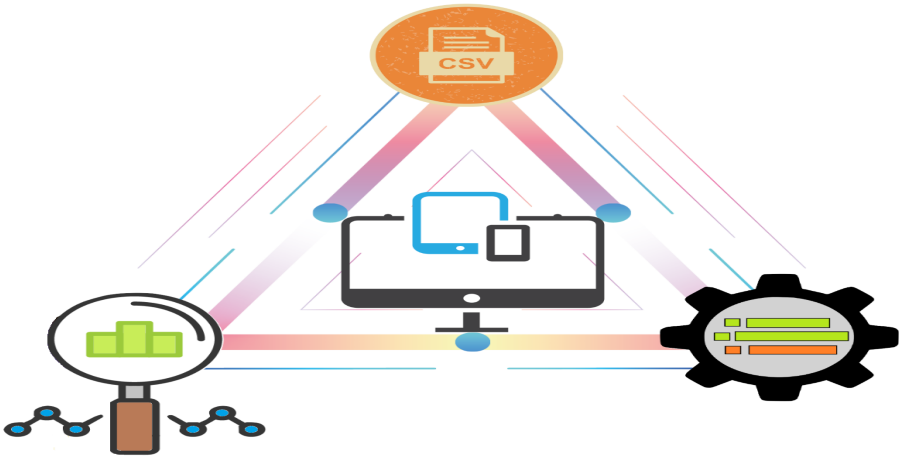
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

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

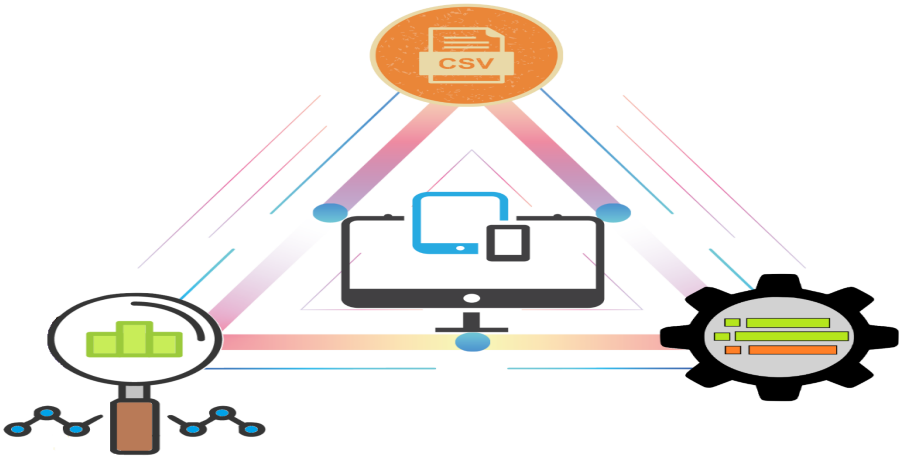
Orientadores : Doutor Nuno Datia

Doutora Matilde Pós-de-Mina Pato

Julho, 2020

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

Julho, 2020

A todos os que ajudaram ao longo do caminho...

Agradecimentos

Começamos por agradecer aos Professores Nuno Datia e Matilde Pato, por toda a orientação e ajuda que disponibilizaram ao longo de todo o desenvolvimento deste projeto, especialmente na ajuda e feedback sobre a realização deste relatório.

Agradeço aos meus amigos e família por toda a ajuda ao longo do caminho e por nunca duvidarem da minha paixão. Como não podia deixar de ser, ao Ruben Café por estar sempre disposto a trabalhar e ao Ivo Pereira pelo seu exemplar desempenho ao longo do trabalho. Eu sei que às vezes não foi fácil mas obrigado por terem aguentado até ao fim. - Duarte Felício

Agradeço aos meus familiares e amigos pelo apoio dado ao longo desta jornada. Por último mas não menos importante, aos meus colegas de grupo, pelo empenho demonstrado e trabalho realizado. - Ruben Café

Gostaria de expressar os meus agradecimentos aos meus caros colegas Duarte Felício e Ruben Café, pelo trabalho que ambos fizeram, pelo apoio demonstrado e pela capacidade metodológica na organização do trabalho em grupo. - Ivo Pereira

Resumo

O número de dados recolhidos e disponibilizados através de diferentes fornecedores aumentam diariamente.

Estes dados, tipicamente disponibilizados em formato tabular (maioritariamente em formato *csv*), devido ao seu tamanho e complexidade, tornam a sua correta análise em relação à identificação de níveis de detalhe e relações entre os dados numa tarefa árdua de realizar manualmente.

Nuno Ribeiro, na sua dissertação de mestrado, abordou este problema criando um processo semi-automático que, através de uma análise extensa aos dados, gera um ficheiro de meta-informação sobre os dados.

No entanto, o algoritmo desenvolvido pode conter alguns erros devido a diversos fatores.

Este projeto aborda o problema desenvolvendo uma aplicação web que, reutilizando o algoritmo de Ribeiro, fornece ao utilizador a possibilidade de analisar e alterar o resultado dessa análise de forma a poder corrigir eventuais erros no algoritmo.

Palavras-chave: Dados tabulares, Processo semi-automático, Meta-informação, Aplicação web

Abstract

The number of data collected and made available through different suppliers increases daily.

This data, typically made available in tabular format (mostly in textit csv format), due to it's size and complexity, makes the correct analysis in relation to the identification of levels of detail and relationships between data a difficult task to perform manually .

Nuno Ribeiro, in his master's thesis, addressed this problem by creating a semi-automatic process that, through extensive data analysis, generates a meta-information file about the data.

However, the developed algorithm may contain some errors due to several factors.

This project addresses the problem by developing a web application that, reusing Ribeiro's algorithm, provides the user with the possibility of analyzing and changing the result of that analysis in order to correct any errors in the algorithm.

Keywords: Tabular data, Semi-automatic process, Meta-information, Web application

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

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 [7]. 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*) [6]. 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.

Face a este problema Ribeiro [20] propôs um algoritmo 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 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 detete essa coluna como sendo um nível de detalhe de outra.

A partir do trabalho desenvolvido pelo Ribeiro em 2019, pretendemos contribuir com uma *user-interface* interativa que, após autenticação, permite carregar ficheiros *csv*, pedir ao algoritmo para os analisar dando hipótese ao utilizador de 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 - Região - Concelho*. É também relevante observar que o mesmo valor pode aparecer associado a diferentes níveis de detalhe, neste caso, o valor **Lisboa** tanto representa uma *Região* como um *Concelho*. 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. No capítulo 5 é descrito como foi feito o *deploy* da aplicação numa máquina virtual. 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 descrita a abordagem ao problema 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.

Smart Geo-Catalog Na dissertação Smart Geo-Catalog [2], 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 possibilita 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.

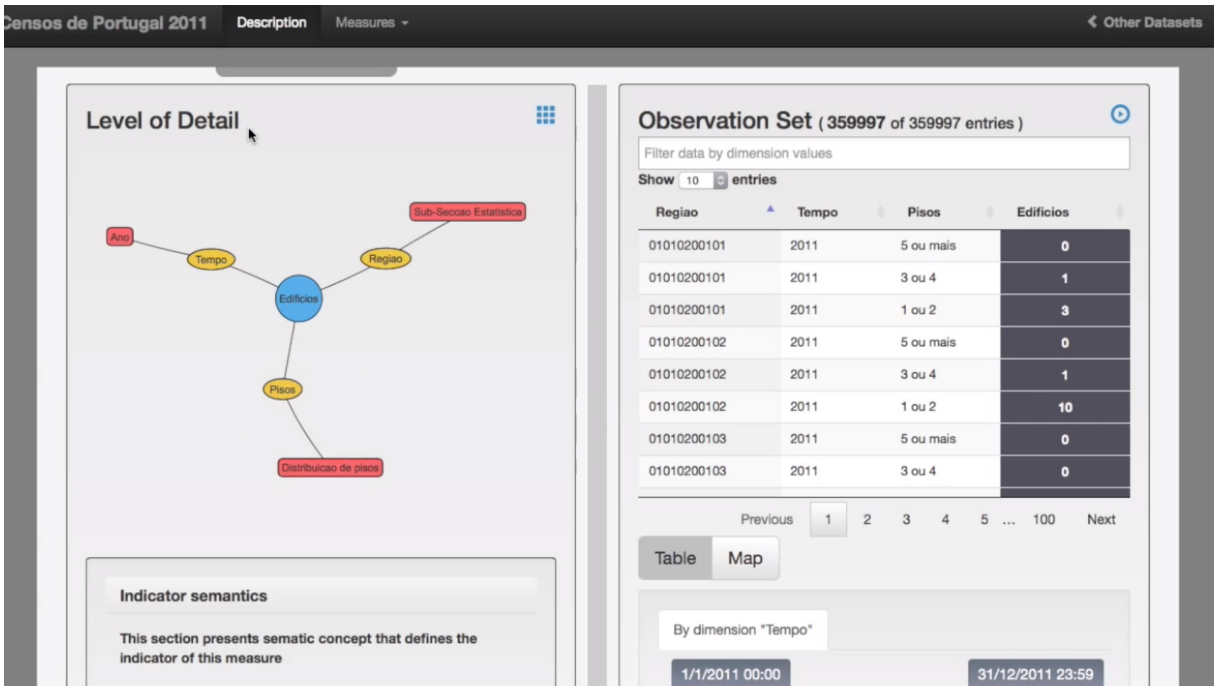


Figura 2.1: Interface Visual - Smart Geo-Catalog [2]

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[1] é uma ferramenta de software para análise e visualização de dados. Esta ferramenta consegue identificar que colunas são dimensões e métricas através de uma análise aos dados de cada coluna. Esta 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.

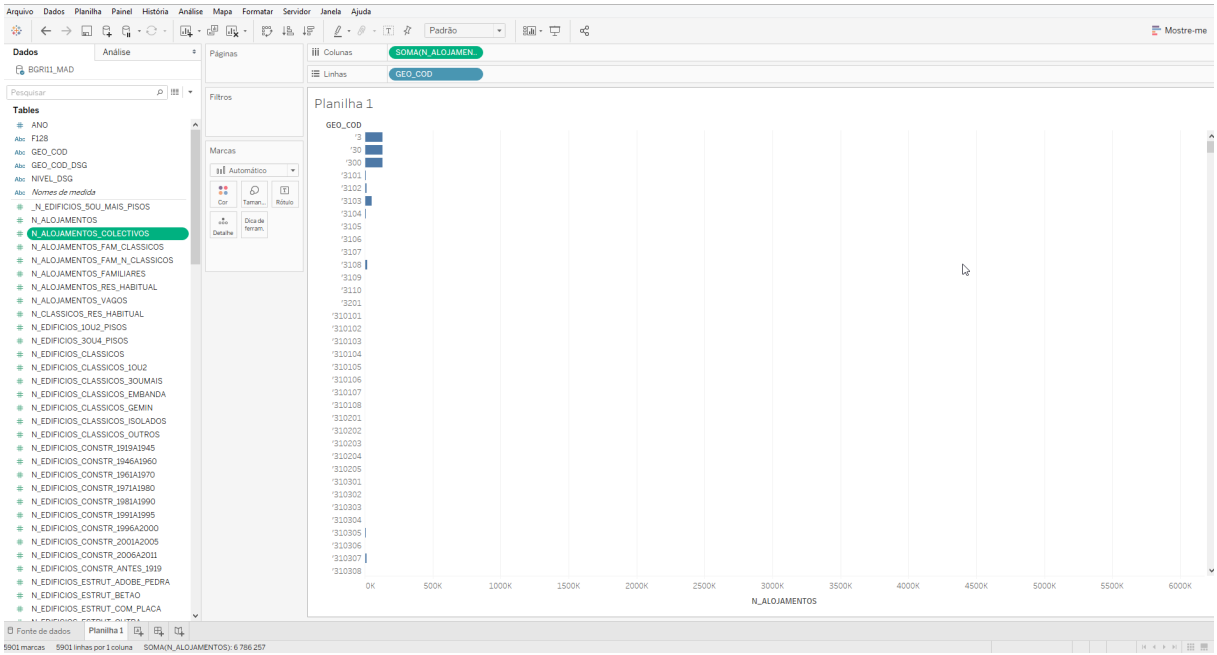


Figura 2.2: Tableau Desktop - BGRI11_MAD

Porém este software tem como objetivo auxiliar na análise de dados e não a geração de metadados. O software não suporta a funcionalidade de análise automática para identificação de domínios e níveis de detalhe que pretendemos com o nosso projeto. Para além disso, é um produto para fins lucrativos e não é *open source*, sendo ambos requisitos que temos o objetivo de cumprir.

[1] https://www.tableau.com/products/desktop

Alteryx Designer Alteryx Designer[2] 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-data para o ficheiro, no entanto, é uma meta-data bastante incompleta pois apenas fornece informação sobre o tipo de dados presente em cada coluna. A Figura 2.3 apresenta a meta-data gerada para o ficheiro de censos da Madeira.

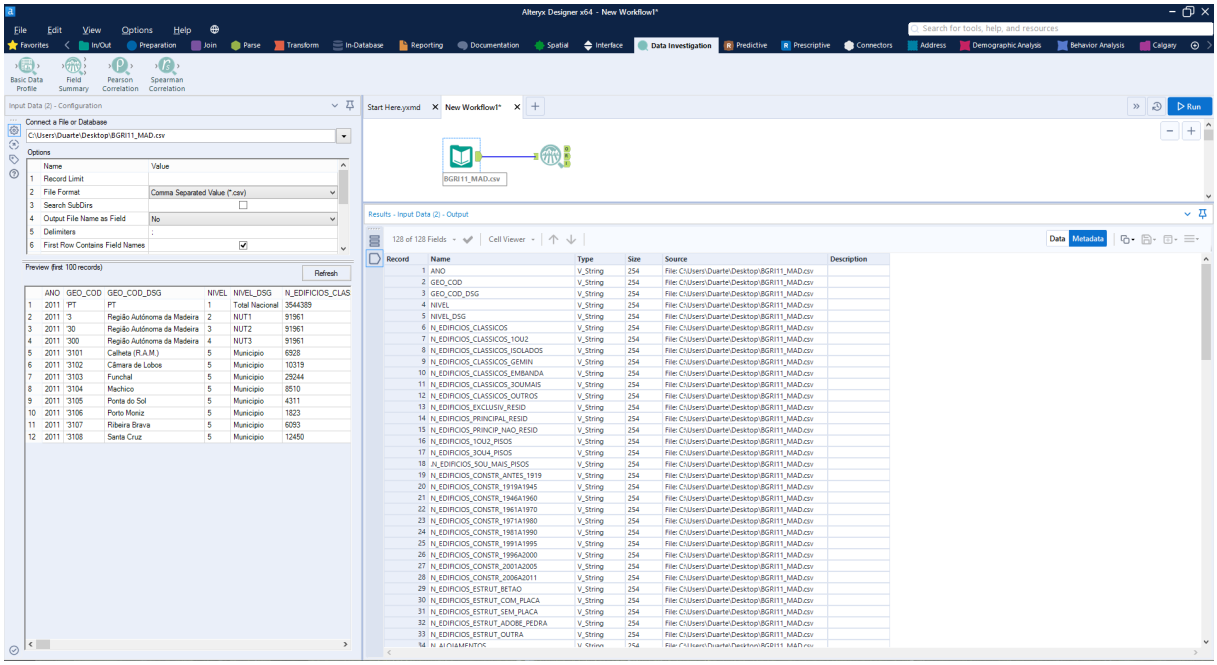


Figura 2.3: Alteryx Designer - BGRI11_MAD

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.

[2] https://www.alteryx.com/products/alteryx-platform/alteryx-designer

3

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* que está desacoplada do servidor e que comunica com este através do protocolo *Advanced Message Queuing Protocol* (**AMQP**) [3]. Estes três pontos foram divididos entre os membros do grupo de modo a ser possível organizar o progresso em *use cases* de todas as componentes. A Figura 3.1 apresenta a arquitetura da solução proposta para o projeto onde as setas representam o fluxo de dados.

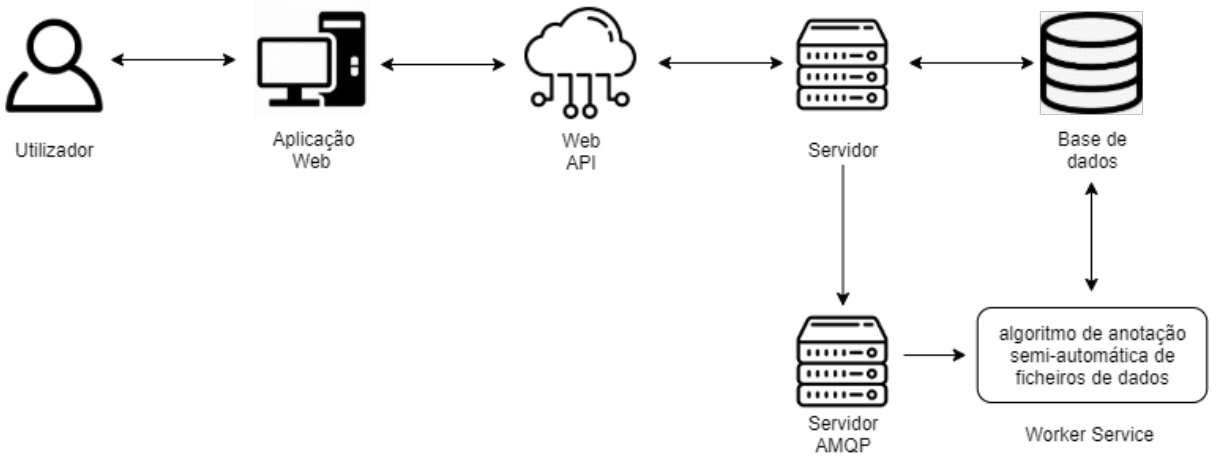


Figura 3.1: Arquitetura do Projeto

3.1 Algoritmo de análise

Existem três principais aspectos relativos aos ficheiros que devem ser analisados automaticamente, nomeadamente, na 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 [20], 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 um aspecto importante para potenciar a correcta comparação de dados provenientes de diferentes conjuntos e/ou fornecedores de dados [20].

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 [20]:

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 representado 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 induzem 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

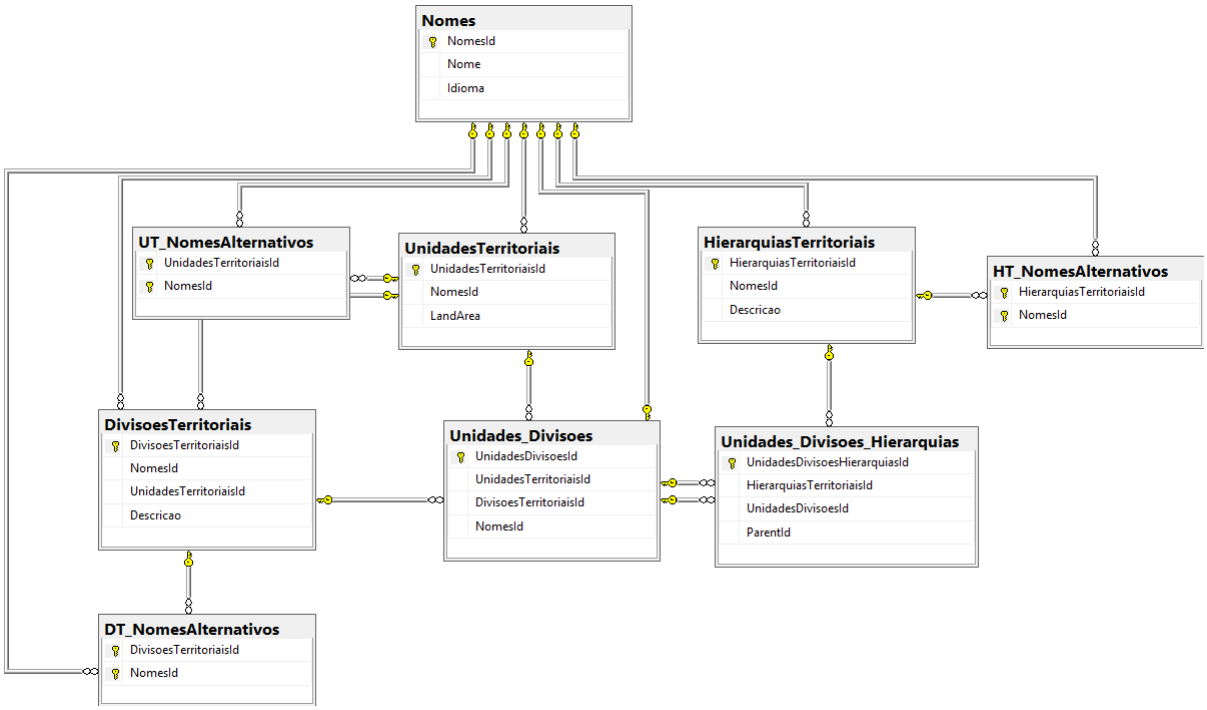


Figura 3.2: Modelo lógico da Base de Dados que suporta a KB (Adaptado de [20])

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 [20] implementou uma versão do algoritmo *Radix Tree*. Uma *radix tree* consiste numa árvore de prefixos com um passo extra que consiste em agrupar todos os nós que formam uma única palavra. 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).

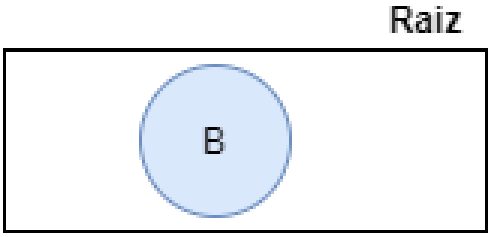


Figura 3.3: Construção da árvore - 1º carácter

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

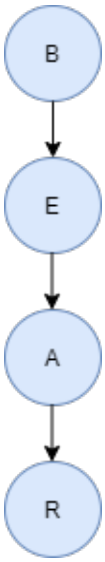


Figura 3.4: Construção da árvore - 1º palavra

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.

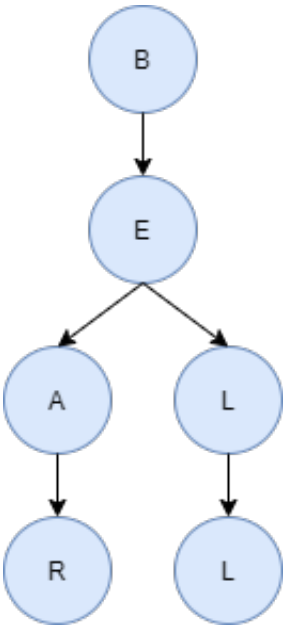


Figura 3.5: Construção da árvore - 2ª palavra

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.

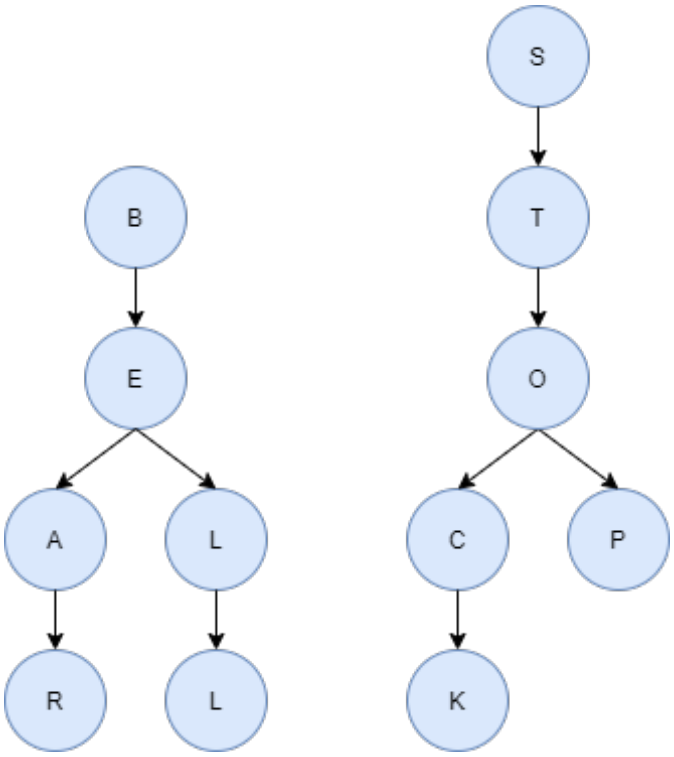


Figura 3.6: Construção da árvore - Árvore final

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 Ribeiro optou 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.

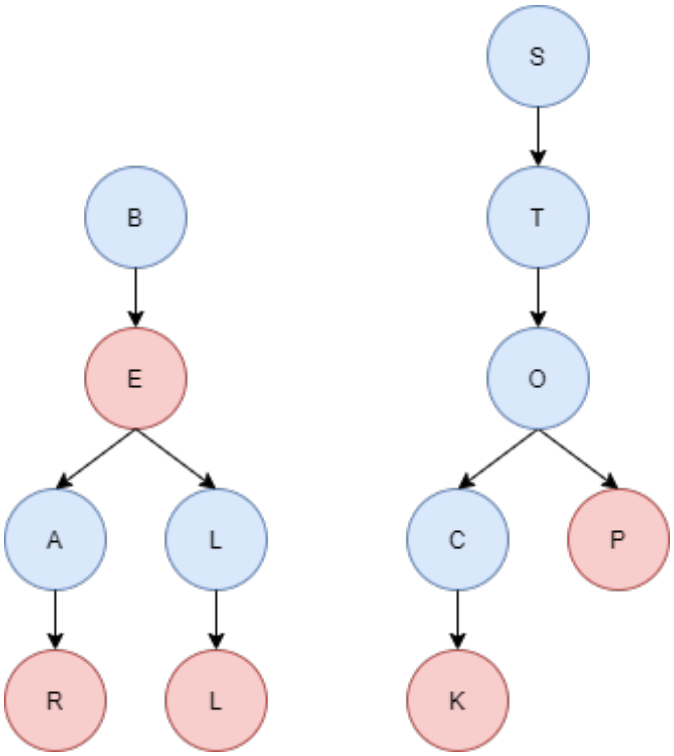


Figura 3.7: Construção da árvore - Árvore final com flag

Após todas as palavras estarem presentes na árvore, o passo final consiste em agrupar certos nós considerados desnecessários sendo esses os nós que apenas contêm 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.

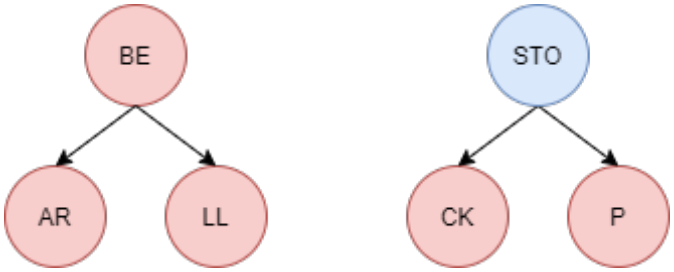


Figura 3.8: Construção da árvore - Árvore final gerada pelo algoritmo

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. Este ficheiro contém diversos dados sendo os mais relevantes para esta aplicação as Dimensões (linha 11-18), as Métricas (linha 26-29) e as Categorias (21-24).

```
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,"
       NumValoresNulos": 0,
13       "TodosDiferentes": false,
14       "TipoValores": [{"Tipo": "numeric","Count": 5901}],
15       "ValoresUnicos": ["2011"],
16       "TipoDominioGeo": null
17     },
18     ...
19   ],
20   "Metricas": {
21     "Categorias": [
22     {"CategoriaId": 1,"Nome": "N_","CategoriaPaiId": null},
23       {"CategoriaId": 2,"Nome": "N_EDIFICIOS_","CategoriaPaiId": 1},
24       ...
25     ],
26     "Colunas": [
27       {"IndiceColuna": 17,"NomeColuna": ".N_EDIFICIOS_5OU_MAIS_PISOS","
       CategoriaId": null,"E_Total": false},
28       {"IndiceColuna": 38,"NomeColuna": "N_CLASSICOS_RES_HABITUAL","
       CategoriaId": 1,"E_Total": false},
29       ...
30 } } ]
```

Listagem 3.1: Excerto de metadados gerados - BGRI11_MAD.json

3.2 Casos de utilização suportados

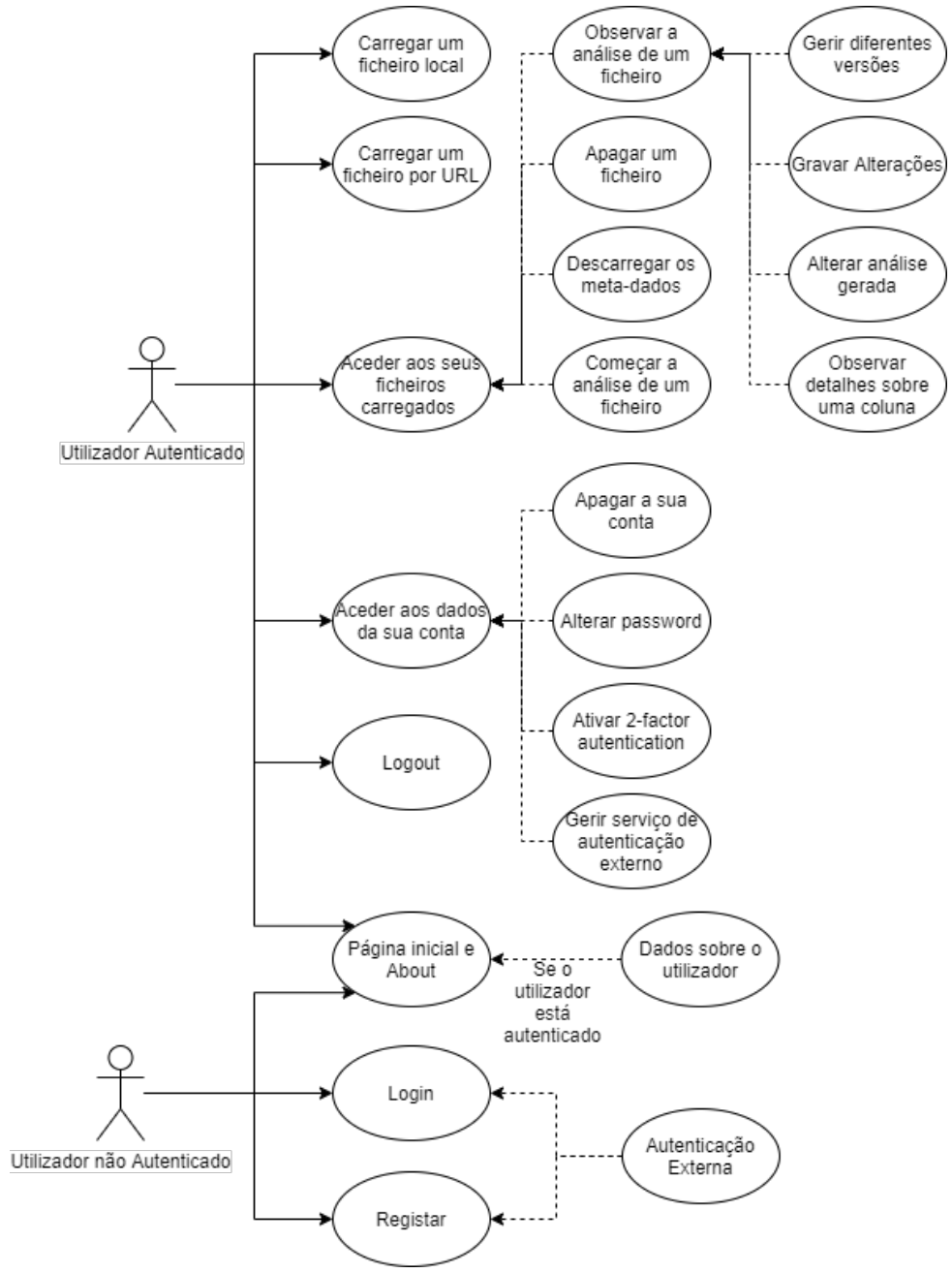


Figura 3.9: Casos de Utilização da aplicação

A aplicação suporta dois atores, um utilizador autenticado e um utilizador não autenticado, tendo ambos acesso a diversas ações sendo que grande parte das funcionalidades requerem que o utilizador esteja autenticado.

Registar 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 sendo que dois utilizadores não podem utilizar o mesmo 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 válidas 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.

Autenticação Externa Um utilizador também se pode autenticar ou registar através de um serviço externo, nomeadamente o *Google* ou o *Facebook* através da tecnologia *OAuth 2.0* [14]. O utilizador é redirecionado para a página do respetivo serviço onde tem que se autenticar com uma conta válida desse serviço. Após se autenticar com sucesso o utilizador é redirecionado novamente para a nossa aplicação, autenticado se já tinha conta ou então para uma página onde apenas tem que fornecer um nome de usuário completando o registo.

Aceder aos dados da sua conta Um utilizador autenticado ao clicar no seu email na barra de acesso é redirecionado para uma página onde tem acesso a funcionalidades relacionadas com a sua conta descritas nos quatro casos abaixo.

Ativar 2-factor autentication Um utilizador autenticado tem a possibilidade de ativar *Two-factor-authentication* de forma a aumentar a segurança e integridade da sua conta [22]. 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.

Gerir serviço de autenticação externo Um utilizador pode adicionar á sua conta um serviço de autenticação externo de forma a facilitar autenticações futuras. Este caso

funciona similarmente ao descrito acima em **Autenticação Externa**.

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 tem a possibilidade de apagar todos os dados e ficheiros associados à sua conta. Este processo é irreversível e é pedido novamente a password ao utilizador para ter a certeza que não foi pedido por engano.

Logout Um utilizador autenticado pode terminar a sua sessão fazendo logout.

Página Inicial - About Na página inicial da aplicação, ao utilizador não autenticado é apresentado uma série de *gifs* que ilustram o espectado uso da nossa aplicação. Se o utilizador está autenticado o caso de utilização abaixo (Dados sobre o utilizador) é o apresentado.

Na página **About** a nossa aplicação e as suas funcionalidades são apresentadas em maior detalhe.

Dados sobre o utilizador Se o utilizador estiver autenticado a página inicial contém um histórico sobre as últimas ações, relacionadas sobre a análise de ficheiros, realizadas pelo utilizador de forma a facilitar um utilizador que não aceda à aplicação à vários dias se relembre das suas ações passadas contendo também um *link* para ir diretamente à análise desses ficheiros. Contém também alguns dados relevantes sobre o utilizador como quantos ficheiros tem carregados, quantos deles estão analisados, quando foi o último *login* entre outros.

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 carregar um ficheiro através de um *URL*. 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. Ao clicar sobre um deles informação sobre o ficheiro é mostrada ao utilizador, 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 quatro outros casos que vão ser explicados nos próximos parágrafos.

Começar a análise de um ficheiro Ao clicar 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 Ao clicar sobre um ficheiro onde a análise já foi feita é possível fazer *download* dessa análise recebendo o respetivo ficheiro *json*.

Apagar um ficheiro Ao clicar sobre um ficheiro é possível apagar esse ficheiro. Um alerta dá *pop-up* pedindo confirmação ao utilizador. Em caso de confirmação o ficheiro e todas as suas análises são apagadas.

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 quatro outros casos de utilização descritos nos próximos parágrafos.

Observar detalhes sobre uma coluna Ao clicar numa das colunas é possível observar alguns dados que a dizem respeito como qual o índice dessa coluna no ficheiro original, a que categoria pertence no caso de ser uma métrica ou se os valores são todos diferentes no caso de uma dimensão. É também possível observar os dez primeiros valores de cada coluna de modo a dar uma melhor compreensão ao utilizador de que tipo de valores estão presentes em cada coluna.

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.

Gravar Alterações Este caso fornece a possibilidade de gravar as alterações feitas pelo utilizador sobre a análise ficando essa a versão mais recente. Um *pop-up* avisa sobre se a operação foi ou não bem sucedida.

Gerir diferentes versões Este caso dá a possibilidade ao utilizador de observar todas as diferentes versões de análise feitas sobre o ficheiro. É possível carregar uma versão anterior fazendo com que a página se altere para essa versão. Outra possibilidade é apagar uma versão, no entanto, não é possível apagar a primeira versão pois foi a análise realizada pelo 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 funcionalidades 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 **Dados sobre o utilizador**;
2. *About*, página com informação detalhada sobre a aplicação que contém o caso **About**;

3. *Register*, para registar um novo utilizador que contém o caso **Registar** e **Autenticação Externa**;

4. *Login*, para autenticar um utilizador registado que contém o caso **Login** e **Autenticação Externa**.

5. 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;

6. *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**

7. *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;

Os detalhes sobre como foram implementadas estas páginas estão descritos no próximo capitulo.

4

Implementação

4.1 Tecnologias *open-source*

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 grande parte das linguagens tem as capacidades necessárias para implementar as metodologias. No entanto, foi nosso objetivo (sem prejuízo para a eficiência e desempenho) 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 [20], 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[1].

Para a implementação da base de dados foi usado *SQL Server*[2]. 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*[3].

[1] https://dotnet.microsoft.com/
[2] https://www.microsoft.com/en-us/sql-server
[3] https://reactjs.org/

4.2 Estrutura da componente servidora

A parte cliente da aplicação comunica com o servidor através de *endpoints*. Todos os *endpoints* do servidor estão presentes em classes controladoras de rotas (*controllers*). Cada *controller* contém os *endpoints* de uma determinada vista da aplicação. A Tabela 4.1 mostra a listagem dos *endpoints* da API da aplicação.

Tabela 4.1: *Endpoints* disponibilizados pela API

MÉTODO	ROTA	DESCRIÇÃO
	FILEIPLOADCONTROLLER	
POST	`FileUploadController/Physical`	Carregar um ficheiro da máquina do utilizador para o servidor
POST	`FileUploadController/Remote`	Carregar um ficheiro de um endereço da Web para o servidor
	WORKSPACECONTROLLER	
GET	`WorkspaceController/GetUserFiles`	Listar ficheiros de um utilizador
DELETE	`WorkspaceController/RemoveFile`	Remover um ficheiro de um utilizador
GET	`WorkspaceController/AnalyseFile`	Analisar um ficheiro
GET	`WorkspaceController/ReturnAnalysis`	Retornar a análise de um ficheiro
GET	`WorkspaceController/ReturnAnalysisVersion`	Retornar uma versão específica de análise de um ficheiro
GET	`WorkspaceController/IsAnalysisComplete`	Verificar se a análise de um ficheiro está completa
GET	`WorkspaceController/ListAnalysis`	Listar todas as versões de análise de um ficheiro
POST	`WorkspaceController/SaveAnalysis`	Guardar uma versão de análise
DELETE	`WorkspaceController/DeleteAnalysisVersion`	Eliminar uma versão de uma análise
GET	`WorkspaceController/GetFirstValues`	Retornar os primeiros valores de uma coluna de um ficheiro
GET	`WorkspaceController/MetricToDimension`	Alterar o classificador de uma coluna de métrica para dimensão
	HOMECONTROLLER	
GET	`HomeController/GetUserDetails`	Retornar alguns detalhes sobre o trabalho do utilizador na aplicação

File System O *file system* é utilizado de forma a guardar os ficheiros carregados pelos utilizadores e as versões de análise, a Figura 4.1 apresenta a estrutura adotada. Neste caso, para um utilizador que tem dois ficheiros carregados, *ficheiro_1* e *ficheiro_2*, onde o *ficheiro_2* já foi analisado pelo algoritmo e neste caso já tem duas versões pois o utilizador alterou a análise e gravou a nova versão. O *ficheiro_1* ainda não foi analisado.

4.3 Autenticação de utilizadores

Na implementação de autenticação, de maneira segura, recorremos à *framework IdentityServer* [11]. Este *middleware* é adicionado na configuração da aplicação e fica responsável por todas as ações relativas à autenticação. Adiciona automaticamente todos os *endpoints* necessários para o registo, autenticação e gestão de conta para um utilizador. Para guardar localmente os utilizadores registados, o *IdentityServer* necessita de uma base de dados local, tipicamente *SQL Server*. Na Figura 4.2 podemos observar o modelo de dados oferecido pelo *IdentityServer*.

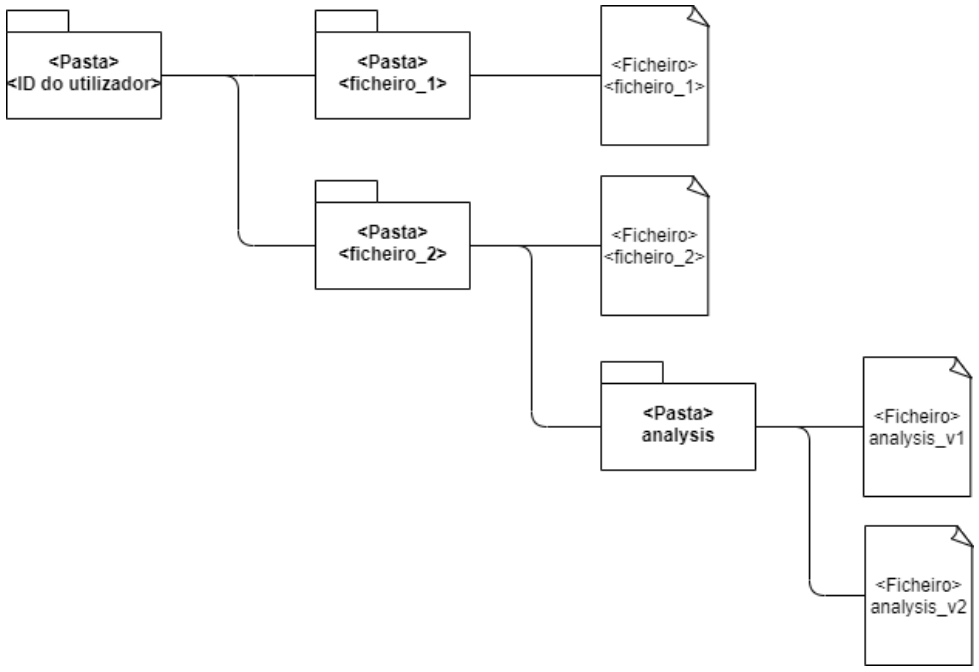


Figura 4.1: Estrutura do *file system* no servidor

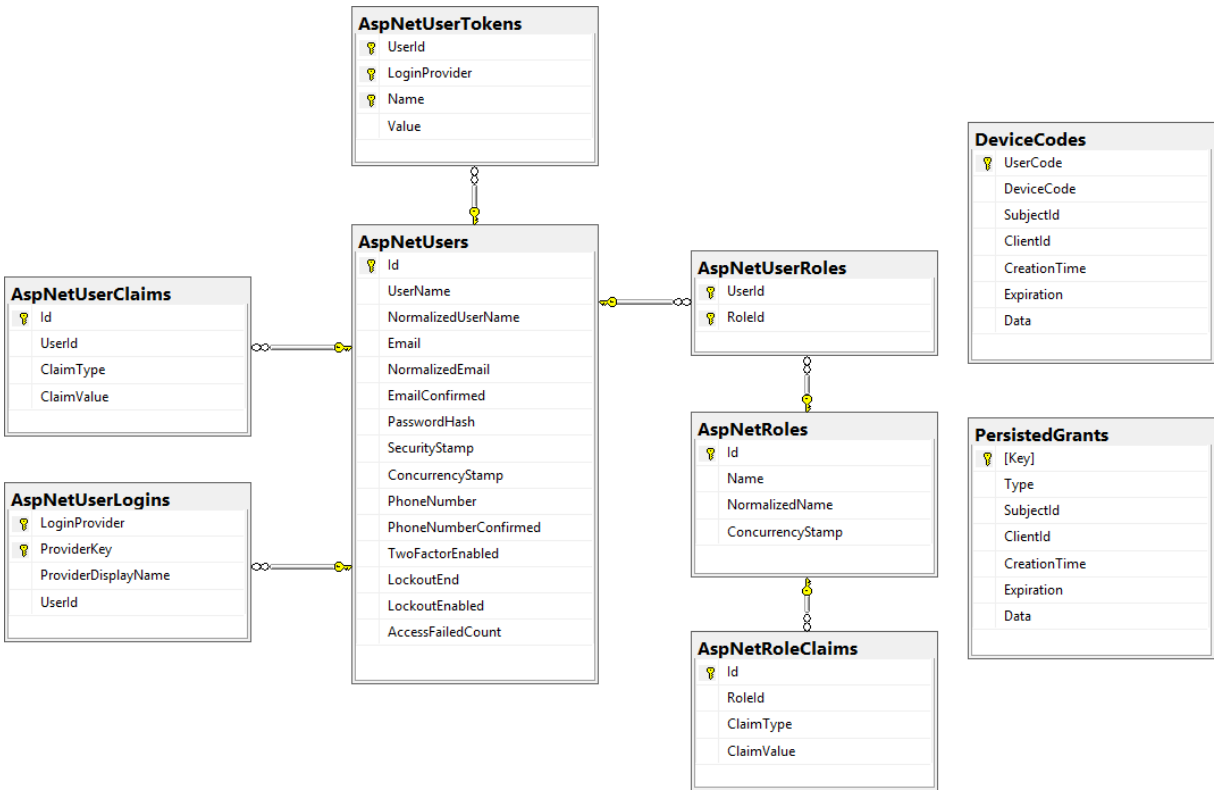


Figura 4.2: Modelo da base de dados para a autenticação, *framework IdentityServer*

A tabela **AspNetUsers** é uma das mais importantes, pois é onde são armazenados os dados do utilizador como o seu *email*, *password* (*hashed*) entre outros. A geração do

hash inclui *salt* por forma a duas *passwords* idênticas não gerarem o mesmo *hash*. 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 o tempo necessário para cada tentativa de *password* aumente linearmente.

4.4 Página de entrada do site

A aplicação foi desenvolvida como *Single Page Application* (SPA). Uma SPA contém uma única página Web, usa *JavaScript*, *HTML5* e *CSS* para a interacção com o utilizador. Na SPA, após o carregamento inicial, não existe mais nenhum carregamento total durante o seu uso [8]. 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, como *AngularJS*[4], *Ember.js*[5], *Meteor.js*[6], *Knockout.js*[7] e *ReactJS*. Aplicações Web tais como *Gmail*, *Google Maps*, *Facebook* ou *GitHub* são SPA [21]. Tendo isto em mente implementámos a nossa aplicação como uma SPA de modo a cumprir os requisitos referidos na secção 3.2.1.

Página Inicial - Autenticado Após a autenticação, na página inicial o utilizador tem informação sobre as suas últimas operações (Figura 4.3.Ⓐ), o número de ficheiros carregados (Figura 4.3.Ⓑ), a data do último acesso (Figura 4.3.Ⓒ). Na página encontram-se dois gráficos de pizza (Figura 4.3.Ⓓ), que mostram a estatística dos ficheiros analisados (a azul) e carregados localmente (a azul).

Sempre que um utilizador faz *login* é adicionada uma nova entrada à tabela **LoginRecord** guardando, também, a data em que foi feito. Quando uma análise sobre um ficheiro termina, e.g. carregada ou gravada, é adicionada uma nova entrada à tabela **ActionRecord**, ficando registado o tipo de ação, a versão da meta-informação que estava a ser analisada e a data. Com esta informação é possível gerar uma hiperligação (*href*) que leve o utilizador diretamente à versão onde se encontrava previamente. Este *href* é disponibilizado a partir da página inicial.

[4] https://angularjs.org/
[5] https://emberjs.com/
[6] https://www.meteor.com/
[7] https://knockoutjs.com/

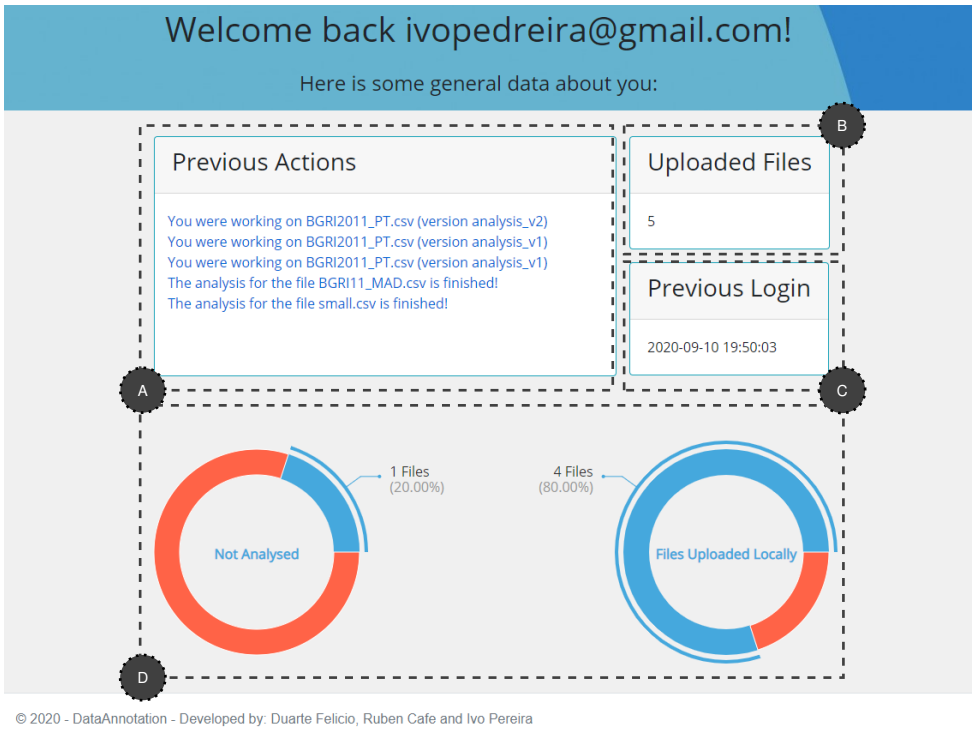


Figura 4.3: Página inicial para um utilizador autenticado. As áreas destacadas representam os principais componentes de interacção com o utilizador, nomeadamente: (A) as últimas operações efectuadas, (B) o número de ficheiros carregados, (C) a data de acesso e (D) uma representação gráfica dos ficheiros analisados e carregados

4.5 Carregamento de ficheiros

A aplicação permite efectuar o carregamento 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 e uma vez carregado na aplicação, é guardado no *file system* do servidor após serem efectuadas algumas verificações, nomeadamente se o ficheiro é realmente *csv* e se não está vazio. Os ficheiros são guardados com um nome gerado aleatoriamente pela aplicação de modo a melhorar a segurança [17]. 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 4.4 podemos observar o modelo de dados que suporta um ficheiro *csv*.

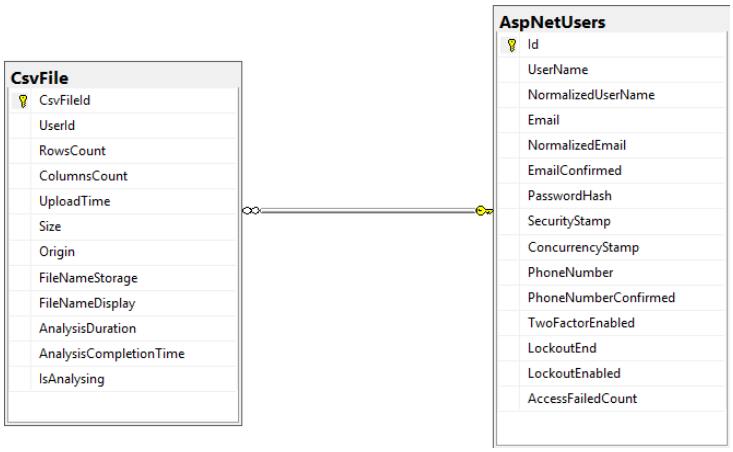


Figura 4.4: Modelo da Base de Dados para o carregamento de ficheiros - `CsvFile`

Lado Cliente A parte cliente para carregamento de ficheiros, como se observa na Figura 4.5, foi implementada através da biblioteca *React Dropzone* que facilita a criação e customização de uma zona de *drag'n'drop* [19] (Figura 4.5.Ⓐ), de modo a só serem aceites ficheiros *csv*.

Consideramos relevante que o progresso de carregamento de ficheiros fique visível ao utilizador, por essa razão implementámos essa funcionalidade (Figura 4.5.Ⓑ). A mesma foi implementada recorrendo à biblioteca *axios* [4]. Esta, permite fazer pedidos HTTP com a peculiaridade de ser possível saber a quantidade de dados já recebidos em comparação ao enviado.

A vista também fornece informação de ficheiros aceites e rejeitados (Figura 4.5.Ⓒ) dentro dos daqueles que foram carregados. Por exemplo, se se arrastar um ficheiro *csv* vazio ou maior do que 1GB (valor escolhido arbitrariamente) para a zona de *drag'n'drop* (Figura 4.5.Ⓐ), este é carregado, no entanto será rejeitado aparecendo na zona dedicada aos ficheiros rejeitados.

Lado Servidor Numa das opções para *upload* de ficheiro, o servidor recebe um ficheiro proveniente do cliente em formato *multipart* [12]. É aberto um *stream* com o conteúdo que vem no pedido, *stream* esse que é guardado no servidor como um 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 [23].

Para o carregamento a partir de um *url* o servidor recebe no pedido o *url* do ficheiro que tem de descarregar. 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.

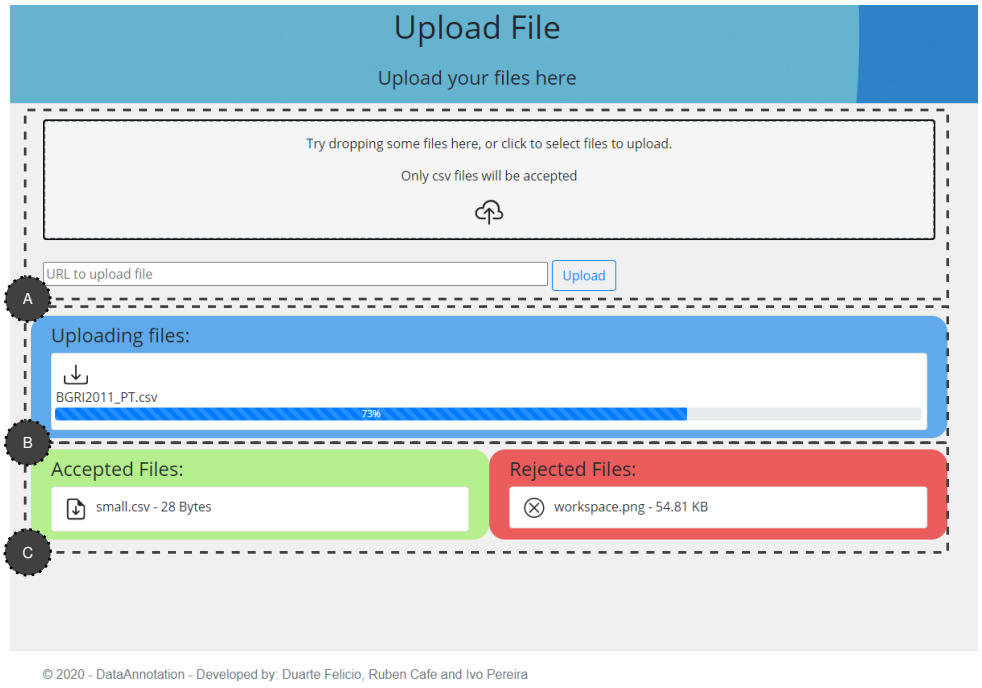


Figura 4.5: Página de utilizador autenticado para carregar ficheiros *csv*. As áreas destacadas representam nomeadamente: (A) área para carregamento de ficheiros *csv*, (B) barra de progresso durante o carregamento, (C) registo de ficheiros aceites e rejeitados

4.6 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.

Na Figura 4.6 podemos observar a página do ambiente de trabalho de um utilizador, a imagem da esquerda apresenta um ficheiro com a análise a decorrer e à direita um ficheiro com a análise completa. Com a análise completa podemos observar novos campos e também a possibilidade de alterar a análise com o botão **Go to Analysis**.

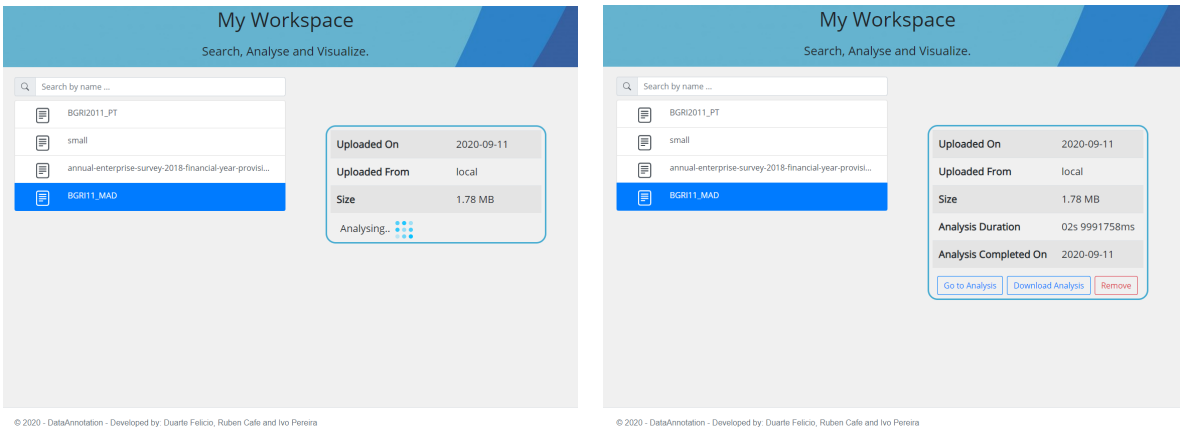


Figura 4.6: Ambiente de Trabalho durante a análise de ficheiros carregados, a imagem da esquerda representa a análise em progresso e à direita quando foi concluída

4.6.1 Análise de ficheiros

A aplicação tem como foco principal a análise semi-automática de ficheiros *csv*, onde a colaboração do utilizador para corrigir ou alterar a análise é essencial. De seguida, será descrita a implementação dos diferentes componentes que tornaram esta opção possível.

Lado Cliente Sempre que um utilizador opta por pedir uma análise, o *browser* irá reencaminhar esse pedido ao servidor. Sempre 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, ver Listagem 4.1 (linha 85–97). 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. Se a análise ainda não terminou o servidor devolve 204, se já terminou o intervalo termina alterando então a vista com os novos dados, ver Listagem 4.1 (linha 99). 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 ambiente de trabalho, se a análise ainda não terminou, o intervalo é criado novamente Listagem 4.1 (linha 98).

Lado Servidor A análise de um ficheiro é realizada através de um algoritmo, aproveitando o trabalho de Ribeiro [20] como mencionado na secção 3.1. Este algoritmo analisa informações como os nomes das colunas do ficheiro *csv* e os próprios dados, dependendo da dimensão do ficheiro, o tempo de processamento poderá estar fora do tempo que os utilizadores acham admissível para um tempo de espera. Para não

degradar o 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.

```
83      async checkAnalysisStatus(id, token) {
84          var map = this.state.files
85          var requestLoop = setInterval(function () {
86              fetch(`Workspace/IsAnalysisComplete?fileId=${id}`, {
87                  method: 'GET',
88                  headers: !token ? {} : {'Authorization':`Bearer ${token}`}
89              }).then(res => {
90                  if (res.status !== 204) {//204=analysis uncomplete
91                      res.json().then(newFile => {
92                          map.set(id, newFile)
93                          stopLoop(map)
94                      })
95                  }
96              })
97          }, 5000); //5 seconds
98          this.setState({ requestLoops: this.state.requestLoops.concat(
    requestLoop) })
99          var stopLoop = (newMap) => {
100             this.setState({ files: newMap })
101             this.forceUpdate()
102             clearInterval(requestLoop)
103     }   }
```

Listagem 4.1: Método para verificar se a análise já terminou

A nossa aplicação irá comunicar com o serviço de análise através de uma fila de mensagens que segue o protocolo AMQP implementado através do *message broker RabbitMQ* [16]. O serviço de análise irá estar à escuta, como consumidor na fila, por instruções provenientes da aplicação, o produtor. Cada mensagem irá ter o *id* do ficheiro e a localização desse ficheiro. O *id* é enviado pois o serviço tem que poder aceder e alterar informação sobre esse ficheiro na base de dados, e a localização para que o serviço possa aceder a esse ficheiro para leitura de forma a analisá-lo. 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 porque desta forma têm de correr na mesma máquina. Dada a possibilidade e probabilidade dos ficheiros serem de grandes dimensões, chegando à ordem dos GB, não seria viável ter que enviar este ficheiro da aplicação para o serviço de análise.

4.6.2 Remoção de ficheiros

A remoção de um ficheiro pode em primeira vista parecer algo banal. Contudo, um ficheiro pode conter várias componentes, nomeadamente as diferentes versões da análise desse ficheiro, a sua entrada na base de dados e o ficheiro em si presente no *file system* do servidor. A remoção de um ficheiro pode tornar-se não linear devido a erros que possam acontecer em cada uma dessas remoções. Sendo que o principal objetivo mantêm-se, i.e. tornar a base de dados e o *file system* consistentes.

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.

Lado Servidor 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 reposta. Na Listagem 4.2 pode ser observado pseudo código do método de remoção de um ficheiro, quando feito com sucesso ou não. A implementação é semelhante à de uma saga com duas transações, sendo estas a remoção da entrada na tabela *CsvFile* e a remoção do ficheiro e todas as suas análises no *file system*.

4.6.3 Processo de descarregar metadados

Um dos requisitos funcionais consiste então em ser possível descarregar os metadados gerados (ver secção 3.1.4). A implementação por parte do servidor é bastante direta, este apenas devolve o ficheiro *.json* com a análise presente no *file system*, logo, apenas a parte cliente vai ser explicada com mais detalhe.

Lado Cliente Quando o utilizador pressiona o botão para descarregar o ficheiro, é feito o pedido ao servidor para obter a análise. De seguida, é criado um *blob* (*Binary Large Object*) que contém o ficheiro *json* gerado pela análise, finalizando com a criação de um novo elemento *HTML*, *a*, que contém os campos *href* que aponta para o *blob* criado e o campo *download* para descarregar o ficheiro. A Listagem 4.3 apresenta o código responsável por descarregar o ficheiro da análise.

```
[HttpDelete]
public IActionResult RemoveFile([FromQuery]int fileId)
{
  if(!checkIfFileExists(fileId))return Error(500)
  string filePath = getFilePath(fileId);
  try{
    removeFileFromDataBase(fileId);
    removeFileFromFileSystem(filePath);
  }
  catch (DataBaseException e){
    return Error(500);
  }
  catch (FileSystemException e){
    re-addFileToDataBase(fileId);
    return Error(500);
  }
  return Ok();
}
```

Listagem 4.2: Método que trata de remoção de um ficheiro - Pseudo código

```
async DownloadAnalyzis(id,fileName) {
  const token = await authService.getAccessToken();
  fetch(`Workspace/DownloadAnalysis?fileId=${id}`, {
    method: 'GET',
    headers: !token ? {} : { 'Authorization': `Bearer ${token}` }
  }).then(response => {
    response.blob().then(blob => {
      let url = window.URL.createObjectURL(blob);
      let a = document.createElement('a');
      a.href = url;
      a.download = fileName.split('.')[0]+'_analysis'+'.json';
      a.click();
}); }); }
```

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

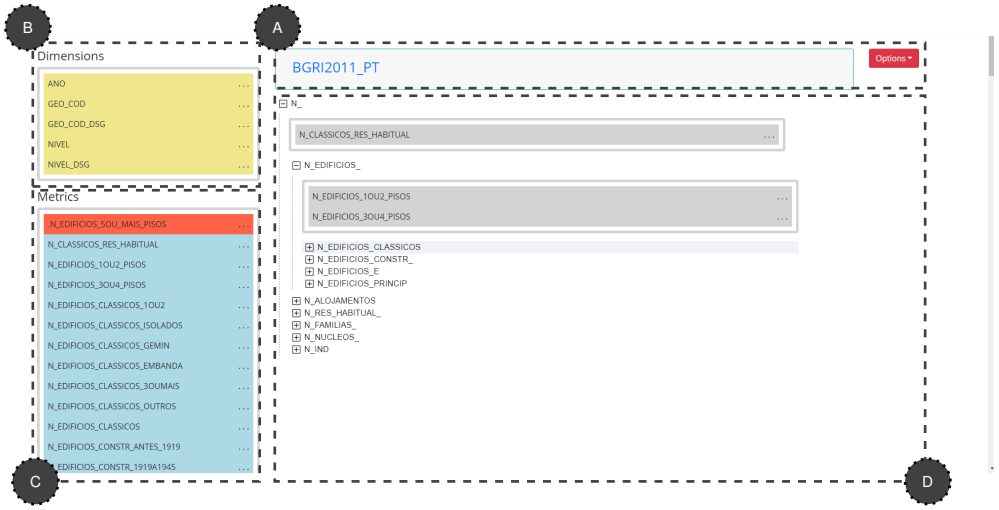
4.6.4 Visualizar e alterar a análise de um ficheiro *csv*

A vista da análise de um ficheiro é constituída por quatro zonas:

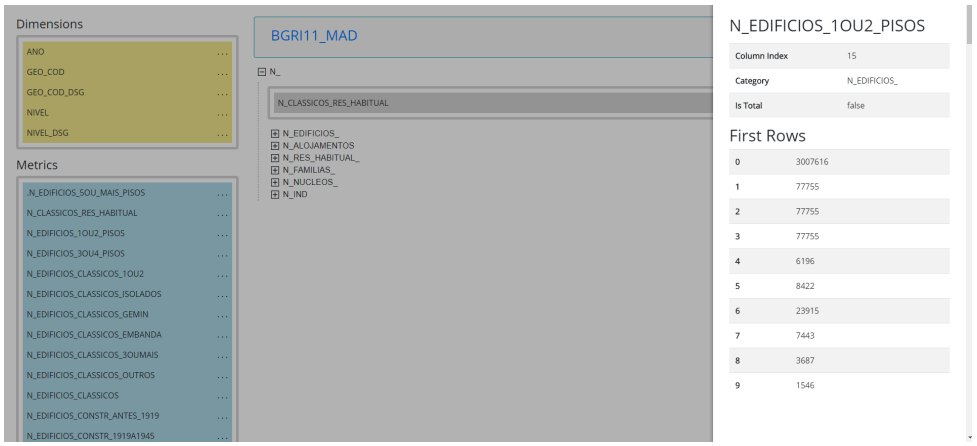
1. A que contém o nome do ficheiro, com a possibilidade de gravar a análise, gerir versões de análises previamente guardadas ou de descarregar a meta-informação (Figura 4.7a.Ⓐ);
2. Com as dimensões (Figura 4.7a.Ⓑ);
3. Com as métricas (Figura 4.7a.Ⓒ);
4. Com todas as categorias e suas colunas (Figura 4.7a.Ⓓ).

Outra funcionalidade relevante na página é a apresentação de detalhes sobre uma coluna específica ao clicar sobre os três pontos presentes em cada coluna. A informação sobre a coluna é apresentada numa janela que contém dados obtidos pela análise e a possibilidade de observar as primeiras linhas de valores sobre a coluna.

Na Figura 4.7a podemos observar a página de análise de um ficheiro, neste caso do ficheiro de censos da Madeira e na Figura 4.7b podemos observar a janela de dados sobre um coluna, e.g. "N_EDIFICIOS_1OU2_PISOS".



(a) Página da análise



(b) Janela de dados

Figura 4.7: Vista da página quando decorre a análise de um ficheiro

Lado Cliente 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 ficheiro apresentado na Figura 4.7. 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_. Com essa informação optámos por transformar esses dados numa árvore n-ária para ser mais fácil desenvolver a vista necessária.

Cada métrica identificada pela análise pertence, ou não, a uma categoria. 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 4.8 está ilustrado um exemplo do esquema da árvore de categorias resultante.

```
...
"Metricas": {
    "Categorias": [
      {
        "CategoriaId": 1,
        "Nome": "N_",
        "CategoriaPaiId": null
      },
      {
        "CategoriaId": 2,
        "Nome": "N_EDIFICIOS_",
        "CategoriaPaiId": 1
      },
      {
        "CategoriaId": 3,
        "Nome": "N_EDIFICIOS_CLASSICOS",
        "CategoriaPaiId": 2
      },
      {
        "CategoriaId": 4,
        "Nome": "N_EDIFICIOS_CONSTR_",
        "CategoriaPaiId": 2
      },
      {
        "CategoriaId": 5,
        "Nome": "N_EDIFICIOS_CONSTR_19",
        "CategoriaPaiId": 4
      },
    ...
  ],
},
...
```

Listagem 4.4: Excerto da metadata - BGRI11_MAD.csv - Categorias

Para facilitar a interação com os utilizadores, implementou-se a funcionalidade de arrastar uma categoria ou dimensão para outra divisão válida na interface. Usámos então

a biblioteca *react-beautiful-dnd* [18], que fornece dois componentes principais, nomeadamente, *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 num *droppable*, o mesmo, ou não.

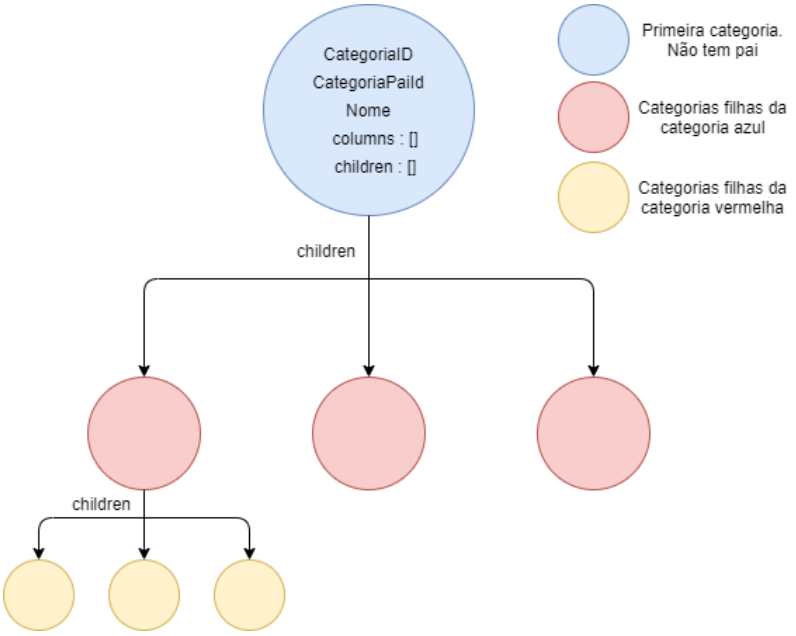


Figura 4.8: Árvore n-ária de categorias resultante da organização da meta-data

Assim, cada categoria consiste numa zona *droppable* que é inicializada com as suas colunas, sendo cada uma um componente *draggable*. A árvore criada anteriormente é então transformada nesses componentes através dum método recursivo que percorre cada categoria da árvore. Na Figura 4.9 pode ser observada 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 por sua vez, o nível 2 é um detalhe do nível 1.

Lado Servidor Existem diversas ações na análise que necessitam de fazer pedidos ao servidor. Uma das ações é gravar uma nova versão de meta-informação após alterações. Isto implica enviar a nova versão ao servidor onde este a guarda no *file system* utilizando uma convenção nos nomes das versões de forma a saber qual a mais recente. Para ser possível um utilizador aceder a todas as versões de uma certa análise é também necessário ser feito um pedido ao servidor, cuja resposta contém uma lista com todas as versões dessa análise juntamente com as datas da ultima alteração de cada versão. Nesta lista o utilizador pode ainda fazer duas acções em cada entrada. Uma que é apagar uma versão da análise, em que será feito um pedido de *delete* ao servidor

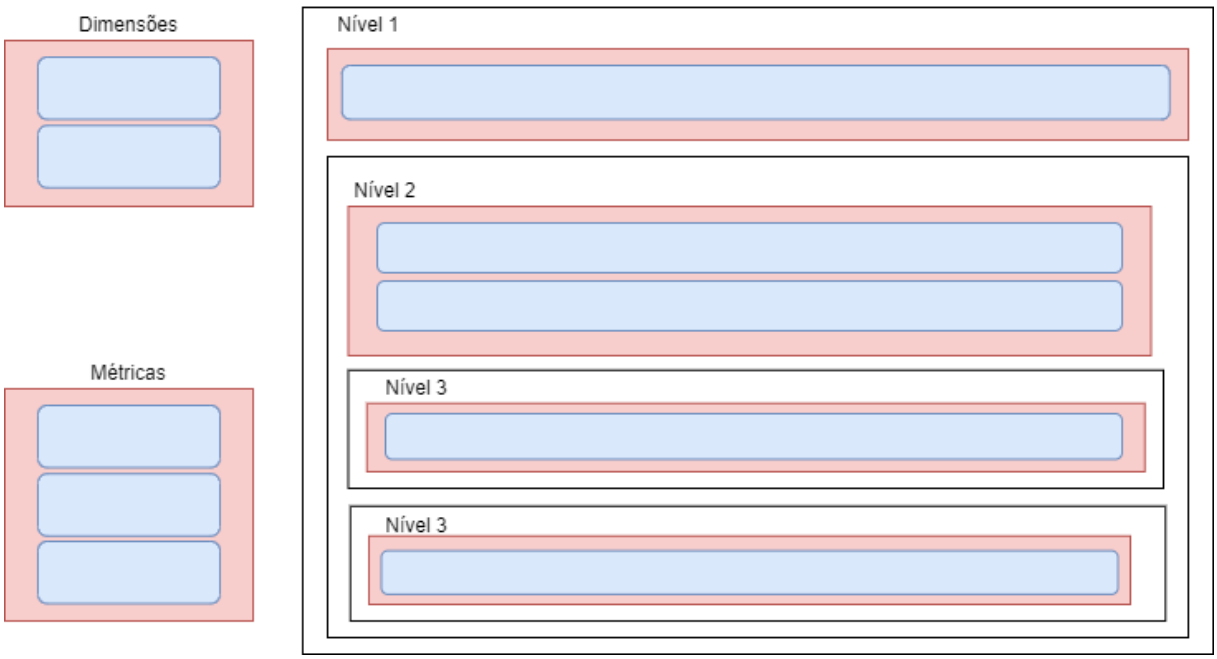


Figura 4.9: Representação da vista final

com o nome do ficheiro a eliminar. A outra acção é carregar uma versão de análise para a página de alteração de análise, em que é feito um pedido ao servidor para obter toda a informação de uma dada versão.

Uma métrica e uma dimensão contêm campos diferentes provenientes da análise. Todos os campos de uma métrica já estão incluídos nos de uma dimensão. No entanto, existem vários campos numa dimensão que não estão numa métrica, como por exemplo, os tipos de valores presentes nessa coluna ou o tipo de domínio geográfico. Logo, quando o utilizador altera uma métrica para uma dimensão, de forma a manter os dados da meta-informação consistentes, é necessário realizar uma nova análise de forma a obter os dados. Uma análise nova ao ficheiro todo seria desnecessária pois basta uma análise à coluna em específico. A análise de Ribeiro [20] usa um ciclo que analisa cada coluna. Foi criado o objecto que representa a coluna e chamamos directamente o método que analisa uma coluna, com a diferença que em vez de analisar os dados de forma a perceber se é um métrica ou dimensão forçamos o algoritmo a classificar a coluna como dimensão. Tendo a análise completa devolvemos a nova dimensão.

5

Instalação da solução numa máquina virtual

No desenvolvimento de software, a disponibilização e instalação da solução desenvolvida é parte integrante do processo de desenvolvimento [1]. Este capítulo descreve o processo de instalação da solução numa máquina virtual, alojado num sistema de virtualização existente no ISEL.

5.1 Máquina Virtual

De forma a facilitar o acesso à aplicação esta foi *deployed* numa máquina virtual, com o sistema operativo Ubuntu 18.04, alojada numa infraestrutura do ISEL. Desenvolver a aplicação no Visual Studio em modo *development* fornece várias funcionalidades automaticamente que têm que ser consideradas na transição para o modo *production* na máquina virtual. Para nos orientar pelo processo utilizámos um guia oficial publicado pela Microsoft [9].

Deste modo, foi necessário instalar todos os programas da aplicação. Designadamente, o *.NET Core runtime* para compilar a aplicação, *Microsoft SQL Server* como base de dados, o *NGINX* para funcionar como um *reverse proxy server* e *RabbitMQ* para gerir as mensagens entre o servidor e o serviço de análise. Este passo é diferente de uma disponibilização noutro tipo de soluções existentes em nuvem (*platform as a service*), onde

parte da complexidade de instalação e configuração da infraestrutura necessária para suportar o tipo de aplicação que desenvolvemos é escondida do programador.

5.1.1 Ficheiros de serviço

De forma a começar, parar e monitorizar a aplicação foi utilizado o software *systemd*. Este software permite a criação de um ficheiro de serviço que fica responsável pela aplicação. Este serviço pode ser iniciado através do comando `sudo systemctl start <nome do ficheiro>`, substituindo `start` por `stop` ou `status` é possível parar ou saber o estado do serviço, respetivamente. É, também possível, observar todos os *logs* e eventos processados através do comando `journalctl`, o que se torna bastante útil no *debug* da aplicação. O serviço oferece também a possibilidade de controlar, quando recomeçar, a aplicação no eventual erro que pare o seu funcionamento. O ficheiro pode conter também variáveis de ambiente sendo o sítio ideal para guardar dados confidenciais, tais como, as credenciais do **OAUTH 2.0** do *Google* e do *Facebook* para a autenticação externa. A Listagem 5.1 apresenta o ficheiro de serviço criado para o servidor.

```
1  [Unit]
2  Description=DataAnnotation web App running on Ubuntu
3
4  [Service]
5  WorkingDirectory=/home/vroot/DataAnnotation/DataAnnotation
6  ExecStart=/usr/bin/dotnet /home/vroot/DataAnnotation/DataAnnotation/bin/
       Release/netcoreapp3.1/publish/DataAnnotation.dll
7  Restart=always
8  RestartSec=10
9  KillSignal=SIGINT
10 SyslogIdentifier=dotnet-DataAnnotation
11 User=root
12 Environment=ASPNETCORE_ENVIRONMENT=Production
13 Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false
14 Environment=Authentication__Google__ClientId=<google_id>
15 Environment=Authentication__Google__ClientSecret=<google_secret>
16 Environment=Authentication__Facebook__AppId=<facebook_id>
17 Environment=Authentication__Facebook__AppSecret=<facebook_secret>
18 [Install]
19 WantedBy=multi-user.target
```

Listagem 5.1: Ficheiro de serviço responsável da aplicação no servidor

5.1.2 Reverse proxy server - NGINX

Um servidor *proxy* é um servidor intermediário que reencaminha solicitações de conteúdo de vários clientes para diferentes servidores na Internet. Um servidor *reverse*

proxy é um tipo de servidor que normalmente fica atrás da *firewall* numa rede privada e redireciona as solicitações do cliente ao servidor *backend* apropriado. Um *reverse proxy* fornece um nível adicional de abstração e controlo para garantir o fluxo uniforme do tráfego de rede entre clientes e servidores. As mais comuns utilizações para este tipo de servidor[1] são:

1. *Load Balancing*, o servidor encontra-se entre clientes e o servidor de *backend*, e, devido a este facto, controla para qual servidor *backend* enviar o pedido de forma a não sobrecarregar nenhum. Se um servidor ficar inoperacional, pode redirecionar o tráfego para os restantes;

2. *Web acceleration*, os servidores *reverse proxy* podem compactar dados de entrada e saída, bem como armazenar em cache conteúdo frequentemente solicitado, o que acelera o fluxo de tráfego entre clientes e servidores. Estes podem também executar tarefas adicionais, como criptografia SSL (*Secure Socket Layer*), para aliviar a carga dos servidores Web, melhorando o seu desempenho;

3. *Security and anonymity*, ao interceptar pedidos direcionados aos servidores *backend*, o servidor *reverse proxy* protege as suas identidades. Também assegura que múltiplos servidores podem ser acedidos por apenas um URL independentemente da estrutura de rede local.

Foi então implementado NGINX [13] como um servidor *reverse proxy* de forma a aumentar a segurança e escalabilidade da aplicação. NGINX contém também um ficheiro de serviço *systemd* de maneira a controlar o serviço. O serviço pode então ser começado através do comando: `sudo service nginx start`. É, no entanto, necessário alterar o ficheiro de configuração de forma a ligar com o servidor *backend*. Configurámos então de forma a que o servidor *reverse proxy* esteja à espera de pedidos no próprio IP da máquina virtual e que redirecione os pedidos para o servidor da aplicação, nesta caso *localhost*. A Listagem 5.2 apresenta o ficheiro de configuração para o NGINX.

HTTPS A aplicação trata com dados sensíveis que não podem ser comprometidos. No entanto, para o servidor confirmar que um utilizador está autenticado, este tem que enviar um *token* nos *headers* do pedido, nomeadamente o *access token* que, se estivar válido fornece ao utilizador a possibilidade de aceder às páginas que requerem autenticação. Porém, se este *token* for enviado em claro, torna-se numa falha de segurança, em específico contra ataques de *Man-in-the-Middle* (MITM). Este ataque consiste

[1] https://www.nginx.com/resources/glossary/reverse-proxy-server/

num atacante que se coloca entre o cliente e o servidor, como ilustra a Figura 5.1. O cliente pensa que está a comunicar diretamente com o servidor, mas na realidade toda a comunicação está a passar pelo atacante tendo, este acesso a todos os pedidos realizados. Se, o *access token* for enviado em claro, este atacante tendo agora acesso a ele pode aceder à aplicação como se estivesse autenticado mesmo sem conhecer as credenciais do utilizador.

```
server {
        listen                  *:443 ssl;
        server_name ""
      10.62.73.28
      ;
    ...
        #Redirects all traffic
        location / {
      proxy_pass     https://localhost:5001;
      proxy_http_version  1.1;
      proxy_set_header  Upgrade $http_upgrade;
      proxy_set_header  Connection keep-alive;
      proxy_set_header  Host $host;
      proxy_cache_bypass  $http_upgrade;
      proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header  X-Forward-Proto $scheme;
    }
```

Listagem 5.2: Ficheiro de configuração - NGINX

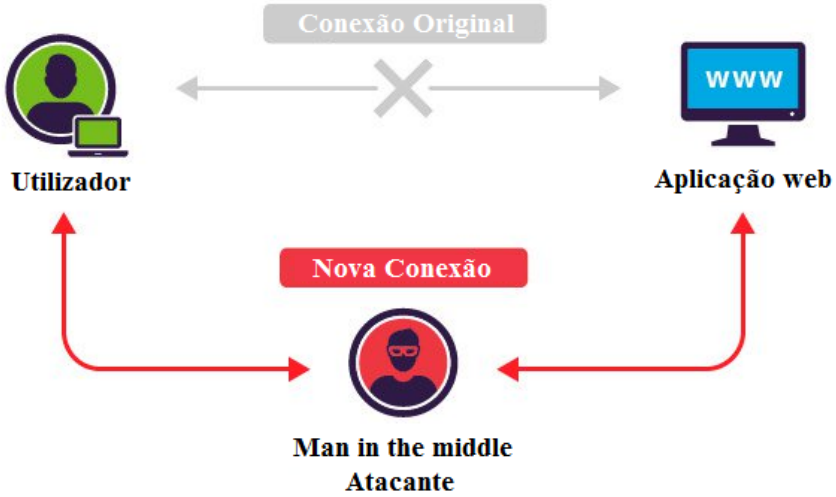


Figura 5.1: *Man-in-the-middle* (MITM), representação visual. Imagem adaptada de *Hacker Noon*[a]

[a] https://hackernoon.com/mitm-man-in-the-middle-attack-eavesdropping-at-its-best-mhih3z6b

O protocolo **Hypertext Transfer Protocol Secure** (HTTPS) [10] é uma solução para proteger contra este tipo de ataques. Este protocolo consiste essencialmente numa extensão do protocolo HTTP onde os dados estão encriptados. O servidor e o NGINX foram ambos configurados para comunicação HTTPS. Foi necessário, então, criar um certificado que servirá para encriptar a comunicação. Este certificado foi criado através da *toolkit* OpenSSL [15] utilizando o comando `openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365`.

O certificado tem o formato X.509 que é o *standard* para certificados utilizados no protocolo HTTPS. Contém uma chave pública, uma entidade e está ou não assinado por uma autoridade de certificação. Neste caso o certificado gerado é um *self-signed*, ou seja, não está assinado através de uma autoridade de certificação. Este certificado não é o ideal mas é suficiente para os objetivos da aplicação. A chave privada é criada através do algorimo RSA, contém 4096 bits e o certificado é válido durante 365 dias.

5.2 Testes de desempenho

Foram realizados alguns testes de desempenho. A ideia é avaliar o processo a "correr" numa máquina local e numa máquina virtual. Nomeadamente, em relação aos tempos de *upload* de um ficheiro e à duração de uma análise.

***Upload* de ficheiros** Para avaliarmos o desempenho da aplicação, medimos o tempo de *upload* de um e de mais ficheiros. É importante referir que para realizar os testes na máquina virtual é necessário ligar-se à VPN do ISEL, a VPN tem um impacto no desempenho das comunicações. Como exemplo, uma ligação de cerca de 85MB de *download* e 21MB de *upload* passou para 20MB de *download* e 2MB de *upload* quando ligado através da VPN.

O primeiro teste consiste no *upload* do ficheiro de censos de Portugal com um tamanho de 87.2MB, o segundo teste consiste no *upload* de três copias desse mesmo ficheiro em simultâneo. Os resultados destes testes encontram-se na Tabela 5.1.

Tabela 5.1: Desempenho do *upload* de ficheiros nas máquinas local e virtual (tempo em segundos)

# ficheiros	Máquina Local	Máquina Virtual
1	2seg	390seg
3	6seg	1240seg

Podemos observar que as diferenças de tempo entre a máquina virtual e local são provavelmente devido à VPN e que o upload de mais ficheiros em simultâneo não tem grande impacto no desempenho visto que 3 ficheiros em simultâneo demoram cerca do triplo de apenas um.

Análise de desempenho da aplicação Para avaliarmos o desempenho da aplicação, medimos o tempo que demora uma análise, para um e mais ficheiros. O primeiro teste consiste na análise do mesmo ficheiro de censos e o segundo teste consiste em começar a análise de duas cópias desse ficheiro em simultâneo. Os resultados destes testes encontram-se na Tabela 5.2. É importante referir que a máquina local onde foram realizados os testes tem 16GB de RAM contra os 4GB da VM.

Tabela 5.2: Desempenho da análise da aplicação nas máquinas local e virtual (tempo em segundos

# ficheiros	Máquina Local	Máquina Virtual
1	215seg	335seg
2	276seg	875seg

No segundo teste, ambas as análises na máquina local, concluíram praticamente ao mesmo tempo (diferença de 3seg) devido a estarem a ser realizadas em paralelo através da *thread pool*. Podemos observar que a máquina virtual demora significativamente mais tempo a realizar a análise. Esta discrepância é ainda mais notável ao analisar dois ficheiros. A diferença é provavelmente devido à RAM disponível na máquina virtual pois para analisar um ficheiro *csv* este é colocado num objeto chamado *DataTable* que representa o ficheiro por completo numa tabela de dados ocupando assim um tamanho significativo de memória.

6

Conclusão

6.1 Conclusões

No início deste projeto foi identificado um problema "Dificuldade em interpretar corretamente ficheiros de dados provenientes de diferentes fornecedores, cada um com a sua forma de os disponibilizar". Estes dados, geralmente em formato *csv*, contêm vários níveis de detalhe e diferentes zonas geográficas. No entanto, devido à total ausência de um ficheiro de metadados, a correta identificação destes dados de forma manual consiste numa tarefa complexa e demorada. Ribeiro [20] propôs então um algoritmo semi-automático que, após uma análise extensa do ficheiros de dados, gera um ficheiro com meta-informação que auxilia a compreensão do conteúdo deste. No entanto, este algoritmo pode conter alguns erros na sua análise devido a diversos fatores, por exemplo um erro sintático no nome da coluna.

O objetivo deste projeto foi desenvolver uma aplicação Web que tomando partido do algoritmo de Ribeiro, permite que um qualquer utilizador, após autenticação, realize o *upload* de ficheiros *csv* e obtenha a análise deste. A aplicação tem uma representação visual do ficheiro de metadados gerado. O utilizador pode, também, alterar a análise de forma a corrigir eventuais erros e, por fim, descarregar a meta-informação.

No início do trabalho, foram avaliadas algumas aplicações já desenvolvidas no sentido de identificar funcionalidades já existentes e por forma a contribuir com algo inovador.

Depois deste estudo, foram escolhidas as tecnologias apresentadas atrás. Seguidamente foram implementadas as funcionalidades pretendidas, nomeadamente 1) a parte

de autenticação de um utilizador utilizando a *framework IdentityServer*. Esta *framework* fornece o seu próprio modelo de base de dados e varias funcionalidades de autenticação tais como o registo e o *login* de um utilizador, 2) *upload* de ficheiros *csv* e realizar a sua análise, aproveitando o algoritmo desenvolvido por Ribeiro [20] e, por fim 3) a página onde é possível alterar a meta-informação gerada de forma a corrigir erros existentes.

Concluída a aplicação, foi possível colocá-la em modo de produção numa máquina virtual do ISEL onde foi preciso instalar todas as tecnologias necessárias para o seu correto funcionamento.

Deste modo, podemos concluir que toda a aplicação Web nunca chegará a um estado imutável, ou seja, haverá sempre algo que poderá ser alterado, melhorado e adicionado como se lê na secção 6.2.

6.2 Trabalho futuro

***Worker Service - File System* distribuído** O processo mais demorado da aplicação é o de análise de um ficheiro. Logo, de forma a melhorar o desempenho do serviço que a realiza este devia estar a correr na sua própria máquina. No entanto devido ao facto que o servidor e o serviço ambos necessitam de ter acesso aos ficheiros *uploaded* pelos utilizadores que se encontram no *file system* isto não é possível sem o uso de um *file system* distribuído que ambos conseguissem aceder.

Cancelar o *upload* de um ficheiro seria outra aspeto relevante a adicionar à aplicação pois o utilizador pode sem querer fazer *upload* de um ficheiro com tamanho significativo ficando sem a possibilidade de cancelar a operação.

Cancelar uma análise a decorrer seria outro cancelamento relevante. Esta funcionalidade seria mais difícil de implementar envolvendo comunicação com o serviço de mensagens e o cancelamento do trabalho de uma *thread* da *thread pool*.

Email de confirmação Quando o utilizador se regista este deve fornecer o seu email, no entanto, nenhum email a confirmar é enviado sendo esta outra funcionalidade relevante a acrescentar.

Maior customização na análise Neste momento a aplicação web apenas deixa alterar as colunas entre os diferentes níveis de detalhe que a análise original definiu. Seria

também relevante deixar o utilizador acrescentar níveis de detalhe aos existentes assim como definir níveis de detalhe nas dimensões pois neste momento não existem. O que implica também expandir a análise de forma a que também analise os níveis de detalhe presentes nas dimensões.

Referências

[1] M. Shahin, M. Ali Babar & L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices", *IEEE Access*, vol. 5, páginas 3909–3943, 2017.

[2] Pedro Amaral, "Smart Geo-Catalog", Tese de Mestrado, Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia, Portugal, 2015.

[3] OASIS. (2012). Advanced Message Queuing Protocol, URL: https://web.archive.org/web/20141010172100/http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html. Accessed: 10-09-2020.

[4] Emily Morehouse, Matt Zabriskie, Nick Uraltsev & Rubén Norte. (2020). HTTP client for the browser and node.js: axios GitHub repository, URL: https://github.com/axios/axios. Accessed: 25-05-2020.

[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] 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.

[7] Ciprian Dobre & Fatos Xhafa, "Intelligent services for big data science", *Future Generation Computer Systems*, vol. 37, páginas 267–281, 2014.

[8] Gil Fink & Ido Flatow, *Pro Single Page Application Development*. Springer, 2014.

[9] Sourabh Shirhatti. (2020). Host asp.net core on linux with nginx, URL: https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-3.1. Accessed: 06-09-2020.

[10] Google. (2020). Secure your site with HTTPS, URL: https://support.google.com/webmasters/answer/6073543?hl=en. Accessed: 07-09-2020.

[11] Brock Allen & Dominick Baier. (2020). Identity Server, URL: https://identityserver4.readthedocs.io/en/latest/. Accessed: 07-06-2020.

[12] W3. (2013). The Multipart Content-Type, URL: https://www.w3.org/Protocols/rfc1341/7_2_Multipart.html. Accessed: 15-04-2020.

[13] NGINX. (2020). Nginx - what is a reverse proxy server?, URL: https://www.nginx.com/resources/glossary/reverse-proxy-server/. Accessed: 07-09-2020.

[14] IETF OAuth Working Group. (2020). Oauth 2.0, URL: https://oauth.net/2/. Accessed: 01-05-2020.

[15] OpenSSL. (2020). OpenSSL Cryptography and SSL/TLS, URL: https://www.openssl.org/. Accessed: 07-09-2020.

[16] RabbitMQ. (2020). AMQP 0-9-1 Model Explained, URL: https://www.rabbitmq.com/tutorials/amqp-concepts.html. Accessed: 04-09-2020.

[17] Sam Allen. (2007). C# path.getrandomfilename method, URL: http://www.dotnetperls.com/path-getrandomfilename. Accessed: 05-06-2020.

[18] Alex Reardon. (2020). Drag and drop with React: react-beautiful-dnd, URL: https://github.com/atlassian/react-beautiful-dnd. Accessed: 07-06-2020.

[19] Sam Corcos. (2020). HTML5 drag-drop zone with React.js: dropzone, URL: https://github.com/react-dropzone/react-dropzone. Accessed: 01-05-2020.

[20] Nuno Ribeiro, "Anotação e Extração Semi-Automática de dados multidimensionais", Tese de Mestrado, Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia, Portugal, 2019.

[21] Neoteric in Medium. (2020). Single-page application vs. multiple-page application, URL: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58. Accessed: 01-05-2020.

[22] 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.

[23] Steve Smith & 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.