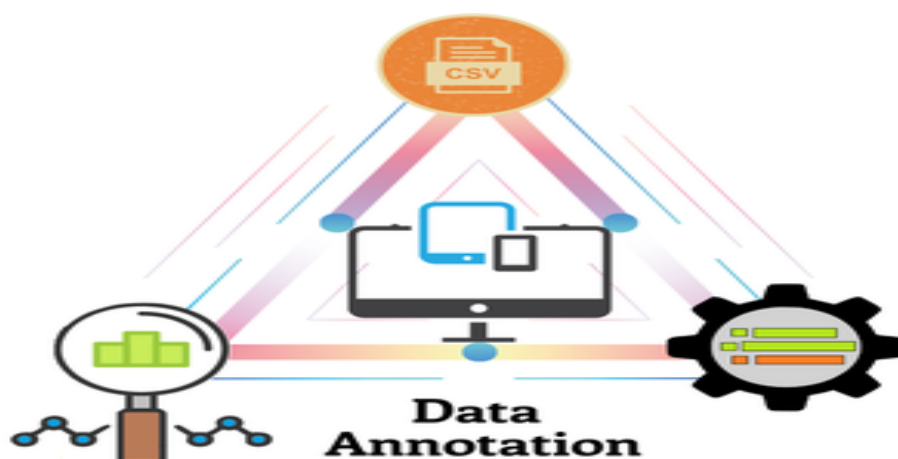




# **INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores**



## **Anotação semi-automática de ficheiros de dados**

**IVO PEREIRA  
RUBEN CAFÉ  
DUARTE FELÍCIO**

Relatório preliminar realizado no âmbito de Projecto e Seminário,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2019-2020

Orientadores : Doutor Nuno Datia  
Doutora Matilde Pós-de-Mina Pato

**Julho, 2020**



# Índice

<b>Lista de Figuras</b>	<b>v</b>
<b>Lista de Tabelas</b>	<b>vii</b>
<b>Lista de Listagens</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Problema . . . . .	2
1.2 Organização do documento . . . . .	4
<b>2 Estado da Arte</b>	<b>5</b>
2.1 Trabalho relacionado . . . . .	5
2.2 Aplicações Similares . . . . .	7
<b>3 Abordagem</b>	<b>9</b>
3.1 Análise de um ficheiro . . . . .	10
3.1.1 Identificação de métricas e dimensões . . . . .	10
3.1.2 Identificação de dados geográficos . . . . .	11
3.1.3 Identificação de relações entre métricas . . . . .	11
3.1.4 Produto final da análise . . . . .	16
3.2 Casos de utilização suportados . . . . .	17
3.2.1 Estrutura da interface visual . . . . .	20

<b>4</b>	<b>Implementação</b>	<b>23</b>
4.1	Tecnologias . . . . .	23
4.2	Autenticação - <i>Identity Server</i> . . . . .	24
4.3	Página inicial . . . . .	25
4.4	Carregamento de ficheiros . . . . .	26
4.4.1	Client Side . . . . .	26
4.4.2	Server Side . . . . .	27
4.5	Ambiente de trabalho . . . . .	28
4.5.1	Análise de ficheiros . . . . .	28
4.5.2	Remoção de ficheiros . . . . .	31
4.5.3	<i>Download</i> da meta-data . . . . .	32
4.5.4	Análise . . . . .	33
<b>5</b>	<b>Avaliação</b>	<b>37</b>
<b>6</b>	<b>Conclusão</b>	<b>39</b>
	<b>Referências</b>	<b>41</b>

# Lista de Figuras

2.1	Interface Visual - Smart Geo-Catalog . . . . .	6
2.2	Tableau Desktop - BGRI11_MAD . . . . .	7
2.3	Alteryx Designer - BGRI11_MAD . . . . .	8
3.1	Arquitetura do Projeto . . . . .	9
3.2	Modelo lógico da Base de Dados que suporta a KB - Análise . . . . .	12
3.3	Construção da árvore - 1º carácter . . . . .	13
3.4	Construção da árvore - 1º palavra . . . . .	13
3.5	Construção da árvore - 2ª palavra . . . . .	14
3.6	Construção da árvore - Árvore final . . . . .	14
3.7	Construção da árvore - Árvore final com flag . . . . .	15
3.8	Construção da árvore - Árvore final gerada pelo algoritmo . . . . .	15
3.9	Casos de Utilização da aplicação . . . . .	17
4.1	Modelo da Base de Dados - IdentityServer . . . . .	24
4.2	<i>Home Page</i> da aplicação . . . . .	25
4.3	Modelo da Base de Dados - CsvFile . . . . .	26
4.4	Página para carregar ficheiros . . . . .	27
4.5	Ambiente de Trabalho . . . . .	29
4.6	Análise do ficheiro de censos - BGRI11_MAD . . . . .	33
4.7	Árvore n-ária resultante da organização da meta-data . . . . .	35

4.8	Representação da vista final . . . . .	36
-----	--	----

# Lista de Tabelas

1.1	População de Portugal em 2011 . . . . .	2
1.2	Edifícios de Portugal em 2011 . . . . .	3
3.1	Excerto censos 2011 - Madeira . . . . .	12





## Lista de Listagens

3.1	Excerto de metadados gerados - BGRI11_MAD.json . . . . .	16
4.1	Método para verificar se a análise ja terminou . . . . .	29
4.2	Método que trata de remoção de um ficheiro . . . . .	31
4.3	Método que realiza o <i>download</i> da análise . . . . .	32
4.4	Excerto da metadata - BGRI11_MAD.csv - Categorias . . . . .	34



# 1

## Introdução

Nos tempos atuais a quantidade de dados recolhidos e armazenados mundialmente aumenta de dia para dia. É estimado que sejam gerados 4.1 terabytes de dados diariamente por quilómetro quadrado de área urbana [1]. Esta recolha de dados aumentou tanto em volume como em detalhe ao longo do tempo. Os vários fornecedores e/ou agregadores de dados têm diferentes formas de os disponibilizar, originando análises cada vez mais complexas.

Estes dados podem ser disponibilizados em vários formatos diferentes, um dos mais populares sendo o formato *csv* (*comma separated values*) [2]. Estes ficheiros podem ser acompanhados de um ficheiro de meta-informação, que contém dados referentes aos dados do ficheiro, fornecendo ao leitor uma explicação sobre a estrutura do ficheiro e dos dados. No entanto, a meta-informação presente neste tipo de ficheiros consiste apenas no nome de cada coluna tornando a análise destes ficheiros extremamente complexa e demorada se esta for feita manualmente.

É então necessário utilizar um processo semi-automático que, sobre um conjunto de dados, gere meta-informação para identificar o domínio das variáveis, relações entre os dados e os diferentes níveis de detalhe.

A partir do trabalho desenvolvido pelo Ribeiro [3] em 2019, pretendemos contribuir com uma (user-interface) interativa que gera semi-automaticamente a meta-informação com diferentes níveis de detalhe dos ficheiros em formato *csv* analisados. Ribeiro propôs um algoritmo que gera a meta-informação de ficheiros de um modo estático. A partir deste algoritmo, o processo de análise proporciona ao utilizador uma melhor

compreensão dos vários níveis de detalhe e relações entre os dados inseridos.

Este processo semi-automático pode, no entanto, apresentar desafios à sua implementação. Quer sejam por problemas devido a erros sintáticos no ficheiro *csv*, ou devido a erros no reconhecimento de níveis de detalhe por parte do algoritmo num dado domínio. Um dos problemas mais comuns é uma das colunas não se encontrar escrita da forma que o algoritmo espera, o que leva a que este não detecte essa coluna como sendo um nível de detalhe de outra.

A aplicação, após autenticação, permite carregar ficheiros *csv*, pedir ao algoritmo para os analisar dando hipótese ao utilizador para alterar a informação gerada pelo algoritmo de forma a corrigir eventuais erros e permitir customização, para finalmente exportar essa meta informação.

## 1.1 Problema

Dado à natureza de um ficheiro *csv* e de ele poder agrupar uma base de dados completa nos seus dados, a hierarquia de tabelas presentes na base de dados original é perdida. Na Tabela 1.1 é apresentado um excerto do ficheiro de censos onde é possível observar o nível de detalhe presente dentro de cada coluna.

Tabela 1.1: População de Portugal em 2011

ANO	GEO_COD	GEO_COD_DSG	N_INDIVIDUOS_PRESENT
2011	'PT	PT	10282306
2011	'15	Algarve	462862
2011	'17	Lisboa	2731459
2011	'1105	Cascais	198340
2011	'1106	Lisboa	547348
2011	'1107	Loures	196682

Este excerto apresenta a contagem de indivíduos presentes numa certa localização geográfica, essa localização conta no entanto com uma hierarquia, neste caso podemos observar na coluna 3 a hierarquia *País - Distrito - Concelho*. É também relevante observar que o mesmo valor pode aparecer associado a diferentes níveis de detalhe, neste caso, o valor **Lisboa** tanto representa o *Distrito* como o *Conselho*. Esta ambiguidade pode no entanto ser resolvida observando outra coluna, neste caso a coluna *GEO\_COD* resolve a ambiguidade.

Os níveis de detalhe podem também estar presentes entre diferentes colunas. Na Tabela 1.2 consta um excerto do mesmo ficheiro de censos onde podemos observar relações entre diferentes colunas.

Tabela 1.2: Edifícios de Portugal em 2011

ANO	GEO_COD_DSG	N_EDIFICIOS_3OU4_PISOS	.N_EDIFICIOS_5OU MAIS_PISOS	N_EDIFICIOS_CONSTR_ANTES_1919
2011	PT	432760	104013	206343
2011	Algarve	20775	5591	11109
2011	Lisboa	89135	50672	22297
2011	Cascais	9561	2606	758
2011	Lisboa	16184	15658	10279
2011	Loures	4927	3101	911

Este excerto apresenta diferentes contagens sobre o número de edifícios presentes numa determinada região. Apesar de cada uma representar um número de edifícios, estas estão em diferentes níveis de detalhe quando comparadas umas às outras. Observando os nomes das colunas podemos concluir que a coluna **N\_EDIFICIOS\_3OU4\_PISOS** e a coluna **.N\_EDIFICIOS\_5OU MAIS\_PISOS** estão ao mesmo nível de detalhe, representando o número de edifícios com um certo número de pisos (uma coluna que representa o número de edifícios com um ou dois pisos também está presente nos dados mas foi omitida da tabela devido ao seu tamanho). No entanto, a última coluna representa o número de edifícios construídos antes de 1919, estes edifícios estão também contabilizados nas três colunas mencionadas acima tornando esta num nível de detalhe das outras.

É também possível observar no exemplo que uma análise aos nomes das colunas possa gerar essa hierarquia entre elas, no entanto, como mencionado na introdução, qualquer erro sintático no nome irá provocar um erro na análise. Em específico ao exemplo, a coluna **.N\_EDIFICIOS\_5OU MAIS\_PISOS**, devido ao “.” no início do seu nome esta coluna não será identificada com o mesmo nível que a coluna à sua esquerda.

Outro aspeto relevante é a distinção entre uma dimensão e uma métrica. Uma dimensão consiste numa característica que adiciona contexto a uma métrica, sendo uma métrica uma medição caracterizada por uma ou mais dimensões. Na Tabela 1.2 é possível constatar que as duas primeiras colunas são dimensões pois dão contexto sobre as métricas que são as restantes colunas. Ou seja, por exemplo, a métrica **N\_EDIFICIOS\_3OU4\_PISOS** representa o número de edifícios num certo **ANO** e num certo local geográfico, **GEO\_COD\_DSG**.

## 1.2 Organização do documento

A organização deste documento divide-se em 6 capítulos. No capítulo 2 são apresentados trabalhos semelhantes ou com objetivos similares ao trabalho realizado. No capítulo 3 são definidos objetivos e de que forma planeamos cumpri-los. No capítulo 4 é definido e descrito de que forma implementámos o nosso trabalho. O documento termina com o capítulo 6 onde serão apresentadas as conclusões de todo o trabalho realizado assim como algumas sugestões de trabalho futuro.

# 2

## Estado da Arte

Neste capítulo é feita uma análise do trabalho relacionado com o tema tratado neste projeto. Na secção 2.1 é descrito a abordagem ao problema descrita numa dissertação de mestrado na qual este projeto se baseia (e a complementa). Na secção 2.2 são apresentadas duas aplicações que foram estudadas e analisadas de forma a auxiliar na criação da nossa interface com o utilizador.

### 2.1 Trabalho relacionado

Nesta secção são apresentados dois trabalhos que tentam solucionar um problema semelhante ao identificado nesta dissertação: criar uma solução de suporte à análise semi-automática de ficheiros de dados de forma a gerar um ficheiro de meta-informação onde constam os diferentes níveis espaciais presentes, as dimensões, métricas, relações e os níveis de detalhe das variáveis desse ficheiro ( 1.1).

**Smart Geo-Catalog** Na dissertação Smart Geo-Catalog [4], Pedro Amaral destaca a mais-valia na utilização homogénea de conjuntos de dados provenientes de fornecedores diferentes. De forma a representar a meta-informação presente nos dados, nomeadamente os domínios, dimensões, níveis de detalhe e pacotes de dados, desenvolvendo uma gramática **XML**, chamada **SGC-XML**(*Smart Geo-Catalog XML*) onde estes são guardados em tabelas criadas que representam os dados presentes no ficheiro. O

autor desenvolveu depois uma interface gráfica que permite explorar os dados do repositório, possibilitando a navegação pelos vários níveis de detalhe. A interface possibilitava também consultar um dos domínios existentes no repositório e verificar quais os níveis de detalhe disponíveis para essa dimensão. Na Figura 2.1 podemos observar uma página da interface visual, neste caso os níveis de detalhe presentes no ficheiro de censos de Portugal em 2011 e um excerto dos dados.

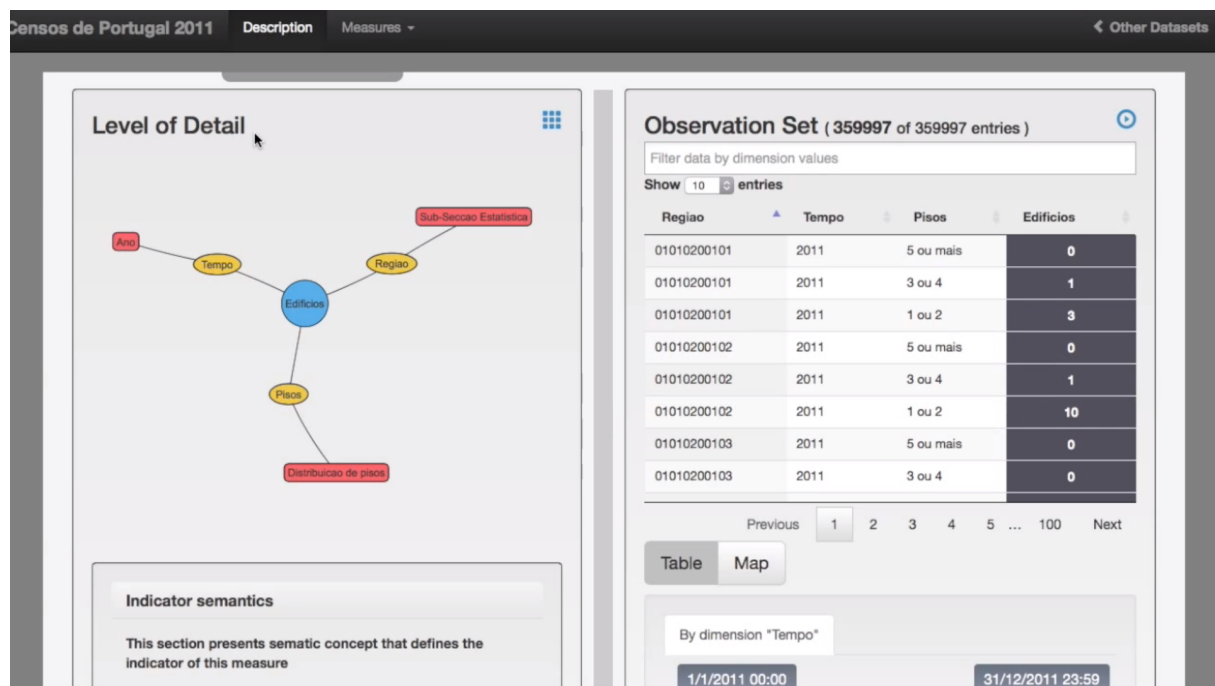


Figura 2.1: Interface Visual - Smart Geo-Catalog [4]

Foi possível tirar ideias de como desenvolver a interface, todavia, esta interface corre localmente, não estando disponível através da web. Para além disso não permite alteração feita pela análise.

**Anotação e Extração Semi-Automática de Dados multidimensionais** Dado o problema de uma crescente quantidade de dados disponibilizados por diferentes fornecedores que torna a análise manual desses dados cada vez mais complexa. Nuno Ribeiro propõe um algoritmo que passa por analisar esses dados fornecidos em formato csv através de um algoritmo que, após análise aos nomes das colunas, ao tipo de dados de cada coluna e aos dados em si, é capaz de identificar relações e o domínio de variáveis, assim como níveis de detalhe.

No entanto esta solução ao problema não fornece uma interface com o utilizador iterativa, nem permite uma customização por parte do utilizador dos resultados dados pelo algoritmo, como por exemplo alteração de níveis de detalhe de colunas (ou variáveis).



## 2.2 Aplicações Similares

Nesta secção são apresentados alguns produtos que partilham semelhanças com este trabalho, nomeadamente a identificação das métricas e dimensões. São ferramentas de análise de dados com uma interface interativa com o utilizador. Estas ferramentas são comerciais e estão no mercado há vários anos.

**Tableau Desktop** O Tableau Desktop<sup>1</sup> é uma ferramenta de software para análise e visualização de dados. Esta ferramenta consegue identificar que colunas são dimensões e métricas e também permite alterar essa distinção. A Figura 2.2 demonstra a página de análise de um ficheiro, neste caso o ficheiro de censos da Madeira em 2011 onde podemos observar na coluna da esquerda as diferentes métricas e dimensões.

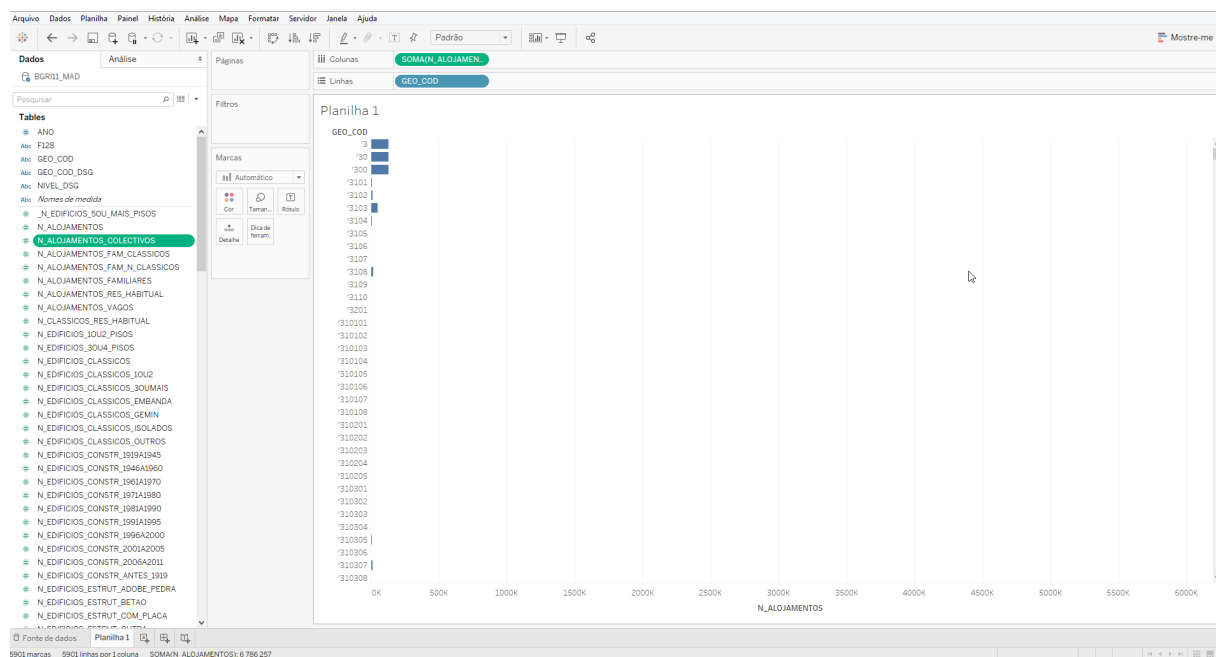


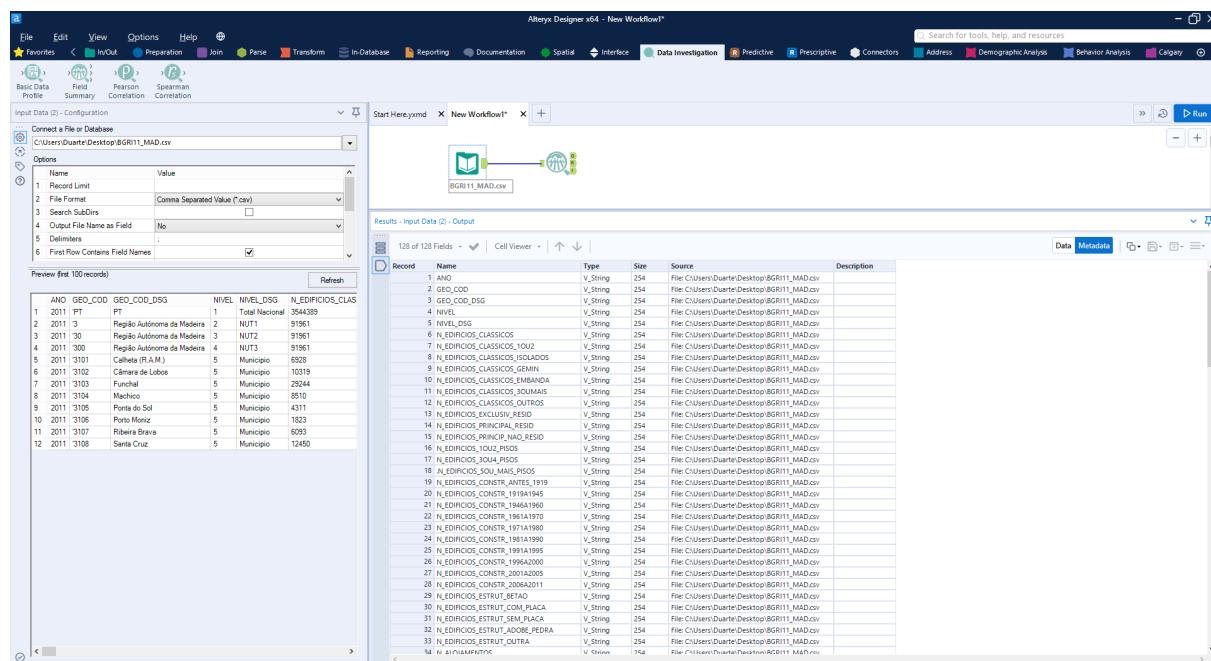
Figura 2.2: Tableau Desktop - BGR11\_MAD

Porém este software tem como objetivo auxiliar na análise de dados e não a geração de metadados. Carece a funcionalidade de análise automática para identificação de domínios e níveis de detalhe que pretendemos com o nosso projeto. E, para além disso, é um produto para fins lucrativos, não é *open source*, que o nosso projeto tem o objetivo de ser.

<sup>1</sup><https://www.tableau.com/products/desktop>

**Alteryx Designer** Alteryx Designer<sup>2</sup> é uma aplicação que fornece aos utilizadores a capacidade de preparar e analisar os seus dados, fornecendo a possibilidade de conectar com dados de outras plataformas e juntar esses dados para análise.

Esta aplicação tem a possibilidade de gerar meta-dado para o ficheiro, no entanto, é uma meta-dado bastante incompleta pois apenas fornece informação sobre o tipo de dados presente em cada coluna. A Figura 2.3 apresenta a meta-dado gerada para o ficheiro de censos da Madeira.



Record	Name	Type	Size	Source	Description
1	ANO	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
2	GEO_COD	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
3	GEO_COD_DSS	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
4	NIVEL	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
5	NIVEL_DSS	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
6	N_EDIFICIOS_CLASSICOS	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
7	N_EDIFICIOS_CLASSICOS_TOTL	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
8	N_EDIFICIOS_CLASSICOS_ISOLADOS	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
9	N_EDIFICIOS_CLASSICOS_GEMIN	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
10	N_EDIFICIOS_CLASSICOS_BANDEJA	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
11	N_EDIFICIOS_CLASSICOS_SOMAS	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
12	N_EDIFICIOS_CLASSICOS_OUTROS	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
13	N_EDIFICIOS_EXCLUSIV_RESID	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
14	N_EDIFICIOS_PRINCIPAL_RESID	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
15	N_EDIFICIOS_PRINCIPAL_NAO_RESID	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
16	N_EDIFICIOS_TOTL_RESID	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
17	N_EDIFICIOS_SOMA_RESID	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
18	N_EDIFICIOS_SOMA_NAO_RESID	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
19	N_EDIFICIOS_CONSTR_ANTES_1919	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
20	N_EDIFICIOS_CONSTR_1919A1945	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
21	N_EDIFICIOS_CONSTR_1946A1960	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
22	N_EDIFICIOS_CONSTR_1961A1970	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
23	N_EDIFICIOS_CONSTR_1971A1980	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
24	N_EDIFICIOS_CONSTR_1981A1990	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
25	N_EDIFICIOS_CONSTR_1991A1995	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
26	N_EDIFICIOS_CONSTR_1996A2000	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
27	N_EDIFICIOS_CONSTR_2001A2005	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
28	N_EDIFICIOS_CONSTR_2006A2011	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
29	N_EDIFICIOS_ESTRUT_BETAO	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
30	N_EDIFICIOS_ESTRUT_COM_PLACA	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
31	N_EDIFICIOS_ESTRUT_SEM_PLACA	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
32	N_EDIFICIOS_ESTRUT_ADOME_PEDRA	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
33	N_EDIFICIOS_ESTRUT_OUTRA	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	
34	N_EDIFICIOS_ESTRUT_OUTRA	V_String	254	File: C:\Users\Duarte\Desktop\BGRI11_MAD.csv	

Figura 2.3: Alteryx Designer - BGRI11\_MAD

Porém tal como a aplicação acima mencionada, esta é uma ferramenta de auxílio a análise de dados que se enquadra na área do nosso trabalho, mas com um objetivo diferente que no nosso caso é o de recolha de meta-dados automaticamente.

<sup>2</sup><https://www.alteryx.com/products/alteryx-platform/alteryx-designer>

## Abordagem

O projeto a realizar contém três componentes principais: 1) a componente servidora que trata de receber e tratar pedidos de acordo com a lógica da aplicação, 2) a componente cliente que trata de apresentar uma interface web interativa ao utilizador e que comunica com a parte servidora e, por fim 3) a parte algorítmica de tratamento de ficheiros *csv*. Logo dividimos estes três pontos entre o grupo de modo a ser possível organizar o progresso em *use cases* de todas as componentes. A Figura 3.1 apresenta a arquitetura do projeto.

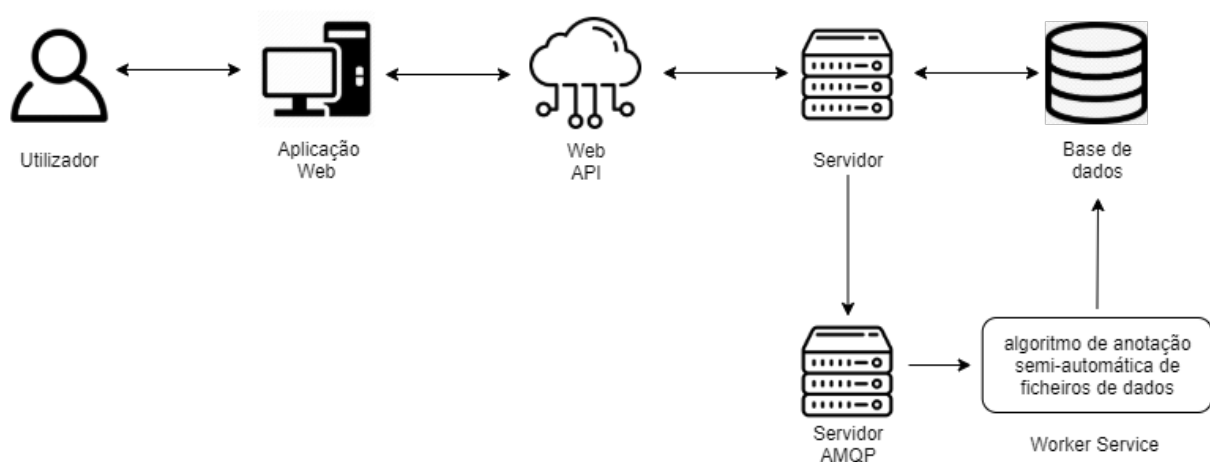


Figura 3.1: Arquitetura do Projeto

## 3.1 Análise de um ficheiro

Existem três principais aspectos relativos aos ficheiros que devem ser analisados automaticamente, nomeadamente a identificação de:

1. métricas e dimensões;
2. dados geográficos;
3. relações entre métricas e dimensões.

Os algoritmos que resolvem cada um dos aspectos acima foram idealizados por Ribeiro [3], tendo sido re-implementados neste trabalho. A próxima secção explica o racional por detrás de cada um.

### 3.1.1 Identificação de métricas e dimensões

De forma a distinguir se uma coluna consiste numa Dimensão ou numa Métrica foram usadas uma série de regras. Uma coluna é identificada como uma dimensão se ocorrer alguma das seguintes situações:

1. A coluna contém valores do tipo textual;
2. Todos os valores da coluna são diferentes;
3. O tipo de valores é numérico mas os valores apresentam uma ordenação perfeitamente definida;
4. Todos os valores da coluna são iguais.

Se nenhum dos casos anteriores ocorrer, então a coluna é identificada como Métrica, ou seja:

1. Se existirem valores numéricos do tipo decimais;
2. Se não existir uma ordenação definida entre os diferentes valores.

### 3.1.2 Identificação de dados geográficos

É comum encontrar dados relativos a localizações geográficas em diferentes conjuntos de dados, particularmente naqueles que retratam eventos resultantes de observações de actividade humana. A correcta identificação do domínio espacial, bem como da granularidade a que se encontram é aspecto importante para potenciar a a correcta comparação de dados provenientes de diferentes conjuntos e/ou fornecedores de dados [3].

De forma a identificar corretamente os domínios espaciais de um determinado ficheiro foi usada uma base de dados, denominada de *Knowledge Database* (KB). A KB representa três conceitos no seu modelo de dados) [3]:

1. locais geográficos, denominados de *Unidade Territorial*, que servem para definir uma área de terreno.
2. divisões geográficas de um determinado território, denominadas de *Divisão Territorial*
3. hierarquias geográficas, denominados de *Hierarquia Territorial*, que identificam e relacionam as diferentes granularidades geográficas existentes.

**Modelo de Dados** É apresentado na Figura 3.2 o modelo de dados do repositório Knowledge Database. Para o repositório foram carregados todas as instâncias das divisões administrativas de Portugal (nomes dos distritos, concelhos, freguesias e regiões autónomas), bem como as respectivas divisões (Distrito, Concelho, Freguesia).

É importante referir que a actual implementação da KB não contempla alterações na organização territorial ao longo do tempo.

### 3.1.3 Identificação de relações entre métricas

Como mencionado na introdução, por vezes o nome das colunas respeitantes a métricas traduzem algum tipo de relação entre elas.

Na Tabela 3.1 está um excerto de um ficheiro de censos realizado em 2011 para a população Madeirense. Como se verifica, o nome das métricas, N\_INDIVIDUOS\_PRESENT e N\_INDIVIDUOS\_PRESENT\_H apresentam uma prefixo comum que resulta na identificação de uma relação entre elas. Podemos então concluir que a última coluna consiste

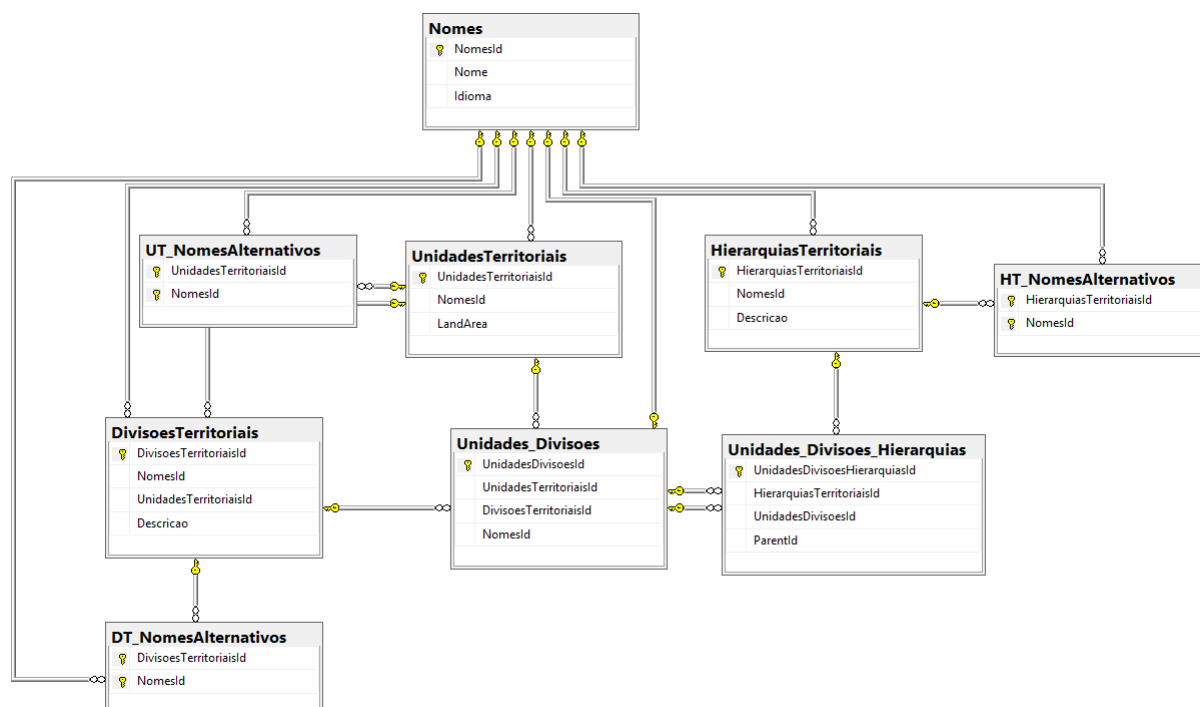


Figura 3.2: Modelo lógico da Base de Dados que suporta a KB - Análise (Adaptado de [3])

Tabela 3.1: Excerto censos 2011 - Madeira

ANO	GEO_COD_DSG	N_INDIVIDUOS_PRESENT	N_INDIVIDUOS_PRESENT_H
2011	Funchal	119423	55684
2011	Machico	21332	10321
2011	Ponta do Sol	8717	4001
2011	Porto Moniz	2643	1149

num nível de detalhe da anterior, onde a última representa o número total de indivíduos masculinos presentes numa determinada região e a coluna anterior representa o número de indivíduos presentes, masculinos e femininos, nessa mesma região.

### 3.1.3.1 Radix Tree

Para auxiliar na identificação destas relações, Ribeiro [3] implementou uma versão do algoritmo *Radix Tree*. Uma *radix tree* consiste numa árvore de prefixos com um passo extra. Esta árvore é composta por vários nós onde cada nó representa um carácter da palavra. Cada nó contém também todos os nós filhos que correspondem ao carácter seguinte da própria palavra.

Usando como exemplo as palavras *Bear*, *Bell*, *Stop*, *Stock* e *Be* a construção de uma *tree* com estas palavras processa-se da seguinte maneira: começando com a primeira

palavra, *Bear*, verifica-se se na raiz da árvore se existe algum nó com o carácter *B*. Como não existe devido à árvore estar vazia cria-se um nó com o carácter (Figura 3.3).

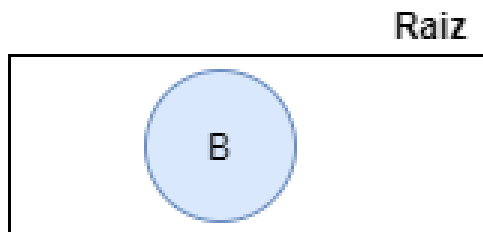


Figura 3.3: Construção da árvore - 1º carácter

A próxima iteração será colocar o segundo carácter da palavra *BEAR*, ou seja o *E*, e desta vez, em vez de procurar na raiz, procura nos filhos do carácter anterior, e como o *E* não existe ele é acrescentado como filho do *B* e assim sucessivamente para a palavra completa como pode ser observado na Figura 3.4

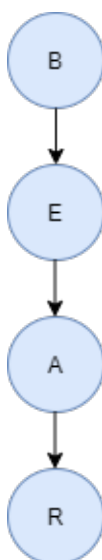


Figura 3.4: Construção da árvore - 1ª palavra

A próxima iteração consiste então em adicionar a palavra *BELL* que como o carácter *B* já existe na raiz da árvore este não é adicionado, idem para o carácter seguinte, *E*. É sim criado um novo filho no carácter *E* com as últimas duas letras da palavra. A árvore resultante pode ser observada na Figura 3.5.

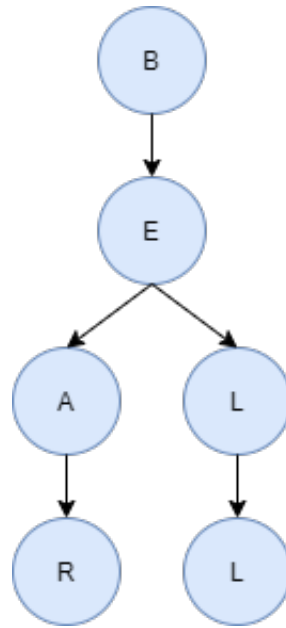


Figura 3.5: Construção da árvore - 2ª palavra

As próximas iterações para as palavras *Stop* e *Stock* seguem então a mesma lógica, como *S* não existe na raiz é criada uma nova árvore onde os 3 primeiros caracteres irão ser comuns. A árvore final pode então ser observada na Figura 3.6.

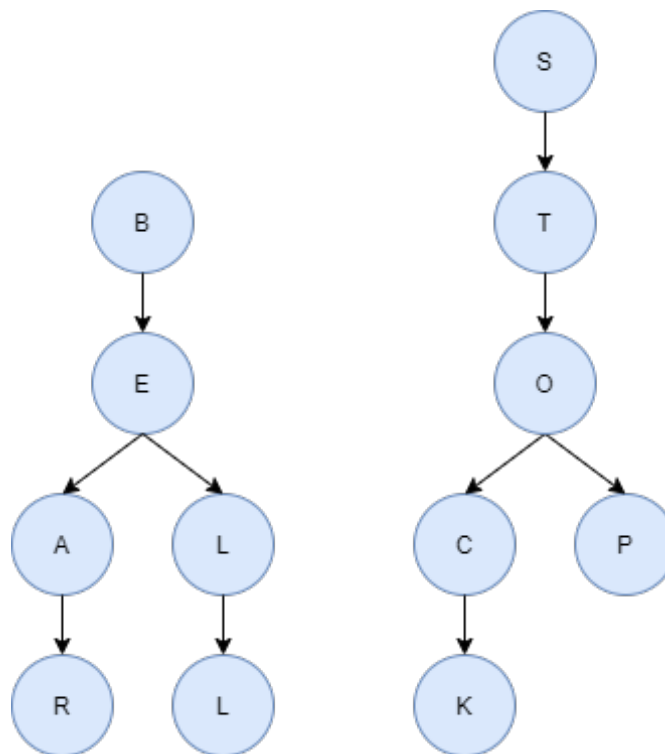


Figura 3.6: Construção da árvore - Árvore final



Podemos observar que não é possível distinguir onde acaba uma palavra existente, a palavra *Be* como é englobada pelas palavras *Bear* e *Bell* é indistinguível, logo para resolver esse problema a solução comum consiste em adicionar um caractere especial que significa a terminação de uma palavra. No entanto optámos por adicionar uma *flag* a cada nó que diz se o caminho até esse nó representa ou não uma palavra real. Na Figura 3.7 podemos observar o efeito dessa *flag* através dos nós em vermelho.

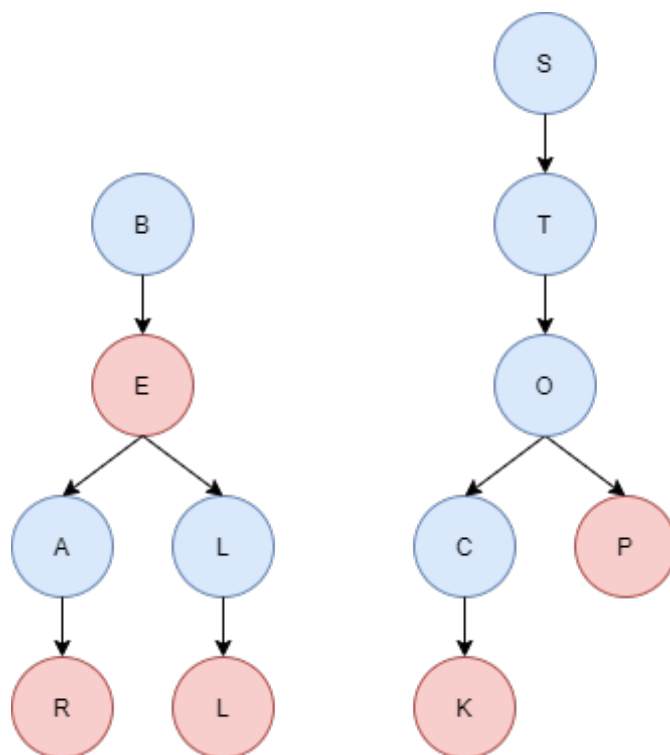


Figura 3.7: Construção da árvore - Árvore final com flag

O passo final consiste em agrupar certos nós considerados desnecessários sendo esses os nós que apenas contém um filho, o que permite poupança no espaço devido ao número inferior dos nós. A Figura 3.8 demonstra então a árvore final gerada pelo algoritmo implementado.

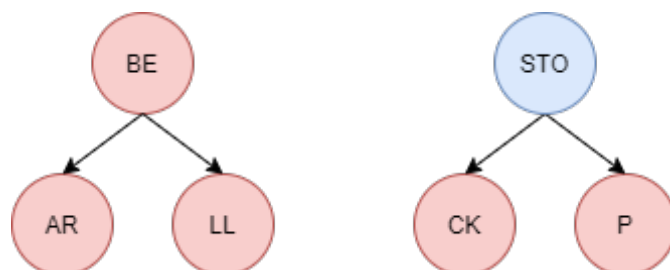


Figura 3.8: Construção da árvore - Árvore final gerada pelo algoritmo

### 3.1.4 Produto final da análise

O produto final da análise consiste então num ficheiro onde os dados resultantes estão representados em formato *json* de modo a proporcionar fácil leitura e análise tanto por parte de um humano como de um algoritmo. É apresentado na Listagem 3.1 um excerto do ficheiro gerado pelo algoritmo para o ficheiro de censos de 2011 para o arquipélago da Madeira.

```

1  {
2    "Nome": "BGRI11_MAD.csv",
3    "NumLinhas": 5901,
4    "NumColunas": 128,
5    "DataGeracao": "2020-06-07T20:54:45.2363137+01:00",
6    "GeoDivisoes": [
7      { "DivisoesTerritoriaisId": 5, "Tipo": "NUTS 0", "Linhas": [0] },
8      { "DivisoesTerritoriaisId": 6, "Tipo": "NUTS 1", "Linhas": [1] },
9      ...
10   ],
11   "Dimensoes": [
12     { "IndiceColuna": 0, "NomeColuna": "ANO", "NumValoresUnicos": 1, "
13       NumValoresNulos": 0,
14       "TodosDiferentes": false,
15       "TipoValores": [{"Tipo": "numeric", "Count": 5901}],
16       "ValoresUnicos": ["2011"],
17       "TipoDominioGeo": null
18     },
19     ...
20   ],
21   "Metricas": {
22     "Categorias": [
23       { "CategoriaId": 1, "Nome": "N_", "CategoriaPaiId": null },
24       { "CategoriaId": 2, "Nome": "N_EDIFICIOS_", "CategoriaPaiId": 1 },
25       ...
26     ],
27     "Colunas": [
28       { "IndiceColuna": 17, "NomeColuna": ".N_EDIFICIOS_50U MAIS_PISOS", "
29         CategoriaId": null, "E_Total": false },
30       { "IndiceColuna": 38, "NomeColuna": "N_CLASSICOS_RES_HABITUAL", "
31         CategoriaId": 1, "E_Total": false },
32       ...
33     ]
34   }
35 }

```

Listagem 3.1: Excerto de metadados gerados - BGRI11\_MAD.json

## 3.2 Casos de utilização suportados

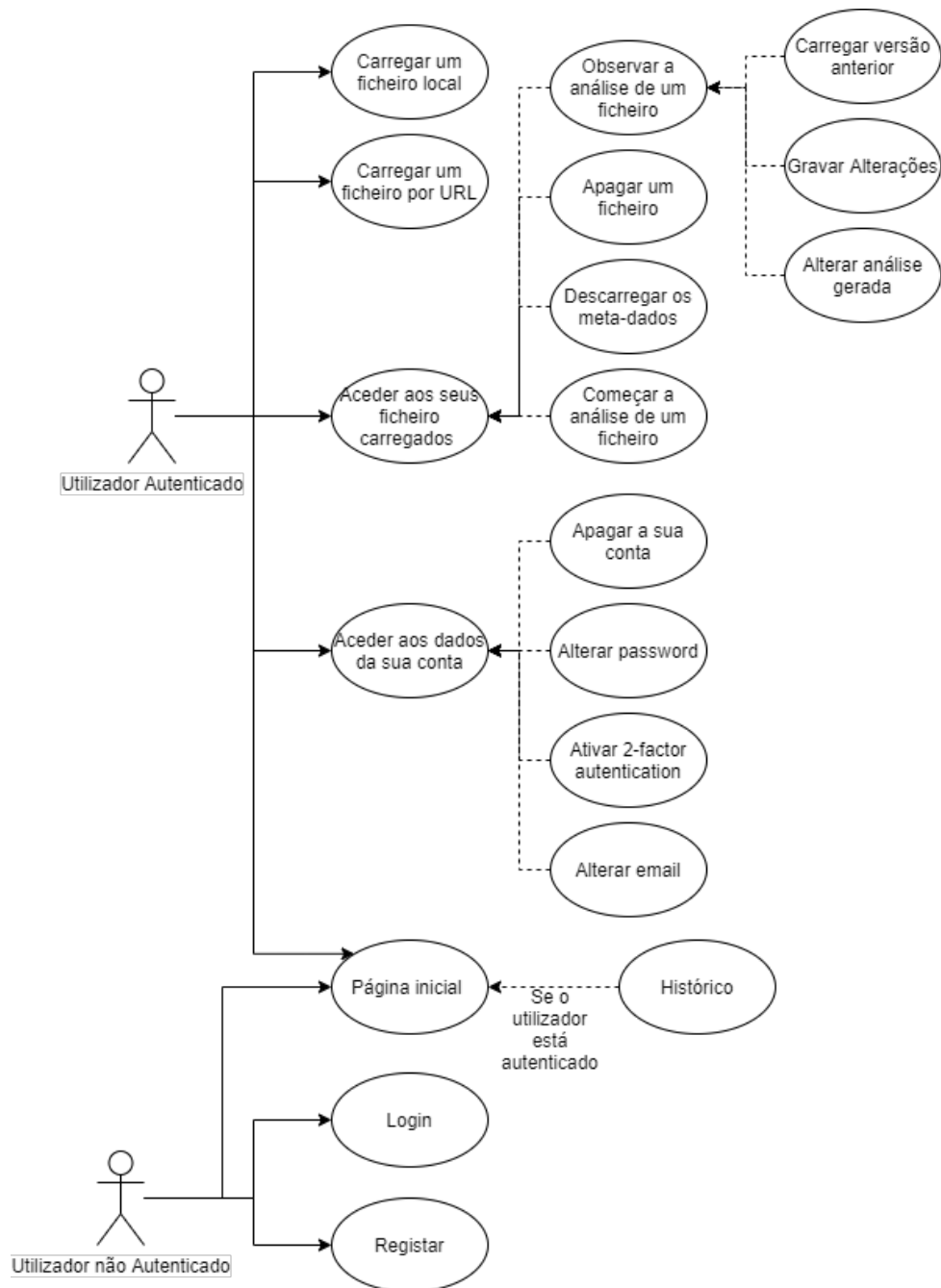


Figura 3.9: Casos de Utilização da aplicação

A aplicação suporta dois atores, um utilizador autenticado e um utilizador não autenticado, tendo ambos acesso a diversas ações.

**Registrar** A autenticação de um utilizador lida com dados sensíveis que nunca devem ser comprometidos. No entanto pouco se pode fazer em termos de segurança se o utilizador utiliza uma password simples como *pass123* que pode ser facilmente comprometida através de um simples ataque *brute force* [5]. Logo, o utilizador é forçado a utilizar diferentes tipos de caracteres e um tamanho mínimo de forma a dificultar tais ataques quando se regista na aplicação. Para um utilizador se registar este tem também que fornecer o seu email e o sistema envia um email a confirmar que o email pertence realmente a quem se está a registar, apenas dando acesso a funcionalidades que requerem autenticação após a verificação do email.

**Login** Para um utilizador se autenticar ele tem que fornecer o seu email e password. Após o sistema verificar as credenciais do utilizador se o utilizador fornecer credenciais validas o utilizador passa a ser um utilizador autenticado e é redireccionado para a página onde se encontrava. No caso de falha de autenticação é apresentada uma mensagem de erro e o utilizador pode voltar a repor as suas credenciais.

**Alterar email** Um utilizador autenticado tem a possibilidade de modificar o email associado à sua conta. Um email de confirmação é enviado ao novo email e o email associado à sua conta é alterado após confirmação.

**Ativar 2-factor authentication** Um utilizador autenticado tem a possibilidade de ativar *Two-factor-authentication* de forma a aumentar a segurança e integridade da sua conta [6]. A aplicação fornece ao utilizador um código ou um *QR code* que o utilizador deve utilizar numa aplicação como *Microsoft Authenticator* para *Android* ou *iOS* ou *Google Authenticator* para os mesmos sistemas operativos. Um código de verificação é dado ao utilizador por uma dessas aplicações que deve ser colocado na nossa aplicação completando o caso de utilização.

**Alterar password** Um utilizador autenticado tem também a possibilidade de alterar a password. Este deve fornecer a sua password corrente para verificar novamente que é o próprio utilizador a tentar alterá-la. Ele deve então fornecer a nova password duas vezes de forma a prevenir contra eventuais erros de escrita na nova password. Se a password antiga estiver correta e as duas novas passwords forem iguais a password do utilizador é então alterada, se alguma dessas verificações falhar aparece uma mensagem de erro.

**Apagar a sua conta** Um utilizador autenticado tem também a possibilidade de alterar a password. Este deve fornecer a sua password corrente para verificar novamente que é o próprio utilizador a tentar alterá-la. Ele deve então fornecer a nova password duas vezes de forma a prevenir contra eventuais erros de escrita na nova password. Se a password antiga estiver correta e as duas novas passwords forem iguais a password do utilizador é então alterada, se alguma dessas verificações falhar aparece uma mensagem de erro.

**Página Inicial - Histórico** Na página inicial da aplicação, ao utilizador não autenticado é apresentado um resumo sobre a aplicação e as suas funcionalidade.

Se o utilizador estiver autenticado a página inicial contém um histórico sobre as últimas ações realizadas pelo utilizador de forma a facilitar um utilizador que não aceda à aplicação à vários dias se lembre das suas ações passadas contendo também um *link* para ir diretamente à página onde se encontrava.

**Carregar um ficheiro local** Este caso de utilização permite a um utilizador autenticado carregar ficheiros locais na aplicação através de *drag'n'drop* de um ou mais ficheiros numa zona dedicada na aplicação, ou fazendo *browse* localmente. A zona de *drag'n'drop* apenas aceita ficheiros *csv*, no entanto, o servidor verifica novamente se o ficheiro é realmente *csv*. Uma barra de progresso aparece enquanto um ficheiro está a ser carregado para o servidor mostrando, em percentagem, o progresso do carregamento. Após o ficheiro ser carregado por completo, é mostrado na página que o ficheiro foi aceite, caso não tenha existido nenhum erro, ou é mostrado que foi rejeitado se ocorreu algum erro durante o carregamento.

**Carregar um ficheiro por URL** Este caso de utilização permite a um utilizador autenticado um ficheiro através de um *URL* que pode, ou não, ter como proveniência uma página com autenticação. Neste cenário o sistema deve conseguir tratar métodos de autenticação como *OAuth 2.0* [7] pois o sistema não deve pedir a password do utilizador para essa página. O servidor tem também que verificar se o ficheiro é realmente *csv* e, similarmente ao *drag'n'drop*, o ficheiro aparece na zona de ficheiros aceites ou rejeitados, dependendo da ocorrência de algum erro. O utilizador pode depois observar que ficheiros tem na aplicação no seu ambiente de trabalho, apresentado na secção seguinte.

**Aceder aos seus ficheiros carregados** Este caso permite ao utilizador aceder a todos os seus ficheiros carregados. Clicando sobre um deles informação sobre o ficheiro é

mostrada ao lado, nomeadamente o tamanho, quando é que foi carregado, de onde foi carregado (localmente ou por URL) e, se já foi analisado, o tempo que a análise demorou e a data a que ela foi feita. Este caso também dá acesso a 4 outros casos que vão ser explicados nos próximos parágrafos.

**Começar a análise de um ficheiro** Clicando sobre um ficheiro que ainda não foi analisado, é então possível começar a sua análise. Esta análise quando terminada dá acesso a novas funcionalidades.

**Descarregar os meta-dados** Clicando sobre um ficheiro onde a análise já foi feita dá a possibilidade ao utilizador de fazer download dessa análise.

**Observar a análise de um ficheiro** Este caso deixa observar a análise automática realizada pelo algoritmo (meta-informação), podendo verificar quais as colunas que foram identificadas como métricas e aquelas que foram identificadas como dimensões. Este caso dá acesso a 3 outros casos de utilização descritos nos próximos parágrafos.

**Gravar Alterações e Carregar versão anterior** Este caso fornece a possibilidade de gravar as alterações feitas pelo utilizador sobre a análise ficando essa a versão mais recente. O caso de carregar a versão anterior deixa o utilizador carregar uma versão mais antiga gravada anteriormente.

**Alterar análise gerada** A atribuição automática gerada pelo algoritmo pode ser alterada, movendo colunas entre listas, através de *drag'n'drop*. A apresentação das métricas e dimensões é feita através de uma estrutura em árvore, fornecendo ao utilizador uma melhor compreensão das relações de nível entre as diferentes colunas. Similarmente ao que acontece com as métricas e dimensões, também é possível alterar o nível de detalhe duma métrica no caso de um eventual erro de atribuição por parte do algoritmo.

### 3.2.1 Estrutura da interface visual

Todas as páginas da aplicação têm no topo da página uma barra de navegação com todas as funcionalidade principais da aplicação para fácil navegação entre elas.

Cada uma das páginas engloba um ou mais dos casos de utilização mencionados acima, estas páginas são:

1. *Home*, página inicial da aplicação que contém os casos **Página Inicial** e **Histórico**;
2. *Register*, para registar um novo utilizador que contém o caso **Registar**;
3. *Login*, para autenticar um utilizador registado que contém o caso **Login**.
4. O email de um utilizador autenticado, para alterar dados da sua conta que contém os casos **Aceder aos dados da sua conta** e os seus casos filhos;
5. *Upload file*, onde o utilizador pode carregar ficheiros para a aplicação que contém os casos **Carregar um ficheiro local** e **Carregar um ficheiro por URL**
6. *My Workspace*, onde o utilizador pode realizar operações sobre os ficheiros carregados que contém o caso **Aceder aos seus ficheiros carregados** e os seus casos filhos;





# 4

## Implementação

### 4.1 Tecnologias

As metodologias usadas no lado do servidor são focadas em análise de documentos com formato *csv* e tratamento de texto, tornando a escolha da linguagem de programação um aspeto menos crítico, uma vez que qualquer linguagem têm as capacidades necessárias para implementar as metodologias. No entanto, foi objectivo utilizar tecnologias *open-source* de forma a não existirem custos de licenciamento. Foi também dada a preferência a tecnologias que permitam desenvolver a solução para ser *cross-platform* para que as nossas aplicações não estejam limitadas quanto às máquinas, e respetivos sistemas operativos, em que podem correr. Além disso, foi integrado código disponibilizado por Ribeiro [3], nomeadamente, o algoritmo de geração de meta informação, desenvolvido em C#. Para além destas restrições, foi tido em consideração a nossa experiência passada, bem como as preferências pessoais. A ferramenta utilizada para desenvolver a aplicação servidor, bem como toda a lógica aplicacional foi a *framework* .NET Core.

Para a implementação da base de dados foi usado *SQLServer*. A tecnologia do lado do cliente da aplicação web, responsável por implementar a interface com o utilizador, foi desenvolvida usando *JavaScript*, recorrendo à *framework* *React*.

## 4.2 Autenticação - Identity Server

De forma a implementar a autenticação de maneira segura utilizámos a *framework IdentityServer* [8]. Esta adiciona automaticamente todos os *endpoints* necessários para o registo, autenticação e gestão de conta para um utilizador. De forma a guardar localmente os utilizadores registados, o *IdentityServer* necessita de uma base de dados local, tipicamente *SQL Server*. Na Figura 4.1 podemos observar o modelo de dados oferecido pelo *IdentityServer*.

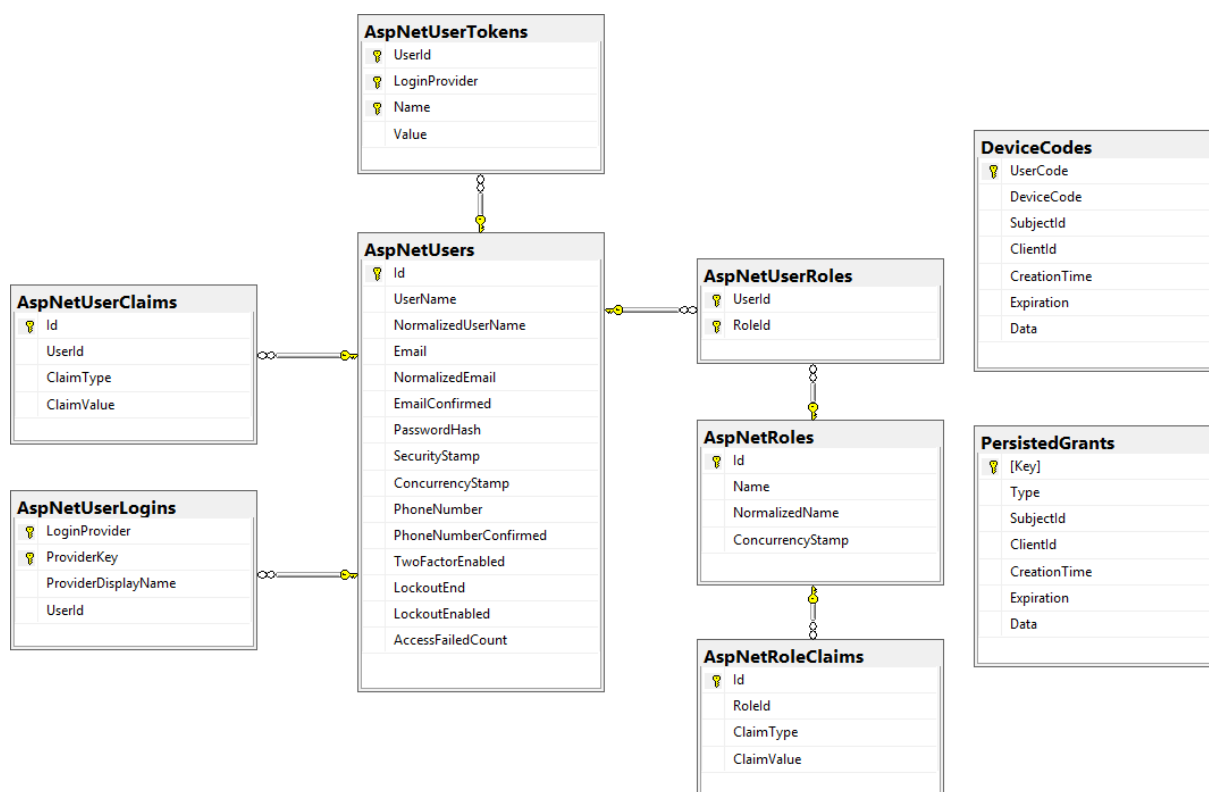


Figura 4.1: Modelo da Base de Dados - IdentityServer

A tabela *AspNetUsers* é a mais relevante pois é onde são armazenados os dados do utilizador como o seu email, password (*hashed*) entre outros. A geração do *hash* inclui *salt* de forma a duas passwords idênticas não gerarem o mesmo *hash* e o algoritmo também realiza o *hash* várias vezes, de forma iterativa, de maneira a que um atacante que esteja a tentar fazer *bruteforce* [5] tenha também de realizar o *hash* o mesmo número de vezes que a aplicação realiza. Levando a que tempo necessário para cada password tentada aumenta linearmente.

## 4.3 Página inicial

A aplicação foi desenvolvida como *Single Page Application* (SPA). Uma SPA é uma aplicação WEB que contém uma única página WEB e que usa *JavaScript*, *HTML5*, e *CSS* para a interação com o utilizador. Numa SPA, após o carregamento inicial, não existe mais nenhum carregamento total durante o seu uso [9]. A interação da página com o utilizador vai iniciar processos de carregamento parciais, executados em segundo plano, usando *JavaScript*. A SPA faz *render* das páginas diretamente no *browser*, isto é facilitado graças às estruturas JavaScript avançadas, como *AngularJS*, *Ember.js*, *Meteor.js*, *Knockout.js* e *ReactJS*. Este tipo de aplicações está presente em várias aplicações web tais como *Gmail*, *Google Maps*, *Facebook* ou *GitHub*. Uma SPA ajuda a manter o utilizador num espaço confortável, onde o conteúdo é apresentado ao utilizador de maneira simples, fácil e rapidamente [10]. Tendo isto em mente implementámos a nossa aplicação como uma SPA de modo a cumprir os requisitos referidos na secção 3.2.1.

Na Figura 4.2 podemos observar a página inicial da aplicação para um utilizador autenticado<sup>1</sup>.

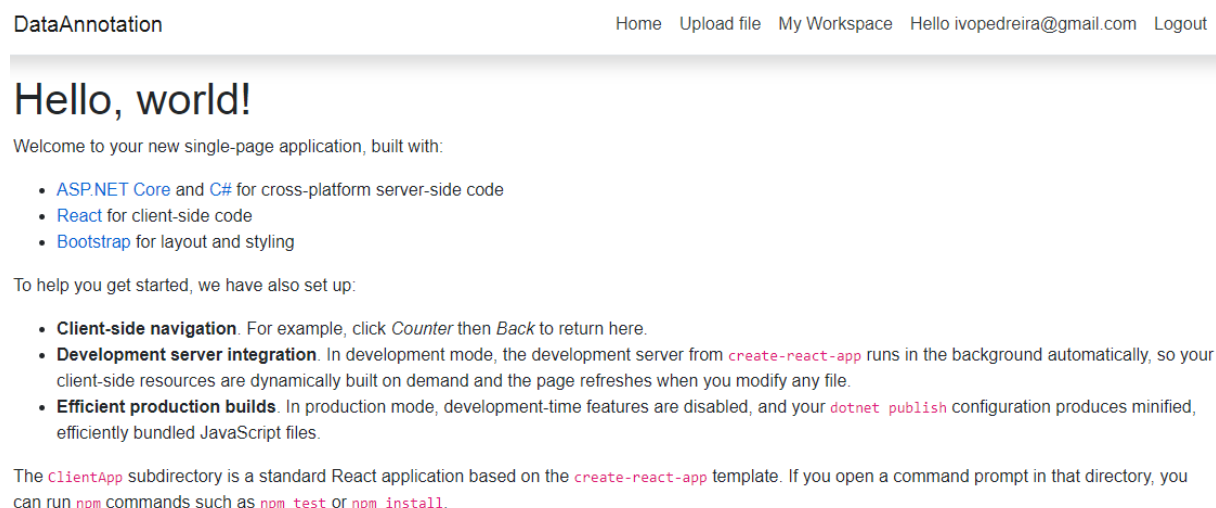


Figura 4.2: *Home Page* da aplicação

<sup>1</sup>A estética da página não foi tida em conta até ao momento, sendo algo a melhorar após terem sido implementados os requisitos principais da aplicação.

## 4.4 Carregamento de ficheiros

A aplicação permite efectuar o *upload* de ficheiros directamente da máquina do utilizador, ou através de um endereço (URL) fornecido pelo utilizador.

Independentemente da fonte de onde o ficheiro foi obtido, este, uma vez carregado na aplicação, é guardado no *file system* do servidor após serem efectuadas algumas verificações. Os ficheiros são guardados com um nome gerado aleatoriamente pela aplicação de modo a melhorar a segurança e simplicidade [11]. Todos os ficheiros de um utilizador são guardados numa pasta cujo nome é o *id* desse utilizador. Na base de dados, foram mapeados os nomes aleatórios com os nomes originais, assim como algumas meta-informações sobre cada ficheiro, tais como o tamanho ou a origem do ficheiro. Na Figura 4.3 podemos observar o modelo de um ficheiro.

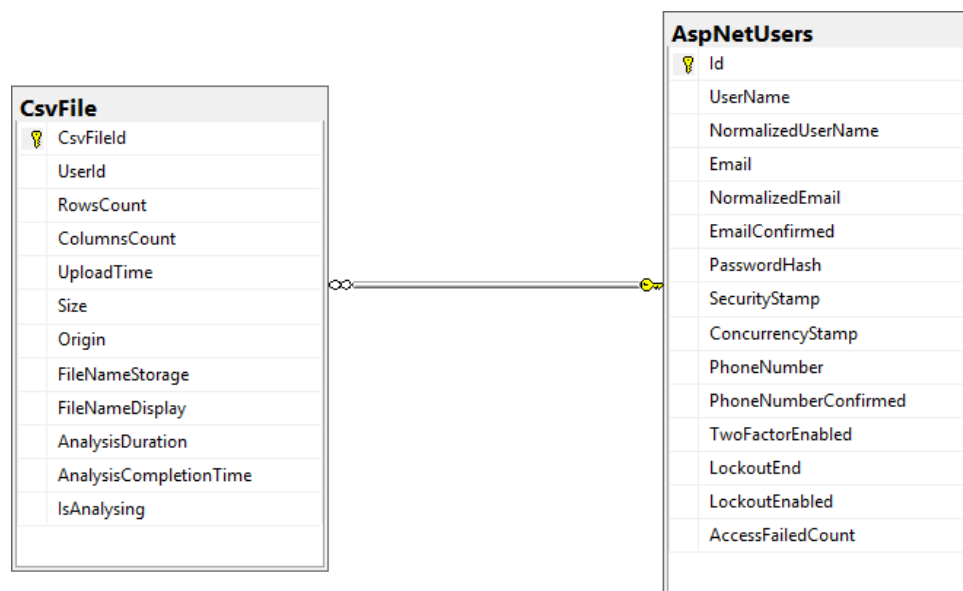


Figura 4.3: Modelo da Base de Dados - CsvFile

Na Figura 4.4 podemos observar a página para carregamento de ficheiros com um ficheiro aceite pelo servidor, um ficheiro rejeitado devido a não ser um ficheiro *csv* e um ficheiro a ser carregado para o servidor.

### 4.4.1 Client Side

A parte cliente para carregamento de ficheiros foi implementada através da biblioteca *React Dropzone* que facilita a criação e customização de uma zona de *drag'n'drop* [12], de modo a só serem aceites ficheiros *csv*.

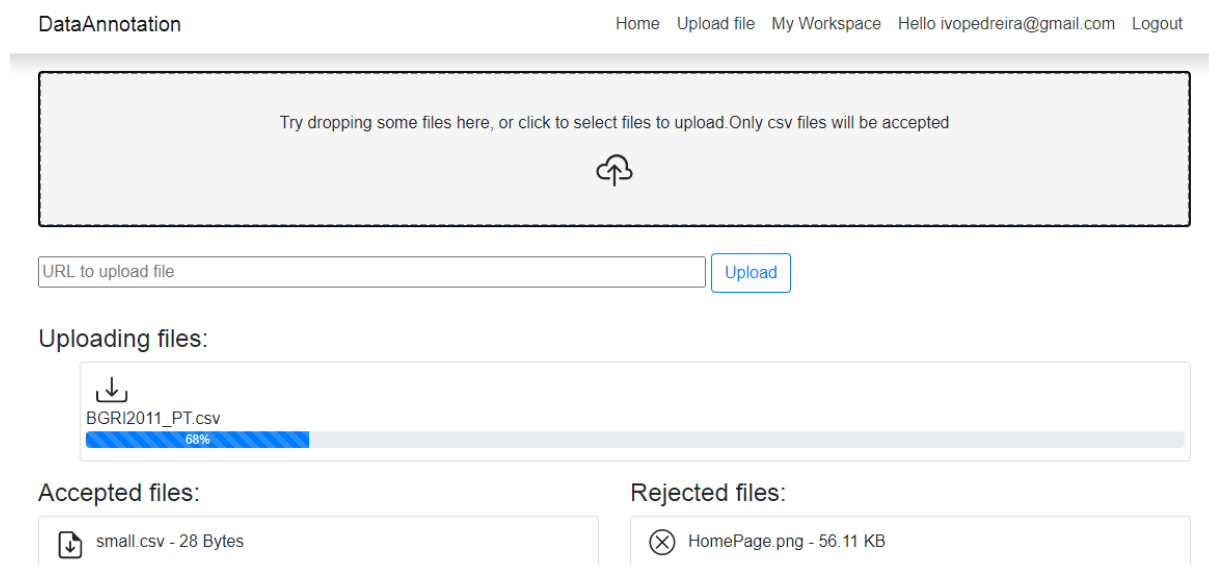


Figura 4.4: Página para carregar ficheiros

Porque lidamos com upload de ficheiros, dava-nos jeito saber o progresso do upload, principalmente em ficheiros de grandes dimensões, por essa razão foi usado a biblioteca *axios* [13], que permite fazer pedidos HTTP com a peculiaridade de ser possível saber a quantidade de data já recebida em comparação ao enviado.

Esta vista também fornece a informação de quais os ficheiros que foram aceites e rejeitados dentro dos que foram carregados. Por exemplo se se arrastar um ficheiro csv vazio para a zona de *drag'n'drop*, este é carregado, no entanto será rejeitado aparecendo na zona dedicada aos ficheiros rejeitados.

#### 4.4.2 Server Side

Numa das opções para upload de ficheiro o servidor recebe um ficheiro proveniente do cliente em formato *multipart*. É aberto um *stream* com o conteúdo que vem no pedido, *stream* esse que é guardado no servidor como ficheiro. É utilizado um *stream* ao invés de um *buffer* porque a nossa aplicação aceita ficheiros de grande dimensão e um *buffer* pode utilizar demasiada memória [14].

Como segunda opção o servidor recebe no pedido o *uri* do ficheiro que tem de efectuar o download. São feitas as verificações de forma a afirmar que o ficheiro é um ficheiro aceitável pela aplicação e de seguida é aberto um *stream* desse endereço e guardado como ficheiro no servidor.

## 4.5 Ambiente de trabalho

No ambiente de trabalho o utilizador tem acesso aos seus ficheiros carregados, bem como os detalhes dos mesmos.

É possível então realizar um conjunto de operações sobre esses ficheiros, sendo elas:

1. A análise do ficheiro;
2. Remoção do ficheiro;
3. *Download* da meta-data gerada pela análise;
4. Ir para uma página que permite visualizar e alterar a meta-data gerada.

Estas operações vão ser explicadas em maior detalhe nas secções seguintes.

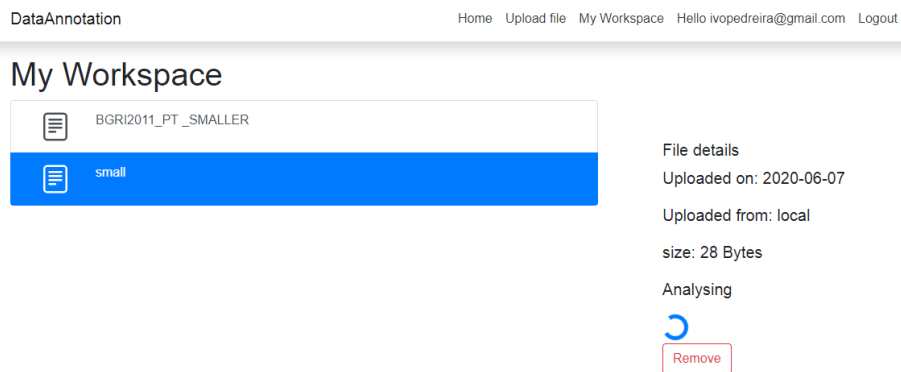
Nas Figuras 4.5a e 4.5b podemos observar a página do ambiente de trabalho de um utilizador, a primeira apresenta um ficheiro com a análise a decorrer e na segunda um ficheiro com a análise completa.

### 4.5.1 Análise de ficheiros

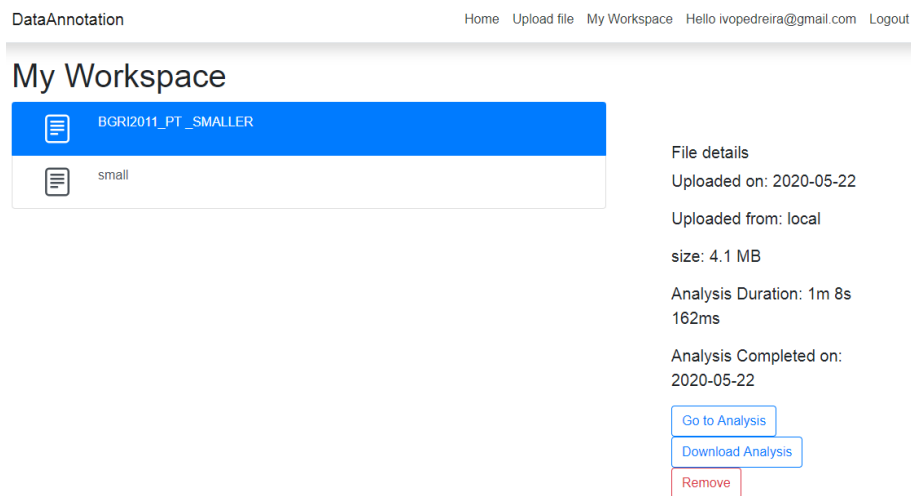
A aplicação tem como foco principal a análise de ficheiros *csv*, e a colaboração do utilizador para corrigir ou alterar a análise.

#### 4.5.1.1 Client Side

Sempre que um utilizador opta por pedir uma análise, o *client* irá reencaminhar esse pedido ao servidor. Cada vez que uma análise é iniciada é criado um *setInterval* que de 5 em 5 segundos realiza um pedido ao servidor a perguntar se a análise já terminou. O intervalo de tempo pode numa versão futura ser uma variável que dependendo do tamanho do ficheiro realiza o pedido com um intervalo menor ou maior. Este intervalo apenas termina quando o servidor devolver que a análise já terminou alterando então a vista com os novos dados. No entanto, os pedidos também terminam se o utilizador mudar de página pois não é necessário saber se a análise terminou estando noutra página. Quando o utilizador volta a aceder à página do workspace se a análise ainda não terminou o intervalo é criado novamente. A Listagem 4.1 apresenta o método responsável pela criação e lógica do intervalo.



(a) Análise a decorrer



(b) Análise finalizada

Figura 4.5: Ambiente de Trabalho

```

73  async checkAnalysisStatus(id, token) {
74      var array = this.state.files
75      var requestLoop = setInterval(function () {
76          fetch(`Workspace/IsAnalysisComplete?fileId=${id}`, {
77              method: 'GET',
78              headers: !token ? {} : { 'Authorization': `Bearer ${token}`
          }
79      }).then(res => {
80          if (res.status !== 204) { //204 = empty response -> not yet
            completed analysis
81              res.json().then(newFile => {
82                  array.forEach(f => {
83                      if (f.csvFileId === id) {
84                          f.analysisCompletionTime = newFile.
analysisCompletionTime

```

```

85         f.analysisDuration = newFile.analysisDuration
      analysisDuration
86         f.rowsCount = newFile.rowsCount
87         f.columnsCount = newFile.columnsCount
88         f.isAnalysing = newFile.isAnalysing
89     }
90 })
91 stopLoop(array)
92 })
93 }
94 })
95 }, 5000); //5 seconds
96 this.setState({ requestLoops: this.state.requestLoops.concat(
requestLoop) })
97
98 var stopLoop = (newArray) => {
99     this.setState({ files: newArray })
100     this.forceUpdate()
101     clearInterval(requestLoop)
102 }
103 }

```

Listagem 4.1: Método para verificar se a análise já terminou

#### 4.5.1.2 Server Side

A análise de um ficheiro é realizada através de um algoritmo, aproveitando o trabalho de Ribeiro como mencionada na secção 3.1. Este algoritmo analisa informações como os nomes das colunas do ficheiro *csv* e os próprios dados dos ficheiros, pelo que uma análise a um ficheiro de grandes dimensões pode levar a uma análise que dure um tempo considerável, podendo chegar mesmo a demorar vários minutos. Para não degradar desempenho da aplicação e não prejudicar a interface com o utilizador criámos um serviço que correrá em paralelo com a aplicação e a análise do ficheiro será realizada nesse serviço.

A nossa aplicação irá comunicar com o serviço de análise através de uma fila de mensagens. O serviço de análise irá estar à escuta, como consumidor na fila, por instruções provenientes da aplicação. Cada mensagem irá ter o id do ficheiro e a localização desse ficheiro. O id para que o serviço de análise possa aceder e alterar informação sobre esse ficheiro na base de dados. A localização para que o serviço possa aceder a esse ficheiro para leitura. Por fim a aplicação, para saber se a análise está completa, irá verificar



a existência de um ficheiro resultante da análise. A funcionalidade deste sistema depende, não só, de que ambos os serviços (aplicação e serviço de análise) tenham acesso à mesma base de dados, como também tenham um *file system* partilhado.

Esta dependência por um *file system* partilhado pode prejudicar a escalabilidade da aplicação devido ao facto que desta forma têm de correr na mesma máquina. A nossa solução utiliza um *file system* partilhado pois o serviço de análise precisa de acesso ao ficheiro para o analisar. Devido à possibilidade e probabilidade de os ficheiros serem de grandes dimensões, chegando à ordem dos gigabytes, não seria viável ter que enviar este ficheiro da aplicação para o serviço de análise.

## 4.5.2 Remoção de ficheiros

A remoção de um ficheiro pode em primeira vista parecer algo banal, mas, devido às várias componentes que um ficheiro pode ter, nomeadamente todas as diferentes versões da análise desse ficheiro, a sua entrada na base de dados e o ficheiro em si, a remoção de um ficheiro pode tornar-se não linear devido a erros que possam acontecer em cada uma dessas remoções. Tendo sempre como principal objetivo manter consistente a base de dados e o *file system*.

Como a parte cliente da aplicação apenas realiza o pedido de remoção e atualiza a vista de acordo com o resultado, apenas a parte servidora será explicada em maior detalhe.

### 4.5.2.1 Server Side

Devido à necessidade de manter a base de dados e o *file system* consistentes foi necessário garantir que no caso de erro na remoção da entrada na base de dados o ficheiro não seria mesmo eliminado, ou caso ocorra um erro na eliminação do ficheiro a entrada desse ficheiro na base dados seja repostada. Na Listagem 4.2 pode ser observado o método de remoção de um ficheiro e os casos em que não é bem sucedido.

```
1 [ HttpDelete ]
2 public IActionResult RemoveFile([FromQuery] int fileId){
3     var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
4     CsvFile file = _context.CsvFile.Where(f => f.UserId == userId
5         && f.CsvFileId == fileId).FirstOrDefault();
6     if (file.CsvFileId == 0) return NotFound();
7     string fileFolderPath = Path.Combine(_targetFilePath,
8         Path.Combine(userId, file.FileNameStorage));
9     try{
10         _context.CsvFile.Remove(file);
```

```

11         _context.SaveChanges();
12         System.IO.Directory.Delete(fileFolderPath, true);
13     } catch (DbUpdateException e) {
14         return StatusCode(500, e);
15     } catch (Exception e) {
16         _context.CsvFile.Add(file);
17         _context.SaveChanges();
18         return StatusCode(500, e);
19     } finally { _context.Dispose(); }
20     return Ok();
21 }

```

Listagem 4.2: Método que trata de remoção de um ficheiro

### 4.5.3 Download da meta-data

Um dos requisitos funcionais consiste então em ser possível fazer *download* da meta-data gerada. A implementação por parte do servidor é bastante direta, apenas devolve o ficheiro *.json* com a análise, logo, apenas a parte cliente vai ser explicada em mais detalhe.

#### 4.5.3.1 Client Side

Quando o utilizador pressiona o botão para fazer *download* do ficheiro, é feito o pedido ao servidor para obter a análise. De seguida, é feito o *blob* da análise (*Binary Large Object*) finalizando com a criação de um novo elemento html, *a*, que contém os campos *href* que aponta para o *blob* em si e o campo *download* para fazer download do ficheiro. A Listagem 4.3 apresenta o código responsável pelo *download* do ficheiro da análise.

```

109     async DownloadAnalyzis(id, fileName) {
110         const token = await authService.getAccessToken();
111
112         fetch(`Workspace/DownloadAnalysis?fileId=${id}`, {
113             method: 'GET',
114             headers: !token ? {} : { 'Authorization': `Bearer ${token}` }
115         }).then(response => {
116             response.blob().then(blob => {
117                 let url = window.URL.createObjectURL(blob);
118                 let a = document.createElement('a');
119                 a.href = url;

```

```

120         a.download = fileName.split('.')[0] + '_analysis' + '.json'
121     ;
122     a.click();
123     });
124     });
125 }

```

Listagem 4.3: Método que realiza o *download* da análise

#### 4.5.4 Análise

A vista da análise de um ficheiro é constituído por 4 divisões, uma que contém o nome do ficheiro e a possibilidade de gravar a análise, fazer *load* da uma análise previamente guardada ou *download* da mesma, outra onde estão as dimensões, outra com as métricas (cuja análise não determinou a que categoria pertencem) e por ultimo a divisão com todas as categorias e suas colunas.

Dentro das categorias o utilizador tem a opção de poder ordenar as colunas ou até mesmo move-las para outra categoria ou para outra divisão.

Na Figura 4.6 podemos observar a página de análise de um ficheiro, neste caso do ficheiro de censos da Madeira.

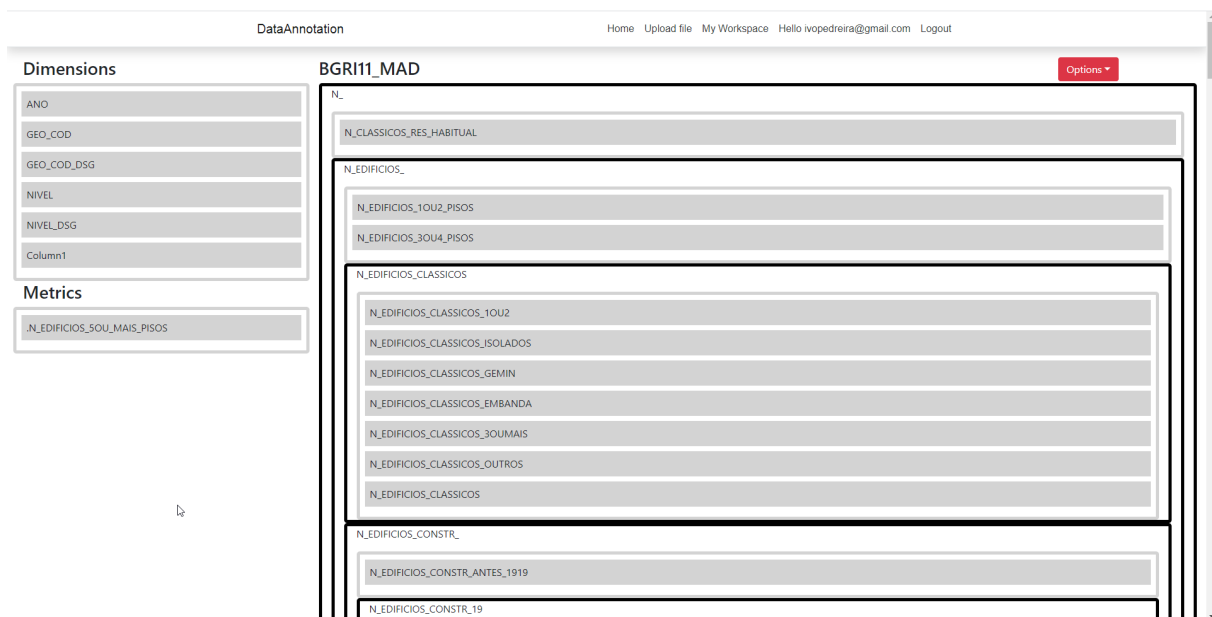


Figura 4.6: Análise do ficheiro de censos - BGRI11\_MAD

#### 4.5.4.1 Client Side

O lado cliente recebe a versão mais recente da meta-data do ficheiro em formato *json* como apresentado na secção 3.1.4. Uma das secções desse ficheiro consiste na secção das Categorias dentro da secção das Métricas, na Listagem 4.4 é apresentado um excerto dessa secção para o mesmo ficheiro apresentado anteriormente.

```
1  ...
2  "Metricas": {
3      "Categorias": [
4          {
5              "CategoriaId": 1,
6              "Nome": "N_",
7              "CategoriaPaiId": null
8          },
9          {
10             "CategoriaId": 2,
11             "Nome": "N_EDIFICIOS_",
12             "CategoriaPaiId": 1
13         },
14         {
15             "CategoriaId": 3,
16             "Nome": "N_EDIFICIOS_CLASSICOS",
17             "CategoriaPaiId": 2
18         },
19         {
20             "CategoriaId": 4,
21             "Nome": "N_EDIFICIOS_CONSTR_",
22             "CategoriaPaiId": 2
23         },
24         {
25             "CategoriaId": 5,
26             "Nome": "N_EDIFICIOS_CONSTR_19",
27             "CategoriaPaiId": 4
28         },
29         ...
30     ],
31 },
32 ...
```

Listagem 4.4: Excerto da metadata - BGRI11\_MAD.csv - Categorias

No excerto podemos observar que cada categoria possui o campo *CategoriaPaiId*, esse campo dá-nos a informação que, por exemplo, a categoria *N\_EDIFICIOS\_CLASSICOS*

consiste num nível de detalhe da categoria `N_EDIFICIOS_`. Tendo essa informação optámos por transformar esses dados numa árvore n-ária para ser mais fácil desenvolver a vista necessária.

É também importante saber que métricas pertencem a que categoria, para isso, o ficheiro de metadados contém a que categoria certa métrica pertence através do campo `CategoriaId` que associa com o campo respetivo da categoria. Logo, aproveitando o algoritmo que desenvolve a árvore das categorias, é adicionado um campo a cada nó da árvore chamado *columns* que contém uma lista de métricas pertencentes à sua categoria. Na Figura 4.7 está representado o esquema da árvore resultante.

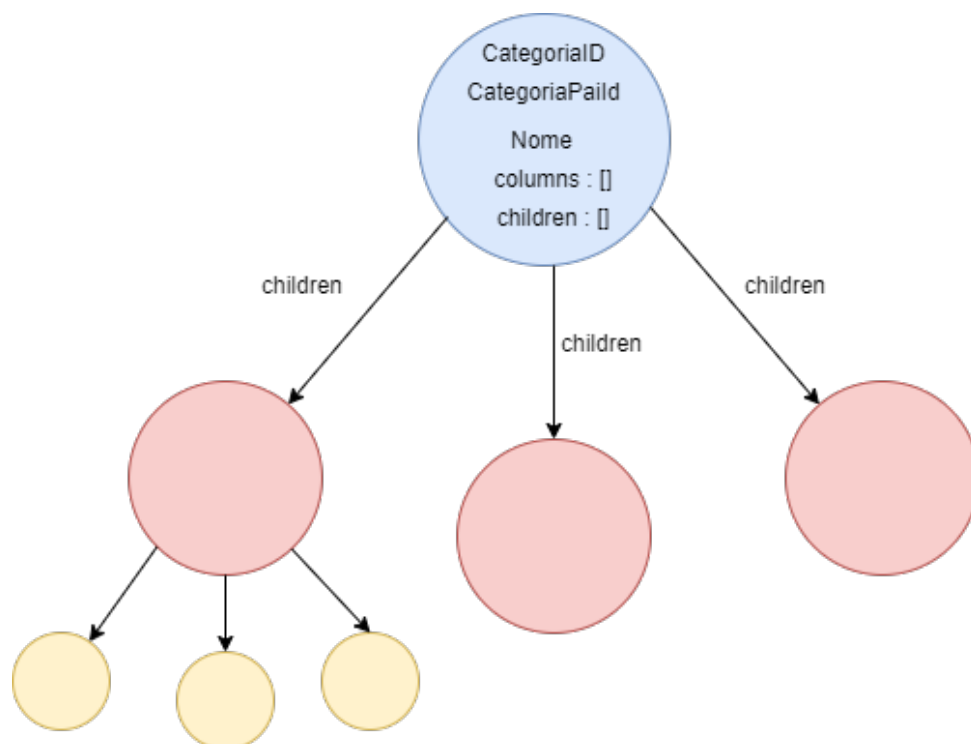


Figura 4.7: Árvore n-ária resultante da organização da meta-data

Para fornecer a possibilidade de arrastar uma categoria ou dimensão para outra divisão válida foi utilizada a biblioteca *react-beautiful-dnd* [15]. Esta biblioteca fornece dois componentes principais, *droppable* e *draggable*. Um *droppable* consiste numa zona da página onde é possível largar um ou mais componentes *draggable*, um *draggable* consiste num componente que possa ser arrastado e largado no mesmo ou não *droppable*.

Assim, cada categoria consiste numa zona *droppable* que é inicializada com as suas colunas, onde cada uma é um componente *draggable*. A árvore criada anteriormente é então transformada nesses componentes através dum método recursivo. Na Figura 4.8 pode ser observado uma visualização da vista final onde os componentes *droppable* estão representados em vermelho e os *draggable* em azul. É também possível aferir que

os níveis 3 são níveis de detalhe do nível 2 e que o nível 2, e o por sua vez os níveis 3, são detalhes do nível 1.

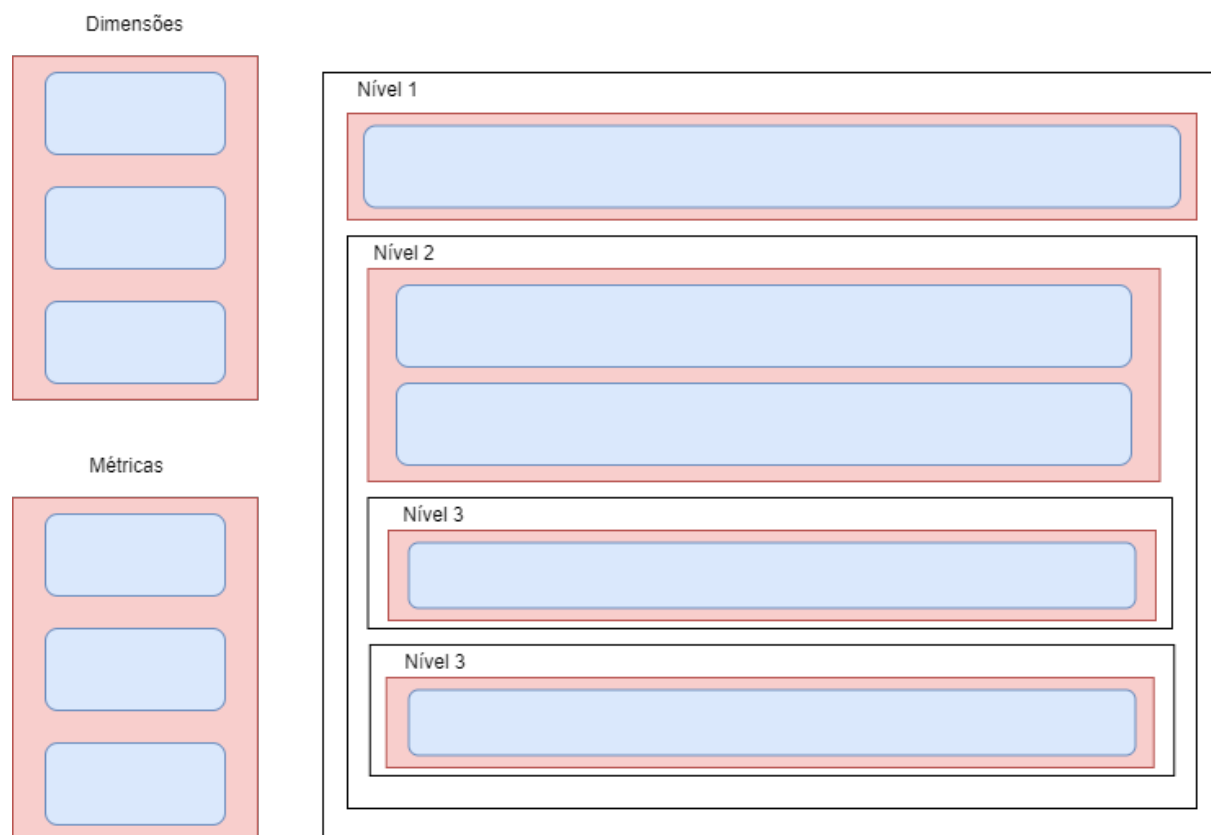


Figura 4.8: Representação da vista final



## **Avaliação**

A desenvolver durante a realização do projecto.







## **Conclusão**

O plano está a ser cumprido de acordo com o delimitado.



# Referências

- [1] Ciprian Dobre e Fatos Xhafa, “Intelligent services for big data science”, *Future Generation Computer Systems*, vol. 37, pp. 267–281, 2014.
- [2] Y. Shafranovich. (2005). Common Format and MIME Type for Comma-Separated Values (CSV) Files: RFC 4180, URL: <https://tools.ietf.org/html/rfc4180>. Accessed: 04-04-2020.
- [3] Nuno Ribeiro, “Anotação e Extração Semi-Automática de dados multidimensionais”, Tese de Mestrado, Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia, Portugal, 2019.
- [4] Pedro Amaral, “Smart Geo-Catalog”, Tese de Mestrado, Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia, Portugal, 2015.
- [5] Kaspersky Lab. (2020). What’s a brute force attack?, URL: <https://www.kaspersky.com/resource-center/definitions/brute-force-attack>. Accessed: 01-05-2020.
- [6] Uku Tomikas in Messente. (2019). What are the benefits of two-factor authentication?, URL: <https://messente.com/blog/most-recent/benefits-of-two-factor-authentication>. Accessed: 01-05-2020.
- [7] IETF OAuth Working Group. (). OAuth 2.0, URL: <https://oauth.net/2/>. Accessed: 01-05-2020.
- [8] Brock Allen e Dominick Baier. (2020). Identity Server, URL: <https://identityserver4.readthedocs.io/en/latest/>. Accessed: 07-06-2020.
- [9] Gil Fink e Ido Flatow, *Pro Single Page Application Development*. Springer, 2014.

- [10] Neoteric in Medium. (). Single-page application vs. multiple-page application, URL: <https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>. Accessed: 01-05-2020.
- [11] Sam Allen. (2007). C# path.getrandomfilename method, URL: <http://www.dotnetperls.com/path-getrandomfilename>. Accessed: 05-06-2020.
- [12] Sam Corcos. (2020). HTML5 drag-drop zone with React.js: dropzone, URL: <https://github.com/react-dropzone/react-dropzone>. Accessed: 01-05-2020.
- [13] Emily Morehouse, Matt Zabriskie, Nick Uraltsev e Rubén Norte. (). HTTP client for the browser and node.js: axios GitHub repository, URL: <https://github.com/axios/axios>. Accessed: 25-05-2020.
- [14] Steve Smith e Rutger Storm in Microsoft Docs. (2020). Upload files in ASP.NET core, URL: <https://docs.microsoft.com/en-us/aspnet/core/mvc/models/file-uploads?view=aspnetcore-3.1>. Accessed: 08-04-2020.
- [15] Alex Reardon. (). Drag and drop with React: react-beautiful-dnd, URL: <https://github.com/atlassian/react-beautiful-dnd>. Accessed: 07-06-2020.