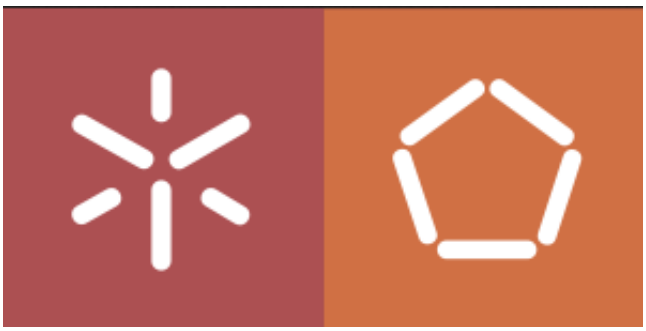

Universidade do Minho

Mestrado em Engenharia Informática

Laboratório em Engenharia Informática

CLAV - Migrador de Dados

Membros do Grupo:

João Vieira	A76516
Manuel Monteiro	A74036
Miguel Dias	PG41089

24 de Julho de 2020

Conteúdo

1 Introdução

Existe uma enorme quantidade de informação por informatizar, tal como o exercício de funções públicas. O CLAV é uma plataforma que pretende Classificar e Avaliar essa informação.

No âmbito desta disciplina foi-nos proposto implementar uma aplicação *web* para efetuar a migração dos dados referidos anteriormente. A aplicação está então dividida pelo migrador, um servidor que utiliza as funcionalidades do migrador e por fim uma interface que permite fornecer as utilidades sobre os dois primeiros sistemas. Sendo assim, a resolução do grupo passou por implementar em *Node.JS* o migrador e o servidor *backend*. O *frontend* foi desenvolvido utilizando uma *framework* de interfaces reativas o *Vue.JS*.

Este relatório descreve a arquitetura, lógica e desenvolvimento de todos os componentes deste projeto.

2 Requisitos e Funcionalidades

Após diversas reuniões com o nosso orientador de projeto foi estipulado que a aplicação teria as seguintes funcionalidades:

- Importação e leitura das folhas de cálculo
- Tratamento da informação extraída da leitura
- Impressão da informação tratada em formato *Turtle*
- Exposição e correção de erros provenientes da falta de informação nas folhas de cálculo
- Implementação de uma interface gráfica que apresente as opções acima referidas

3 Arquitetura da Aplicação

Para atingirmos os requisitos propostos de uma maneira mais eficiente, decidimos separar a aplicação em três componentes diferentes, tal como foi referido anteriormente.

O migrador é um componente *standalone* ou seja pode ser utilizado sem recurso a qualquer servidor, implementado em *Node.JS*. Como teríamos de ter uma interface gráfica para que a utilização do migrador fosse mais "amiga do utilizador", foi realizado uma interface *web* com recurso ao *Vue.JS*. Por fim, para fazer a ligação dos dois componentes foi feito um servidor *web*, implementado utilizando a ferramenta *Express.JS*

Figura 1: Arquitetura da aplicação

4 Desenvolvimento

Neste tópico vamos demonstrar toda a nossa ideologia na solução atingida pelo grupo, começando pela especificação do migrador de dados.

4.1 Migrador de Dados

Este é o componente fundamental deste projeto, pois é este que vai ler, tratar e imprimir todos os dados a migrar, como também expor e corrigir os erros que sejam encontrados.

4.1.1 Leitura de Dados

Esta é a primeira fase que será executada pelo nosso migrador.

Como foi referido anteriormente, o ficheiro contém várias folhas de cálculo para serem lidas que possuem informações que podem estar relacionadas com outros dados de outras folhas. Então o grupo optou por primeiro fazer uma leitura dessas folhas, carregando os dados para uma estrutura. É utilizada uma biblioteca *Sheet.JS* que nos facilita este processo.

Após a leitura dos dados, passamos à próxima fase do migrador, em que são tratados os dados.

4.1.2 Tratamento de Dados

Após os dados serem carregados na nossa estrutura, serão tratados.

Este tratamento passa por utilizar expressões regulares para retirar espaços que tenham sido inseridos a mais, como também, dentro certas secções são utilizadas para a identificação e separação de certas informações. É nesta fase também que são criados identificadores para cada um dos dados das entradas das folhas de cálculo.

É também nesta fase que são reportados erros de certas informações que possam estar em falta.

Depois de estarem tratados e devidamente estruturados, passamos para a fase de impressão dos dados.

4.1.3 Impressão de Dados

Tendo os dados carregados e tratados na estrutura, basta então fazer a impressão das informações em formato *Turtle*.

Tendo tudo estruturado torna-se mais fácil a implementação desta fase, pois apenas temos de verificar se certos dados estão na estrutura, pois se não estiverem é porque não estão presentes nas folhas de cálculo. Estando essas informações presentes, apenas temos de as imprimir de acordo com o formato estipulados.

É nesta fase que são chamadas funções que identificam e corrigem erros estruturais entre relações de informações das diversas folhas de cálculo. Este processo está especificado na próxima alínea deste documento.

4.1.4 Invariantes e Erros

Aquando da migração dos dados, são produzidos ficheiros de *log* que reportam faltas de informação durante a leitura do ficheiro de *input*, como referido anteriormente. Também são reportados erros quando alguns dos invariantes não são respeitados.

Para tal, definiu-se um índice de modo a identificar o que são os erros de *parsing*, e no que diz respeito aos invariantes, para melhor identificar qual o invariante e a que grupo de invariantes pertence. Foi dada prioridade aos invariantes que apresentavam erros na plataforma do CLAV, assim como aqueles que foram dados como prioritários durante as reuniões com o coordenador.

Alguns dos invariantes foi possível corrigir no migrador, os que não foram possíveis corrigir, ou que têm certos casos que não são corrigidos, são então reportados nos ficheiros de erros. A seguir, juntamente com o índice identificamos quais os invariantes que são corrigidos e quais são reportados.

```
0 - Parsing
1 - Classes
2 - Classes de 3º nível com desdobramento
    > 2.2 -> CORRIGIDO + REPORT
    > 2.4 -> CORRIGIDO + REPORT
    > 2.5 -> REPORT
3 - Invariantes sobre os PNs (classe 3)
    > 3.1  -> REPORT
    > 3.14 -> CORRIGIDO
    > 3.15 -> CORRIGIDO
4 - Invariantes sobre a relação suplementar - implicações no PCA
    > 4.1 -> REPORT
    > 4.2 -> CORRIGIDO
5 - Invariantes sobre a relação síntese - implicações no DF
    > 5.3 -> REPORT
    > 5.4 -> CORRIGIDO
6 - Invariantes sobre a relação complementar - implicações no DF
    > 6.2 -> REPORT
    > 6.3 -> CORRIGIDO
7 - Invariantes sobre o destino final (DF)
    > 7.1 -> REPORT
8 - Invariantes das relações temRelProc. Só se aplica ao 3º nível. (por ordem de pr
```

4.2 Servidor *Backend*

No que diz respeito ao servidor *Web*, este vai ter a função de receber o ficheiro *Excel* de entrada e enviá-lo para o serviço do migrador, para realizar a respetiva migração dos dados. Sendo assim, este servidor é muito simples, contendo um roteador que recebe pedidos **HTTP POST**, contendo o respetivo ficheiro a ser migrado e um pedido **HTTP GET** para extrair os ficheiros que apresentam os erros.

4.3 Servidor *Frontend*

Como foi dito anteriormente o servidor *frontend* foi implementado em *Vue.JS*.

Este apenas tem uma *view* na qual podemos fazer importação do ficheiro *excel* a ser migrado. Depois temos um botão que inicia o processo de migração e após este estar completo, temos um botão no qual podemos retirar o ficheiro *Turtle* pronto a ser importado para uma base de dados, e outro botão para as *logs* dos erros. Além disso são mostrados os erros capturados de invariantes que não sejam preservados.

5 Conclusão

Com o fim do desenvolvimento do projeto, podemos afirmar que a experiência foi bastante enriquecedora quanto à aplicação dos vários conhecimentos obtidos durante a nossa formação académica. Tendo em destaque o facto de que a aplicação resultante deste projeto, é parte integrante de um projeto de grande dimensão e em constante produção, como é o **CLAV**, o que nos proporcinou a oportunidade de fazer parte de um projeto maior do que estaríamos antes habituados.

Foram vários os desafios enfrentados, pois neste tipo de migrações temos que ter muita atenção aos diferentes tipos de dados, assim como todas as relações entre cada tipo de informação contida no ficheiro.

Em suma, é opinião do grupo que as funcionalidades da aplicação foram implementadas com sucesso, estando esta pronta a ser utilizada. Contundo, este migrador é sempre passível de ser melhorado, acompanhando também a constante evolução da própria plataforma do **CLAV**.