

Processamento de Linguagens e Compiladores (3<sup>o</sup> 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

November 7, 2021

# Contents

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Problema Proposto</b>	<b>3</b>
2.1	Descrição . . . . .	3
2.2	Requisitos . . . . .	3
<b>3</b>	<b>Concepção/desenho da Resolução</b>	<b>4</b>
3.1	Organização . . . . .	4
3.2	Funcionalidades . . . . .	4
3.2.1	Nome dos utilizadores ordenados e respetiva entidade. . . . .	4
3.2.2	Entidades ordenadas e número de utilizadores registos em cada uma. . . . .	4
3.2.3	Distribuição dos utilizadores por nível de acesso. . . . .	4
3.2.4	Utilizadores agrupados por entidade, ordenados por nome e entidade. . . . .	4
3.2.5	Mostrar alguns indicadores. . . . .	4
3.2.6	Imprimir num ficheiro Json os 20 primeiros registos. . . . .	4
<b>4</b>	<b>Demonstração de Funcionamento</b>	<b>5</b>
4.1	Nome dos utilizadores ordenados e respetiva entidade. . . . .	5
4.2	Entidades ordenadas e número de utilizadores registos em cada uma. . . . .	5
4.3	Distribuição dos utilizadores por nível de acesso. . . . .	5
4.4	Utilizadores agrupados por entidade, ordenados por nome e entidade. . . . .	5
4.5	Mostrar alguns indicadores. . . . .	5
4.6	Imprimir num ficheiro Json os 20 primeiros registos. . . . .	5
<b>5</b>	<b>Conclusão</b>	<b>6</b>
<b>A</b>	<b>Código do Programa</b>	<b>7</b>

# Chapter 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.

## Chapter 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'.

## Chapter 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

**3.2.1 Nome dos utilizadores ordenados e respetiva entidade.**

**3.2.2 Entidades ordenadas e número de utilizadores registos em cada uma.**

**3.2.3 Distribuição dos utilizadores por nível de acesso.**

**3.2.4 Utilizadores agrupados por entidade, ordenados por nome e entidade.**

**3.2.5 Mostrar alguns indicadores.**

**3.2.6 Imprimir num ficheiro Json os 20 primeiros registos.**

## Chapter 4

# Demonstração de Funcionamento

- 4.1 Nome dos utilizadores ordenados e respetiva entidade.
- 4.2 Entidades ordenadas e número de utilizadores registos em cada uma.
- 4.3 Distribuição dos utilizadores por nível de acesso.
- 4.4 Utilizadores agrupados por entidade, ordenados por nome e entidade.
- 4.5 Mostrar alguns indicadores.
- 4.6 Imprimir num ficheiro Json os 20 primeiros registos.

## Chapter 5

# Conclusão

Síntese do Documento [?, ?].

Estado final do projecto; Análise crítica dos resultados [?].

Trabalho futuro.

# Appendix A

## Código do Programa

```
import re
import unicode

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

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()
        result.append((user, entity))

    result.sort(key=lambda x: unicode.unidecode(x[0].casefold()))
    print("\n*** Utilizadores - Entidades ***\n")
    for r in result:
        print(r[0], "-", 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*** Entidades - 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("")
    print("Número de Utilizadores:")
    print(len(users))
    print("")
    print("Número de Entidades:")
    print(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"* {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\":[")
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")
    options.append("Lista ordenada alfabeticamente das entidades referenciadas, indicando, para")
    options.append("Distribuição de utilizadores por níveis de acesso.")
    options.append("Utilizadores, agrupados por entidade, ordenada primeiro pela entidade e depois")
    options.append("Mostrar alguns indicadores.")
    options.append("Imprimir os 20 primeiros registos num novo ficheiro de output mas em formato")
    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()
```

```
menu()
```