
                            `<notas>${res['notas']}</notas>
                            <valores>${res['valores']}</valores>`;
                        if(inp[3].length > 0){
                            re = re + `<formaContagem>
                                ${inp[3].replace(/\n/g,' ')}
                            </formaContagem>`; }
                        return re;
                    }"
            }
        },
        {
            "df": {
                "columns": "[Justificação DF,Notas,Destino final]",
                "separator" : "####",
                "function": "function df(inp) {
                    const isNull = (value) => typeof value === 'object'
                        && !value;
                    const obj = {
                        'Critério de complementaridade informacional':
                        'CriterioJustificacaoComplementaridadeInfo',
                        'Critério de densidade informacional':
                        'CriterioJustificacaoDensidadeInfo',
                        'Critério legal':'CriterioJustificacaoLegal'
                    };
                    inp = inp.split('####');
                    let input = inp[0];
                    input = input.split('#');
                    input.splice(0, 1);
                    let res = {};
                    res['justificacao'] = [];
                    for(let i = 0; i < input.length; i++) {
                        let ind = input[i];
                        let typeTest = ind
                            .match(/^Critério\\s*(de)?\\s*([^:]+)\\:/);
```

```
                let type = (!isNull(typeTest)
                    && typeTest.length > 0) ?
                    typeTest[0].replace(/:/g,'') : '';
                let diplomaTest = ind.match(/\\[([^\\]]*)\\]/g);
                let diploma = isNull(diplomaTest) ? [] :
                    diplomaTest.map(x => {return
                    `<legId>
                        ${x.replace(/[\\[\\]]/g,'')}
                    </legId>`
                }).join('');
                let processesTest = ind
                .match(/[0-9]{3}.[0-9]{2}.[0-9]{3}/g);
                let processes = isNull(processesTest) ? [] :
                    processesTest.map(x=>{return
                    `<procId>c${x}</procId>`})
                    .join('');
                let descriptionTest = ind
                    .match(/^[^:]*:\\n([\\s\\S]*)/);
                let description = isNull(descriptionTest) ?
                    '' : descriptionTest[1];
                if (type.lenght > 0 || diploma.length > 0 ||
                processes.length > 0 || description.length > 0) {
                    res['justificacao'].push(`<item>
                        <tipoId>${obj[`${type}`] ?
                            obj[`${type}`] : ''}
                        </tipoId>
                        <legislacao>${diploma}</legislacao>
                        <processos>${processes}</processos>
                        <conteudo>
                            ${description
                                .replace(/\n/g,' ')}
                        </conteudo>
                        </item>`
                    )};
                }
                let re = '';
                if(res['justificacao'].length === 0)
                    delete res['justificacao'];
```

```
                            else
                                re = re + `<justificacao>
                                    ${res['justificacao'].join('')}
                                    </justificacao>`;
                            res['nota'] = inp[1].length > 0 ?
                                inp[1].replace(/\n/g,' ') : 'null';
                            res['valor'] = inp[2]
                                .replace(/[^A-Za-z]+/g,'');
                            re = re +
                                `<nota>${res['nota']}</nota>
                                <valor>${res['valor']}</valor>`;
                            return re;
                        }"
                }
            }
        ],
        "row" : "item",
        "xmlOutput" : true,
        "outputName": "new_200_utf8",
        "headers": true,
        "delimiter": ";"
}
```