

Processamento de Linguagens e Compiladores (3^o ano LCC)

Trabalho Prático 1

Relatório de Desenvolvimento

Grupo 17

Problema 3

José Pedro Gomes Ferreira
A91636

Pedro Paulo Costa Pereira
A88062

Tiago André Oliveira Leite
A91693

15 de Novembro de 2021

Conteúdo

1	Introdução	2
2	Problema Proposto	3
2.1	Descrição	3
2.2	Requisitos	3
3	Concepção/desenho da Resolução	4
3.1	Organização	4
3.2	Funcionalidades	4
3.2.1	Executar o programa.	4
3.2.2	O menu.	4
3.2.3	Nome dos utilizadores ordenados e respetiva entidade.	5
3.2.4	Entidades ordenadas e número de utilizadores registados em cada uma.	5
3.2.5	Distribuição dos utilizadores por nível de acesso.	5
3.2.6	Utilizadores agrupados por entidade, ordenados por nome e entidade.	6
3.2.7	Mostrar alguns indicadores.	6
3.2.8	Imprimir num ficheiro Json os 20 primeiros registos.	6
4	Demonstração de Funcionamento	8
4.1	O menu.	8
4.2	Nome dos utilizadores ordenados e respetiva entidade.	8
4.3	Entidades ordenadas e número de utilizadores registados em cada uma.	10
4.4	Distribuição dos utilizadores por nível de acesso.	10
4.5	Utilizadores agrupados por entidade, ordenados por nome e entidade.	12
4.6	Mostrar alguns indicadores.	14
4.7	Imprimir num ficheiro Json os 20 primeiros registos.	15
5	Conclusão	20
A	Código do Programa	21

Capítulo 1

Introdução

Neste documento vamos explicar a solução que implementamos para a resolução do problema proposto no âmbito da unidade curricular de Processamento de Linguagens e Compiladores.

O problema proposto consiste em analisar o ficheiro de texto *calv-users.txt* e através da utilização da linguagem Python e da biblioteca de expressões regulares 're' extrair informação de forma a produzir alguns resultados.

No desenvolvimento do programa procuramos utilizar o conhecimento adquirido nas aulas, esperando por isso que o resultado final cumpra todos os requisitos.

Capítulo 2

Problema Proposto

2.1 Descrição

Construa um ou vários programas para processar o texto ' clav-users.txt ' em que campos de informação têm a seguinte ordem: nome, email, entidade, nível, número de chamadas ao backend, com o intuito de calcular alguns resultados conforme solicitado a seguir:

- Produz uma listagem apenas com o nome e a entidade do utilizador, ordenada alfabeticamente por nome;
- Produz uma lista ordenada alfabeticamente das entidades referenciadas, indicando, para cada uma, quantos utilizadores estão registados;
- Qual a distribuição de utilizadores por níveis de acesso?
- Produz uma listagem dos utilizadores, agrupados por entidade, ordenada primeiro pela entidade e dentro desta pelo nome;
- Por fim, produz os seguintes indicadores:
 1. Quantos utilizadores?
 2. Quantas entidades?
 3. Qual a distribuição em número por entidade?
 4. Qual a distribuição em número por nível?

Para terminar, deve imprimir os 20 primeiros registos num novo ficheiro de output mas em formato Json .

2.2 Requisitos

- Utilização da linguagem Python.
- Resolver o problema com uso de expressões regulares.
- Utilizar o modulo 're'.

Capítulo 3

Concepção/desenho da Resolução

3.1 Organização

Por forma a tornar a resolução do trabalho mais simples, decidimos criar uma função específica para resolver cada uma das alíneas do problema. Assim sendo, a solução do problema vai ser composta por 6 funções mais uma que vai servir de menu para o utilizador poder escolher qual das funcionalidades do programa quer utilizar.

Na execução do programa todas as linhas do ficheiro 'clav-users.txt' são lidas para uma variável que depois será utilizada por cada uma das funções.

Por uma questão de simplicidade, sempre que for necessário ordenar, a ordem utilizada é a ordem alfabética.

3.2 Funcionalidades

Todas as funções descritas neste capítulo podem ser encontradas no anexo A do documento.

3.2.1 Executar o programa.

Para executar o programa o utilizador terá que colocar na mesma diretoria os ficheiros 'main.py' e 'clav-users.txt'. De seguida basta abrir o terminal nessa diretoria e executar o comando:

```
>> python3 main.py
```

3.2.2 O menu.

O menu do programa é implementado pela função `menu()`.

Esta função cria um ciclo `while` que mostra ao utilizador todas as opções de funcionalidades disponíveis. De seguida lê a opção escolhida pelo utilizador e caso a opção esteja entre 1 e 6 vai invocar a função que implementa a funcionalidade selecionada. Se a opção escolhida pelo utilizador for menor que 0 ou maior que 6 vai ser mostrada uma mensagem de erro sendo solicitado ao utilizador que escolha novamente. Caso a opção seja 0, o programa termina a sua execução.

Para tornar a experiência mais agradável aos olhos, entre a execução de cada funcionalidade, é feita uma limpeza do ecrã através da impressão de 50 parágrafos.

3.2.3 Nome dos utilizadores ordenados e respetiva entidade.

Esta funcionalidade é implementada pela função `name_entity_list()`.

Na sua implementação vamos iterar por cada linha do texto e em cada uma vamos procurar o nome de utilizador e a entidade correspondente. Para capturar o nome do utilizador vamos utilizar a função `re.match` com a expressão regular `'(\w+\s*(-?\w+\s*)*\b)'` e para capturar a entidade utilizamos a função `re.search` com a expressão regular `'ent_\w*'`. De seguida armazenamos o resultado num dicionário em que a chave é o nome do utilizador e o valor é uma lista com as entidades desse utilizador. Caso o utilizador ainda não exista no dicionário é criada uma nova entrada, caso contrário, a entidade capturada é acrescentada na lista do utilizador.

Por fim, transformamos o dicionário numa lista composta pelos pares (nome de utilizador, entidade(s)), ordenada por nome de utilizador, e fazemos a impressão de cada utilizador e respetiva(s) entidade(s).

Nota: Na implementação desta função decidimos diferenciar utilizadores cujo nome apenas difere no facto de as letras estarem em maiúsculas ou minúsculas. Por exemplo `'xxxx'`, `'XXXX'` e `'Xxxx'` são utilizadores distintos.

3.2.4 Entidades ordenadas e número de utilizadores registados em cada uma.

Esta funcionalidade é implementada pela função `entity_num_elements_list()`.

Na sua implementação vamos iterar por cada linha do texto e através da função `re.search` com a expressão regular `'ent_\w*'` vamos capturar a entidade presente nessa linha. A entidade capturada é armazenada num dicionário cuja chave é o nome da entidade e o valor é o número de utilizadores nessa entidade. Caso a entidade ainda não exista no dicionário é criada uma nova entrada com o valor 1, caso contrário, o valor dessa entidade no dicionário é incrementado em 1.

Por fim, transformamos o dicionário numa lista composta pelos pares (entidade, nº de utilizadores), ordenada por entidade, e fazemos a impressão de cada entidade e respetivo nº de utilizadores.

Nota: Na implementação desta função decidimos diferenciar utilizadores cujo nome apenas difere no facto de as letras estarem em maiúsculas ou minúsculas. Por exemplo `'xxxx'`, `'XXXX'` e `'Xxxx'` são utilizadores distintos.

3.2.5 Distribuição dos utilizadores por nível de acesso.

Esta funcionalidade é implementada pela função `dist_users_level()`.

Na sua implementação vamos iterar por cada linha do texto e partir cada uma destas pelo separador `'::'` com o uso da função `re.split`. De seguida vamos capturar o nome de utilizador no primeiro elemento (index 0) da linha partida com a função `re.match` e a expressão regular `'(\w+\s*(-?\w+\s*)*\b)'` e vamos capturar o nível de acesso no quarto elemento da linha partida (index 3) com a função `re.search` e a expressão regular `'\d+\.\d*'`. O resultado de capturar o nível de acesso e nome de utilizador são armazenados num dicionário em que a chave é o nível de acesso e o valor é um conjunto com os utilizadores com esse nível de acesso. Optamos por um conjunto para evitar repetições, uma vez que existem utilizadores com mais do que um nível de acesso. Caso o nível de acesso capturado não exista no dicionário, é criada uma entrada cuja chave é o nível de acesso e cujo valor é um conjunto com o utilizador capturado, caso contrário, o utilizador é acrescentado ao conjunto de utilizadores já existente para aquela chave. Para ser mais fácil de contar o nº de utilizadores total utilizamos também um conjunto onde vamos colocando cada utilizador capturado.

Por fim, transformamos o dicionário numa lista composta pelos pares (nível de acesso, utilizadores) ordenada por nível de acesso e fazemos uma impressão dos utilizadores presentes em cada nível de acesso assim como

a percentagem de utilizadores por nível.

Nota: Na implementação desta função decidimos diferenciar utilizadores cujo nome apenas difere no facto de as letras estarem em maiúsculas ou minúsculas. Por exemplo 'xxxx', 'XXXX' e 'Xxxx' são utilizadores distintos.

3.2.6 Utilizadores agrupados por entidade, ordenados por nome e entidade.

Esta funcionalidade é implementada pela função `name_entity_group()`.

Na sua implementação vamos iterar por cada linha do texto e em cada uma vamos procurar o nome de utilizador e a entidade correspondente. Para capturar o nome do utilizador vamos utilizar a função `re.match` com a expressão regular `'(\w+\s*(-?\w+\s*)*\b)'` e para capturar a entidade utilizamos a função `re.search` com a expressão regular `'ent_\w*'`. De seguida armazenamos o resultado num dicionário em que a chave é a entidade e o valor é uma lista com os utilizadores dessa entidade. Caso a entidade ainda não exista no dicionário é criada uma nova entrada, caso contrário, o utilizador capturado é acrescentado na lista da entidade.

Por fim, transformamos o dicionário numa lista composta pelos pares (entidade, utilizador), ordenada por entidade, e fazemos a impressão de cada entidade e respetivos utilizadores ordenados por nome.

Nota: Na implementação desta função decidimos diferenciar utilizadores cujo nome apenas difere no facto de as letras estarem em maiúsculas ou minúsculas. Por exemplo 'xxxx', 'XXXX' e 'Xxxx' são utilizadores distintos.

3.2.7 Mostrar alguns indicadores.

Esta funcionalidade é implementada pela função `indicators()`.

Na sua implementação vamos iterar por cada linha do texto e partir cada uma destas pelo separador `'::'` com o uso da função `re.split`. De seguida vamos capturar o nome de utilizador no primeiro elemento (index 0) da linha partida com a função `re.match` e a expressão regular `'(\w+\s*(-?\w+\s*)*\b)'` vamos capturar a entidade no terceiro elemento da linha partida (index 2) com a função `re.search` e a expressão regular `'ent_\w*'`, e vamos capturar o nível de acesso no quarto elemento da linha partida (index 3) com a função `re.search` e a expressão regular `'\d+\.\d*'`. Os nomes de utilizadores capturados anteriormente são armazenados num conjunto, para evitar repetições, enquanto que as entidades e níveis de acesso capturados anteriormente são armazenados em dois dicionários. Num dos dicionários o par chaves e valores são a entidade e respetivo número de utilizadores enquanto que no outro o par chave e valores são o nível de acesso e respetivo número de utilizadores.

Por fim, fazemos uma impressão do número total de utilizadores, o número total de entidades, o número de utilizadores por entidade e o número de utilizadores por nível de acesso.

Nota: Na implementação desta função decidimos diferenciar utilizadores cujo nome apenas difere no facto de as letras estarem em maiúsculas ou minúsculas. Por exemplo 'xxxx', 'XXXX' e 'Xxxx' são utilizadores distintos.

3.2.8 Imprimir num ficheiro Json os 20 primeiros registos.

Esta funcionalidade é implementada pela função `json_20()`.

Na sua implementação vamos iterar por cada uma das 20 primeiras linhas do texto e partir cada uma destas pelo separador `'::'` com o uso da função `re.split`. De seguida vamos capturar o nome de uti-

lizador no primeiro elemento (index 0) da linha partida com a função `re.match` e a expressão regular `'(\w+\s*(-?\w+\s*)*\b)'`, vamos capturar o email de utilizador no segundo elemento (index 1) da linha partida com a função `re.search` e a expressão regular `'(\w+|\.|@|_|-)+'`, vamos capturar a entidade no terceiro elemento da linha partida (index 2) com a função `re.search` e a expressão regular `'ent_\w*'`, vamos capturar o nível de acesso no quarto elemento da linha partida (index 3) com a função `re.search` e a expressão regular `'\d+\.\d*'`, e vamos capturar o número de chamadas ao backend no quinto elemento da linha partida (index 4) com a função `re.search` e a expressão regular `'\d+'`. O resultado de cada linha é armazenado numa lista de tuplos (node de utilizador, email, entidade, nível de acesso, número de chamadas ao backend).

Finalizadas as capturas pedimos ao utilizador para escolher qual o nome do ficheiro para o qual quer imprimir os dados recolhidos. Depois abrimos o ficheiro e procedemos à impressão de cada registo (tuplo armazenado na lista) num formato json.

Capítulo 4

Demonstração de Funcionamento

4.1 O menu.

*** Selecione Opção ***

1. Listagem com o nome e a entidade do utilizador, ordenada alfabeticamente por nome.
 2. Lista ordenada alfabeticamente das entidades referenciadas, indicando, para cada uma, quantos utilizadores a utilizam.
 3. Distribuição de utilizadores por níveis de acesso.
 4. Utilizadores, agrupados por entidade, ordenada primeiro pela entidade e dentro desta pelo nome.
 5. Mostrar alguns indicadores.
 6. Imprimir os 20 primeiros registos num novo ficheiro de output mas em formato json.
 0. Sair.
- >>

4.2 Nome dos utilizadores ordenados e respetiva entidade.

>> 1

*** Utilizador : Entidade(s) ***

Alda do Carmo Namora Soares de Andrade : ent_FLUL
Alexadre Teixeira : ent_KEEP
Alexandra Lourenço : ent_DGLAB
Alexandra Maria Alves Coutinho Rodrigues : ent_UTAD
Alexandra Testes : ent_A3ES
Alexandre Teixeira : ent_A3ES
Aluno de DAW2020 : ent_A3ES
Ana Lúcia Cabrita Guerreiro : ent_CCDR
Ana Maria Teixeira Gaspar : ent_SGMF
António José Morim Brandão : ent_MdP
Carlos Barbosa : ent_A3ES
Carlos Matoso : ent_IEFP
Cármem Isabel Amador Francisco : ent_CMSNS
Cátia João Matias Trindade : ent_DGLAB
Cátia Trindade : ent_DGLAB

clara cristina rainho viegas : ent_DGLAB
CLAV-migrator : ent_A3ES
David Ferreira : ent_IEFP
DAW2020-teste : ent_A3ES
Design-DGLAB : ent_DGLAB
Duarte Freitas : ent_A3ES
Élia Cristina Viegas Pedro : ent_CCDR
Fernando Manuel Antunes Marques da Silva : ent_STI
Filipa Carvalho : ent_DGLAB
Filipe Ferreira Cardoso Leitão : ent_CMSPS
Formação DGLAB : ent_DGLAB
Frederico Pinto : ent_ACSS
jcm : ent_AAN
jcr-rep-entidade : ent_A3ES
Joana Braga : ent_IEFP
João Paulo de Melo Esteves Pereira : ent_APA
João Pimentel : ent_A3ES
José Carlos Leite Ramalho : ent_A3ES, ent_DGLAB
José Carlos Martins : ent_A3ES
Madalena Ribeiro : ent_DGLAB, ent_DGLAB
Manuel Monteiro : ent_A3ES
Maria Celeste Pereira : ent_DGLAB
Maria José Maciel Chaves : ent_DGLAB
Maria Leonor da Conceição Fresco Franco : ent_CCDR
Maria Matos de Almeida Talhada Correia : ent_ICNF
Maria Rita Gago : ent_DGLAB
Miguel Ferreira : ent_KEEP
Nuno Filipe Casas Novas : ent_CCDR
octavio : ent_A3ES
Paulo Lima : ent_KEEP
Pedro Penteado : ent_DGLAB
PRI2020-teste : ent_A3ES
Regina Neves Lopes : ent_SGMF
Ricardo Almeida : ent_DGEG
Ricardo Canela : ent_BdP
Rui Araújo : ent_II
Rui Araújo Entidade : ent_AAN
Rui Araújo Simples : ent_LNEC
Sandra Cristina Patrício da Silva : ent_CMSNS
Silvestre Lacerda : ent_DGLAB
Sónia Isabel Ferreira Gonçalves Negrão : ent_CMABF
Sónia Patrícia Pinheiro Reis : ent_ICNF
Zélia Gomes : ent_DGLAB

Pressione Enter!

>>

4.3 Entidades ordenadas e número de utilizadores registados em cada uma.

>> 2

*** Entidade : Nº Utilizadores ***

ent_A3ES : 14
ent_AAN : 2
ent_ACSS : 1
ent_APA : 1
ent_BdP : 1
ent_CCDD : 4
ent_CMABF : 1
ent_CMSNS : 2
ent_CMSPS : 1
ent_DGEG : 1
ent_DGLAB : 16
ent_FLUL : 1
ent_ICNF : 2
ent_IEFP : 3
ent_II : 1
ent_KEEP : 3
ent_LNEC : 1
ent_MdP : 1
ent_SGMF : 2
ent_STI : 1
ent_UTAD : 1

Pressione Enter!

>>

4.4 Distribuição dos utilizadores por nível de acesso.

>> 3

*** Nível de Acesso : Distribuição ***

Nível 1 : 40%

* Alda do Carmo Namora Soares de Andrade
* Alexandra Maria Alves Coutinho Rodrigues
* Alexandra Testes
* Ana Lúcia Cabrita Guerreiro
* Ana Maria Teixeira Gaspar
* António José Morim Brandão
* Cármen Isabel Amador Francisco
* Élia Cristina Viegas Pedro

- * Fernando Manuel Antunes Marques da Silva
- * Filipe Ferreira Cardoso Leitão
- * jcm
- * jcr-rep-entidade
- * João Paulo de Melo Esteves Pereira
- * Maria Leonor da Conceição Fresco Franco
- * Maria Matos de Almeida Talhada Correia
- * Nuno Filipe Casas Novas
- * Regina Neves Lopes
- * Ricardo Almeida
- * Rui Araújo Entidade
- * Rui Araújo Simples
- * Sandra Cristina Patrício da Silva
- * Sónia Isabel Ferreira Gonçalves Negrão
- * Sónia Patrícia Pinheiro Reis

Nivel 2 : 12%

- * Aluno de DAW2020
- * Carlos Matoso
- * David Ferreira
- * DAW2020-teste
- * Joana Braga
- * octavio
- * PRI2020-teste

Nivel 3 : 2%

- * Ricardo Canela

Nivel 3.5 : 2%

- * Formação DGLAB

Nivel 4 : 14%

- * Cátia João Matias Trindade
- * Cátia Trindade
- * clara cristina rainho viegas
- * Filipa Carvalho
- * Madalena Ribeiro
- * Maria Celeste Pereira
- * Maria José Maciel Chaves
- * Zélia Gomes

Nivel 5 : 3%

- * Pedro Penteado
- * Silvestre Lacerda

Nivel 6 : 3%

- * Madalena Ribeiro
- * Maria Rita Gago

```
Nivel 7 : 26%
* Alexadre Teixeira
* Alexandra Lourenço
* Alexandre Teixeira
* Carlos Barbosa
* CLAV-migrator
* Design-DGLAB
* Duarte Freitas
* Frederico Pinto
* João Pimentel
* José Carlos Leite Ramalho
* José Carlos Martins
* Manuel Monteiro
* Miguel Ferreira
* Paulo Lima
* Rui Araújo
```

```
Pressione Enter!
>>
```

4.5 Utilizadores agrupados por entidade, ordenados por nome e entidade.

```
>> 4
```

```
*** Utilizadores agrupados por entidade ***
```

```
ent_A3ES:
* Alexandra Testes
* Alexandre Teixeira
* Aluno de DAW2020
* Carlos Barbosa
* CLAV-migrator
* DAW2020-teste
* Duarte Freitas
* jcr-rep-entidade
* João Pimentel
* José Carlos Leite Ramalho
* José Carlos Martins
* Manuel Monteiro
* octavio
* PRI2020-teste
```

```
ent_AAN:
* jcm
* Rui Araújo Entidade
```

ent_ACSS:

* Frederico Pinto

ent_APA:

* João Paulo de Melo Esteves Pereira

ent_BdP:

* Ricardo Canela

ent_CCDR:

* Ana Lúcia Cabrita Guerreiro

* Élia Cristina Viegas Pedro

* Maria Leonor da Conceição Fresco Franco

* Nuno Filipe Casas Novas

ent_CMABF:

* Sónia Isabel Ferreira Gonçalves Negrão

ent_CMSNS:

* Cármen Isabel Amador Francisco

* Sandra Cristina Patrício da Silva

ent_CMSPS:

* Filipe Ferreira Cardoso Leitão

ent_DGEG:

* Ricardo Almeida

ent_DGLAB:

* Alexandra Lourenço

* Cátia João Matias Trindade

* Cátia Trindade

* clara cristina rainho viegas

* Design-DGLAB

* Filipa Carvalho

* Formação DGLAB

* José Carlos Leite Ramalho

* Madalena Ribeiro

* Madalena Ribeiro

* Maria Celeste Pereira

* Maria José Maciel Chaves

* Maria Rita Gago

* Pedro Penteado

* Silvestre Lacerda

* Zélia Gomes

ent_FLUL:

* Alda do Carmo Namora Soares de Andrade

ent_ICNF:

* Maria Matos de Almeida Talhada Correia
* Sónia Patrícia Pinheiro Reis

ent_IEFP:

* Carlos Matoso
* David Ferreira
* Joana Braga

ent_II:

* Rui Araújo

ent_KEEP:

* Alexadre Teixeira
* Miguel Ferreira
* Paulo Lima

ent_LNEC:

* Rui Araújo Simples

ent_MdP:

* António José Morim Brandão

ent_SGMF:

* Ana Maria Teixeira Gaspar
* Regina Neves Lopes

ent_STI:

* Fernando Manuel Antunes Marques da Silva

ent_UTAD:

* Alexandra Maria Alves Coutinho Rodrigues

Pressione Enter!

>>

4.6 Mostrar alguns indicadores.

>> 5

*** Indicadores ***

Número de Utilizadores: 58

Número de Entidades: 21

Distribuição de utilizadores por entidade:

- * ent_A3ES : 14
- * ent_AAN : 2
- * ent_ACSS : 1
- * ent_APA : 1
- * ent_BdP : 1
- * ent_CCDD : 4
- * ent_CMABF : 1
- * ent_CMSNS : 2
- * ent_CMSPS : 1
- * ent_DGEG : 1
- * ent_DGLAB : 16
- * ent_FLUL : 1
- * ent_ICNF : 2
- * ent_IEFP : 3
- * ent_II : 1
- * ent_KEEP : 3
- * ent_LNEC : 1
- * ent_MdP : 1
- * ent_SGMF : 2
- * ent_STI : 1
- * ent_UTAD : 1

Distribuição de utilizadores por nível:

- * Nivel 1 : 23
- * Nivel 2 : 7
- * Nivel 3 : 1
- * Nivel 3.5 : 1
- * Nivel 4 : 8
- * Nivel 5 : 2
- * Nivel 6 : 2
- * Nivel 7 : 16

Pressione Enter!

>>

4.7 Imprimir num ficheiro Json os 20 primeiros registos.

>> 6

Digite Nome Do Ficheiro de Output!

>> json.txt

Ficheiro gerado com sucesso!

Pressione Enter!

>>

Conteúdo do ficheiro json.txt

```
{
  "registos":[
    {
      "utilizador":"Élia Cristina Viegas Pedro",
      "email":"epedro@ccdr-alg.pt",
      "entidade":"ent_CCDR",
      "nivel de acesso":"1",
      "número de chamadas ao backend":"0"
    },
    {
      "utilizador":"Formação DGLAB",
      "email":"lurdes.almeida@dglab.gov.pt",
      "entidade":"ent_DGLAB",
      "nivel de acesso":"3.5",
      "número de chamadas ao backend":"0"
    },
    {
      "utilizador":"Nuno Filipe Casas Novas",
      "email":"nuno.novas@ccdr-lvt.pt",
      "entidade":"ent_CCDR",
      "nivel de acesso":"1",
      "número de chamadas ao backend":"0"
    },
    {
      "utilizador":"Sónia Patrícia Pinheiro Reis",
      "email":"sonia.reis@icnf.pt",
      "entidade":"ent_ICNF",
      "nivel de acesso":"1",
      "número de chamadas ao backend":"0"
    },
    {
      "utilizador":"Sónia Isabel Ferreira Gonçalves Negrão",
      "email":"sonia.negrao@cm-albufeira.pt",
      "entidade":"ent_CMABF",
      "nivel de acesso":"1",
      "número de chamadas ao backend":"0"
    },
    {
      "utilizador":"Filipe Ferreira Cardoso Leitão",
      "email":"arquivo@cm-spsul.pt",
      "entidade":"ent_CMSPS",
      "nivel de acesso":"1",
      "número de chamadas ao backend":"0"
    }
  ]
}
```

```

},
{
  "utilizador":"Ana Lúcia Cabrita Guerreiro",
  "email":"alucia@ccdr-alg.pt",
  "entidade":"ent_CCDR",
  "nivel de acesso":"1",
  "número de chamadas ao backend":"0"
},
{
  "utilizador":"Alda do Carmo Namora Soares de Andrade",
  "email":"aandrade@letras.ulisboa.pt",
  "entidade":"ent_FLUL",
  "nivel de acesso":"1",
  "número de chamadas ao backend":"0"
},
{
  "utilizador":"Ricardo Almeida",
  "email":"ricardo.almeida@dgeg.gov.pt",
  "entidade":"ent_DGEG",
  "nivel de acesso":"1",
  "número de chamadas ao backend":"0"
},
{
  "utilizador":"Sandra Cristina Patrício da Silva",
  "email":"spatricio@mun-sines.pt",
  "entidade":"ent_CMSNS",
  "nivel de acesso":"1",
  "número de chamadas ao backend":"0"
},
{
  "utilizador":"Cátia João Matias Trindade",
  "email":"catia.trindade@dglab.gov.pt",
  "entidade":"ent_DGLAB",
  "nivel de acesso":"4",
  "número de chamadas ao backend":"0"
},
{
  "utilizador":"Ricardo Canela",
  "email":"tyty@tyty.pt",
  "entidade":"ent_BdP",
  "nivel de acesso":"3",
  "número de chamadas ao backend":"0"
},
{
  "utilizador":"Cátia Trindade",
  "email":"matiasjcatia@gmail.com",
  "entidade":"ent_DGLAB",
  "nivel de acesso":"4",

```

```

    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Miguel Ferreira",
    "email": "mferreira@keep.pt",
    "entidade": "ent_KEEP",
    "nível de acesso": "7",
    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Fernando Manuel Antunes Marques da Silva",
    "email": "fernando.marques.silva@marinha.pt",
    "entidade": "ent_STI",
    "nível de acesso": "1",
    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Ana Maria Teixeira Gaspar",
    "email": "ana.gaspar@sgmf.gov.pt",
    "entidade": "ent_SGMF",
    "nível de acesso": "1",
    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Maria Matos de Almeida Talhada Correia",
    "email": "MariaMatos.Correia@icnf.pt",
    "entidade": "ent_ICNF",
    "nível de acesso": "1",
    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Cármem Isabel Amador Francisco",
    "email": "carmem@mun-sines.pt",
    "entidade": "ent_CMSNS",
    "nível de acesso": "1",
    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Maria Leonor da Conceição Fresco Franco",
    "email": "leonor.mina@ccdr-lvt.pt",
    "entidade": "ent_CCDR",
    "nível de acesso": "1",
    "número de chamadas ao backend": "0"
  },
  {
    "utilizador": "Maria Rita Gago",
    "email": "m-rita.gago@dglab.gov.pt",
    "entidade": "ent_DGLAB",

```

```
        "nível de acesso":"6",  
        "número de chamadas ao backend":"3"  
    }  
]  
}
```

Capítulo 5

Conclusão

Com o projeto concluído esperamos ter cumprido todos os requisitos que nos foram propostos e que o nosso programa respeite os princípios da disciplina. Numa primeira abordagem tentando fazer a captura dos nomes de utilizador com a função `findAll`, no entanto não está a ser muito fácil obter resultados por causa do início de linha, pelo que optamos por em todos os exercícios iterar linha a linha. No problema que nos foi proposto, em cada uma das alíneas é nos pedido que façamos um programa, no entanto achamos que a utilização do menu, na qual podemos escolher qual a funcionalidade, torna a experiência do utilizador mais simples e agradável. Todos concordamos que o facto de o projecto ter sido desenvolvido na linguagem 'Python' e com recurso à biblioteca 're', facilitou bastante a sua implementação.

Apêndice A

Código do Programa

Ficheiro main.py

```
import re
import unicode

def name_entity_list():
    result = {}
    for line in text:
        user = re.match(r'(\w+\s*(-?\w+\s*))\b', line).group()
        entity = re.search(r'ent_\w*', line).group()
        if user not in result:
            result[user] = [entity]
        else:
            result[user].append(',')
            result[user].append(entity)
    result = list(result.items())
    result.sort(key=lambda x: unicode.unidecode(x[0].casefold()))
    print("\n*** Utilizador : Entidade(s) ***\n")
    for r in result:
        print(r[0], ":", "".join(r[1]))

def entity_num_elements_list():
    entities = {}
    for line in text:
        entity = re.search(r'ent_\w*', line).group()
        if entity not in entities:
            entities[entity] = 1
        else:
            entities[entity] += 1

    result = list(entities.items())
    result.sort()
    print("\n*** Entidade : Nº Utilizadores ***\n")
```

```

for r in result:
    print(r[0], ":", r[1])

def dist_users_level():
    levels = {}
    users = set()
    for line in text:
        broken_line = re.split(r'::', line)
        user = re.match(r'(\w+\s*(-?\w+\s*)*\b)', broken_line[0]).group()
        level = re.search(r'\d+\.\d*', broken_line[3]).group()
        if level not in levels:
            levels[level] = set()
            levels[level].add(user)
        else:
            levels[level].add(user)
        users.add(user)
    result = list(levels.items())
    result.sort()
    total_users = len(users)
    print("\n*** Nivel de Acesso : Distribuição ***\n")
    for r in result:
        print(f"Nivel {r[0]} : {round((len(r[1])/total_users)*100)}%" )
        for user in sorted(r[1], key=lambda x: unicode.unidecode(x.casefold())):
            print("*",user)
        if(result.index(r) != len(result)-1):
            print("")

def name_entity_group():
    result = {}
    for line in text:
        name = re.match(r'(\w+\s*(-?\w+\s*)*\b)', line).group()
        entity = re.search(r'ent_\w*', line).group()
        if entity not in result:
            result[entity] = [name]
        else:
            result[entity].append(name)

    entities = list(result.keys())
    entities.sort()
    print("\n*** Utilizadores agrupados por entidade ***\n")
    for entity in entities:
        print(f"{entity}:")
        result[entity].sort(key=lambda x: unicode.unidecode(x.casefold()))
        for user in result[entity]:
            print("*",user)
        print("")

```

```

def indicators():
    users = set()
    entities = {}
    levels = {}
    for line in text:
        broken_line = re.split(r'::', line)
        users.add(re.match(r'(\w+\s*(\w+\s*)*\b)', broken_line[0]).group())
        entity = re.search(r'ent_\w*', broken_line[2]).group()
        level = re.search(r'\d+\.? \d*', broken_line[3]).group()

        if entity not in entities:
            entities[entity] = 1
        else:
            entities[entity] += 1
        if level not in levels:
            levels[level] = 1
        else:
            levels[level] += 1

    print("\n*** Indicadores ***\n")
    print("Número de Utilizadores:", len(users))
    print("")
    print("Número de Entidades:", len(entities.keys()))
    print("")
    print("Distribuição de utilizadores por entidade:")
    for entity in sorted(entities.keys()):
        print(f"* {entity} : {entities[entity]}")
    print("")
    print("Distribuição de utilizadores por nível:")
    for level in sorted(levels.keys()):
        print(f"* Nível {level} : {levels[level]}")

def json_20():
    list = []
    for i in range(20):
        broken_line = re.split(r'::', text[i])
        user = re.match(r'(\w+\s*(\w+\s*)*\b)', broken_line[0]).group()
        email = re.search(r'(\w+|\.|@|_|-)+', broken_line[1]).group()
        entity = re.search(r'ent_\w*', broken_line[2]).group()
        level = re.search(r'\d+\.? \d*', broken_line[3]).group()
        calls = re.search(r'\d+', broken_line[4]).group()
        list.append((user, email, entity, level, calls))

    file_name = input("\nDigite Nome Do Ficheiro de Output!\n>> ")
    fp = open(file_name, 'w')

```



```

fp.write("{\n\t\"registos\": [\n")
for i in range(len(list)):
    l = list[i]
    fp.write("\t\t{\n")
    fp.write(f"\t\t\t \"utilizador\": \"{l[0]}\",\n")
    fp.write(f"\t\t\t \"email\": \"{l[1]}\",\n")
    fp.write(f"\t\t\t \"entidade\": \"{l[2]}\",\n")
    fp.write(f"\t\t\t \"nivel de acesso\": \"{l[3]}\",\n")
    fp.write(f"\t\t\t \"número de chamadas ao backend\": \"{l[4]}\",\n")
    if i != 19:
        fp.write("\t\t},\n")
    else:
        fp.write("\t\t}\n")
fp.write("\t]\n}\n")
fp.close()
print("\nFicheiro gerado com sucesso!")

```

```

def menu():
    cls = lambda: print('\n' * 50)
    inputfromuser = ""
    options = []
    options.append("Listagem com o nome e a entidade do utilizador, ordenada alfabeticamente por nome")
    options.append("Lista ordenada alfabeticamente das entidades referenciadas, indicando, para cada entidade, o nome do utilizador")
    options.append("Distribuição de utilizadores por níveis de acesso.")
    options.append("Utilizadores, agrupados por entidade, ordenada primeiro pela entidade e depois pelo nome do utilizador")
    options.append("Mostrar alguns indicadores.")
    options.append("Imprimir os 20 primeiros registos num novo ficheiro de output mas em formato json")
    cls()
    while inputfromuser != '0':
        print("*** Selecione Opção ***\n")
        for i in range(len(options)):
            print(f"{i+1}. {options[i]}")
        print("0. Sair.")

        inputfromuser = input(">> ")
        while not inputfromuser.isdigit() or int(inputfromuser) > len(options) or int(inputfromuser) == 0:
            print("Opção Invalida!")
            inputfromuser = input(">> ")

        if inputfromuser == '0':
            continue
        elif inputfromuser == '1':
            name_entity_list()
        elif inputfromuser == '2':
            entity_num_elements_list()
        elif inputfromuser == '3':

```

```
        dist_users_level()
    elif inputfromuser == '4':
        name_entity_group()
    elif inputfromuser == '5':
        indicators()
    else:
        json_20()

    input("\nPressione Enter!\n>> ")
    cls()
```

```
fp = open('clav-users.txt', 'r')
text = fp.readlines()
fp.close()
menu()
```