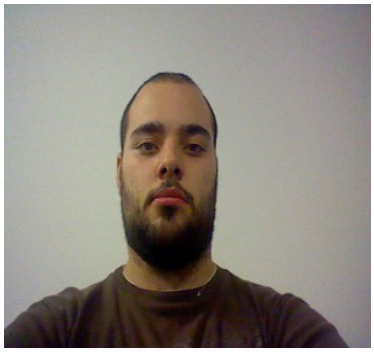
Instituto Politécnico de Beja
Análise de Colocações no Ensino Superior

Pedro Miguel Barbeitas Lemos Figueiredo
Nº 10326



Alexandre Miguel Vasco Leitão
Nº 10331

Data: 02/12/2013

1 Introdução

- ## *Contextualização do trabalho*

Este trabalho foi realizado para a disciplina de Linguagens de Programação de modo a avaliar os conhecimentos dos alunos em relação à linguagem Python. Foi pedido aos alunos que realizassem uma aplicação com interface gráfica que processasse e apresentasse estatísticas da primeira fase de candidaturas de alunos a cursos do ensino superior.

Este trabalho tem como objectivo desenvolver e testar os conhecimentos dos alunos na manipulação de dados, utilização de Python , percepção e utilização dos varios conceitos ideológicos por trás da linguagem(a sua simplicidade e a utilização de bibliotecas).

A entrega do trabalho foi feita através do sistema de controlo de versões “mercurial” sendo criadas para o efeito 3 diretorias, a diretoria /report onde se encontra o ficheiro do relatório em formato pdf, a diretoria /presentation onde se encontra o ficheiro relativo à apresentação do trabalho, e a diretoria /src onde se encontram todos os ficheiros relativos ao código e ficheiros utilizados pelo código.

Na realização deste trabalho foram usadas as bibliotecas sqlite3, xlre, csv, matplotlib e wx.

- ## *Descrição de objetivos realizados*

Este trabalho pedia que nos desenvolvissemos um programa que dado uma folha de excell ele fosse capaz de gerar um base de dados, ficheiros csv e graficos, todos eles referentes a informação da folha de calculo excell. Todos os objetivos pretendidos no trabalho foram executados, no entanto, na execução dos objetivos foram usadas as ferramentas mais básicas. Para a criação da base de dados usámos a biblioteca **sqlite3,** para a busca dos dados num ficheiro excel usámos a biblioteca xlrd, para a passagem das estatísticas para um ficheiro csv utilizámos a biblioteca **csv**, para a criação dos gráficos usámos a biblioteca **matplotlib** e para a criação da interface gráfica usámos primeiramente a aplicação **wxglade** e fomos alterando o código à medida que iamos avançando no trabalho.

- ## *Estrutura do relatório*

O relatório é contituido pela **teoria** em que são explicados os módulos, as bibliotecas e as aplicações utilizadas.

É constituido pela **Parte experimental**, onde na secção “Realização Experimental” se descrevem as opções de escolha da linguagem de programação, do ambiente de desenvolvimento, o sistema operativo e hardware utilizados, onde na secção “Sistema Experimental” é efetuada uma descrição detalhada da estrutura do código e onde na secção “Resultados Experimentais”é descrito o protocolo experimental usada para a obtenção dos resultados, sua apresentação e discussão.

É constituídos também pelas **conclusões** onde o trabalho é resumido e são discutidos aspetos a melhorar no mesmo, pela **Bibliografia** onde se encontram as informações devidamente organizadas de locais onde fomos buscar informação útil para a realização do trabalho e pelos **Anexos,** onde se encontra documentação importante do trabalho.

2 Teoria

- **Linguagem de programação:**
  - **Python** - O Pyhon é uma linguagem de programação de alto nível, dinâmica, suporta a orientação por objetos, o estilo imperativo e funcional. Esta linguagem usa um interpretador, ao inves de um compilador como c, e como tal ela executa as operações imediatamente sem precisar de compilar o codigo. A organização dos seus métodos e classes baseia-se na identação não havendo chavetas como no java. Apresenta uma biblioteca padrão exensa, sendo fácil adicionar novos módulos. No website oficial da linguagem python "www.python.org" é possível encontrar a documentação, tutoriais e guias para aprender a programar em python e muitas outras utilidades referentes à linguagem.

    A linguagem foi criada nos anos 80 por Guido van Rossum, e foi desenhada para ser uma linguagem confortável e de fácil leitura. Esta filosofia está presente em muitos pormenores na linguagem como por exemplo a presença do self em todas funções que pertencam a uma classe, a identação, a não necessidade de declarar as variáveis. Um bom resumo da filosofia é dada pelos cinco aforismos dados "PEP 20 (The Zen of Python)": Belo é melhor que feio, explicito é melhor que implicito, simples é melhor que complexo, complexo é melhor que complicado, leitura resulta. Python não contem todas as funcionalidades ditas desejadas, mas é incrivelmente extensivel, e pode ser embutida em applicações. A liguagem resultante é uma linguagem que contem um nucleo de facto pequeno mas que pode ser extendido indefenidamente por bibliotecas.

    O nome (Python) tem origem no grupo comico britanico Monty Python. Python foi desenhada para ser engraçada, facil e , principalmente, para ser algo completamente diferente.

- **Ambientes de desenvolvimento:**
  - **Geany –** O Geany é um editor de texto multiplataforma desenvolvido em GTK+ possuidor de funções básicas de ambiente de desenvolvimento integrado (IDE), tendo algum suporte em relação ao python.
  - **Idle v2.7–** O idle é um IDE especializado em Python possuidor de uma Shell onde o código pode ser corrido pelo programador.
  - **Wxglade –** É uma Graphical User Interface (GUI) que permite criar a interface gráfica de uma aplicação, através de widgets de bibliotecas wx.

- **Bibliotecas:**
  - **sqlite3 –** É um módulo que nos permite através de alguns códigos criar, alterar e apagar uma base de dados. Atrávés da sua biblioteca em Python podem ser guardados objetos conhecidos do módulo de forma ciar e mexer numa base de dados.
  - **Xlrd –** Permite que se interaja com e que se criem ficheiros em formato excel.
  - **csv –** Permite que se interaja com e se criem ficheiros em formato csv.
  - **Matplolib –** É uma blioteca de esboços em 2 dimensões, que permite a criação de figuras que ajudam na interpretação de valores , tais como gráficos, formas e imagens.
  - **Wx –** É um a biblioteca que permite a criação de interfaces gráficas, tendo os elementos mais comuns das interfaces gráficas métodos dentro desta biblioteca que nos permitem posicioná-los, da foma que nós quisermos dentro das próprias limitações da biblioteca.

3 Parte Experimental

- ***Realização Experimental***
  - **Linguagem de programação** – Foi utilizada a linguagem de programação Python por requisito do trabalho. Neste trabalho usámos a versão 2.7 do python por requisito do professor, havendo já disponível versões até ao 3.3.3 no entanto, as mesmas não são tão estáveis e tem sérios problemas de retrocompatibilidade e de suporte de unicode, pontos que são extensamente usados ao longo do trabalho, por isso escolhemos a versão 2.7, que se mostra apropriada para o desenvolvimento deste trabalho.
  - **Ambientes de desenvolvimento**
    - **Idle** - Foi utilizado de início por vir com a linguagem logo predifinida, e também voltou a ser utilizado quando havia problemas com a identação feita através do Geany.
    - **Geany –** Foi utilizado o Geany, visto que permite ter vários documentos com código, abertos ao mesmo tempo e organizados por tabs, tem um bom suporte Python, completa automaticamente o código indo buscar por exemplo nomes de variáveis e funções, apresenta no geral uma GUI muito mais elegante e confortavel do que a do idle e permite uma navegação pela pasta source de igual conforto.
    - **Wxglade –** O objetivo inicial seria fazer a aplicação em pyqt, no entanto, verificámos que com o tempo que era mais economico usar o wxGlade, que tem uma GUI que permite criar rapidamente uma interface funcional. É extremamente simples de utilizar e permite a rapida criação de uma interface capaz.
  - **Sistemas Operativos:**
    - **Linux Mint 14 (64-bits):** Foi utilizado pelo elemento Alexandre Leitão por ser o Sistema Operativo que já tinha instalado no seu computador e por ser um sistema operativo que funciona também bastante bem como estação de trabalho. Este tem todas as biblitecas necessarias para a realização do trabalho. É importante notar que as distribuições Mint tem muitas semelhanças com as distribuições uBuntu.
    - **Debian 7.1(32-bits):** Foi utilizado pelo elemento Pedro Figueiredo por ser o sistema operativo fornecido pelo professor através de uma máquina virtual, vindo já com tudo o que era necessário para a execução do trabalho instalado, permitindo ao elemento concentrar-se na programação da da aplicação. Permite também guardar a máquina virtual com todas as janelas necessárias abertas de modo a que quem esteja a trabalhar e tenha que sair e desligar o computador, depois regresse e tenha tudo preparado para continuar a trabalhar.
    - **Linux Mint 15 (64-bits):** Foi utilizado quando o elemento Pedro Figueiredo se deslocou para a casa de família, visto que só tinha acesso a um computador e tem uma grande preferência pelo gestor de janelas cinnamon que permite uma fluência bastante rápida.
  - **Hardware:**
    - **ASUS N53-S –** Laptop utilizado pelo elemento Pedro Figueiredo, foi através do programa Oracle VirtualBox Manager 4.2.12 que o elemento usou a máquina virtual cedida pelo professor.
    - **Toshiba Satelitte A660**- Laptop utilizado pelo Elemento Alexandre Leitão, contem em dual boot os SOs Windows 8 e Linux Mint 14 (o Windows 8 não foi usado neste trabalho)
    - **Lenovo –** Computador utilizado pelo elemento Alexandre Leitão para substituir o Toshiba quando este teve problemas técnicos. Foi efetuada uma mudança de disco rígido de forma a existirem menos constrangimentos na realização do trabalho.
    - **Packard Bell –** Computador que se encontrava na casa de família do elemento Pedro Figueiredo. Devido ao fato de o elemento já não ter acesso físico ao mesmo, não sabe as especificações técnicas nem o nome do modelo do computador. Foi apenas utilizado para desenrascar.

- ***Sistema Experimental***

  - **Código trabalho.py**

Código responsável por tudo o que não é gráfico na aplicação, este código cria a base de dados, faz a passagem dos dados da folha excel para a base de dados, passa as estatísticas para ficheiros csv e chama funções do **script.py** que tratam das estatísticas e desenham gráficos.

```
import xlrd
from xlrd import
open_workbook
```

Importação da biblioteca que permite ler os dados do ficheiro excel e do método da biblioteca responsável por essa ação.

```
import sqlite3
```

Importação da biblioteca que permite criar alterar e apagar bases de dados.

```
import script as sd
```

Importação do script criado por nós que contém métodos responsáveis por tratar das estatísticas e criação de gráficos.

```
import csv
```

Importação da biblioteca responsável por passar os dados da estatística para ficheiros csv

```
class Trabalho:
```

Classe responsável por todas as funções não gráficas da aplicação.

```
def __init__(self):

    pass
```

Método construtor da classe

```
def criar_base_de_dados(self, ficheiro):
```

Método que cria a base de dados, recebe uma string com nome da base de dados que fica guardada na variável ficheiro.

```
self.conexao = sqlite3.connect(ficheiro)
```

Coneta o apontador do sqlite3 à base de dados ficheiro e guarda-o na variável geral conexao

```
self.c = self.conexao.cursor()
```

Cria um cursor e guarda-o na variável geral c

```
self.conexao.text_factory = str
```

Altera a text_factory da base de dados de forma a evitar erros de formatação das strings

```
self.c.execute('drop table if exists resultados_cna')
```

Se já existir uma tabela na base de dados de nome `resultados_cna` esta é eliminada

```
self.c.execute('''create table if not exists resultados_cna
                  (cInstituicao int, cCurso int,
                  instituicao text, curso text, grau text,
                  vagasIniciais int, colocados int,
                  notaMaisBaixa double, vagasSobrantes int)''')
```
O cursor executa a query dentro da string com os elementos a serem os que estão dentro dos parêntises da string.

```
def ler_ficheiro_excel(self, ficheiro):
```
Método responsável pela leitura do ficheiro excel, guardando o nome do ficheiro excel na variável ficheiro

```
self.wb = open_workbook(ficheiro)
```
Guarda o ficheiro excel na variável geral wb

```
self.folha = self.wb.sheet_by_index(0)
```
Guarda a 1ª folha do ficheiro excel na variável geral folha

```
def passagem_de_dados(self, ficheiro_excel, ficheiro_base_de_dados):
```
Método responsável pela passagem dos dados do ficheiro excel para a base de dados.

```
self.ler_ficheiro_excel(ficheiro_excel)
```
Executa o método para o ficheiro excel pretendido

```
self.criar_base_de_dados(ficheiro_base_de_dados)
```
Executa o método para o ficheiro pretendido

```
for i in range(3, self.folha.nrows - 2):
        #query que envia uma linha excel para a base de dados
        self.c.execute("insert into resultados_cna
values(?,?,?,?,?,?,?,?,?)",
                       (self.folha.cell(i,0).value,
                        self.folha.cell(i,1).value,
                        self.folha.cell(i,2).value,
                        self.folha.cell(i,3).value,
                        self.folha.cell(i,4).value,
                        self.folha.cell(i,5).value,
                        self.folha.cell(i,6).value,
                        self.folha.cell(i,7).value,
                        self.folha.cell(i,8).value))
        pass
```
Guarda os elementos da folha excel de forma organizada na base de dados

```
self.conexao.commit()
```
Efetua as alterações na base de dados

```
def estatistica1(self, ficheiro_base_de_dados):
```

Método responsável pela criação da tabela de estatísticas relacionada com os institutos

```
self.c.execute("""select * from resultados_cna""")
```

Seleciona todos os dados da tabela resultados_cna da base de dados.

```
sd.tabela_escolas(self.c.fetchall())
```

Execução do método do ficheiro script com os dados organizados da base de dados de forma a que possam ser lidos.

```
sd.tabela_escolas(self.c.fetchall())
```

Método responsável pela criação da tabela de estatísticas relacionada com os distritos

```
self.c.execute('''select * from
escolas''')
```

Seleciona todos os dados da tabela escolas da base de dados

```
sd.tabela_distritos(self.c.fetchall())
```

Execução do método do ficheiro script com os dados organizados da base de dados de forma a que possam ser lidos.

```
def estatisticasCSV(self, ficheiro_base_de_dados):
```

Método responsável pela passagem das estatísticas para os ficheiros csv, recebe uma string com o nome do ficheiro da base de dados, seleciona os dados das estatísticas e cria um ficheiro com as estatísticas das instituições e outro com as estatísticas dos distritos.

```
def criacaoGraficoEntradasNorte(self, ficheiro_base_de_dados):
```

Método responsável pela criação do gráfico com a estatística dos alunos que entraram em escolas de distritos da região norte de Portugal. Existem vários métodos parecidos mas relecionados com outro tipo de estatísticas.

- ## Código script.py

Neste ficheiro está contido o código que é responsável por tratar das estatísticas e guardá-las em novas tabelas da base de dados. Aqui é também importada a biblioteca **matplolib** que será utilizada para a criação dos gráficos relativos às estatísticas.

```
def tabela_distritos(lista):
```

Método responsável pela criação da tabela distritos com as estatísticas correspondentes aos distritos.

```
for distrito in distritos:
        for row in lista:
            if distrito in row[0]:
                if distrito != 'Braga':
                    entradas += row[1]
                    vagas += row[2]
                pass
            if distrito == 'Faro':
                if 'Algarve' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
```

```
            pass
        if distrito == 'Vila Real':
            if      'Beira Interior' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            if      'Alto Douro' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            pass
        if distrito == 'Braga':
            if      'Minho' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            if 'cávado' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            pass
        if distrito == 'Santarém':
            if 'Tomar' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            pass
        if distrito == 'Lisboa':
            if 'ISCTE' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            if 'Infante D. Henrique' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            if 'Hotelaria e Turismo' in row[0]:
                    entradas += row[1]
                    vagas  += row[2]
                    pass
            pass
```

Nesta parte do código são escolhidas os distritos a que as escolas pertencem.se os distritos contiverem as palavras acima referidas nos if's é porque pertencem aos distritos correspondentes. Em relação ao if distrito != 'Braga', serve apenas para excepcionalizar o distrito de Bragança, visto que a palavra Braga pertence à palavra Bragança.

def tabela_escolas(lista):

Este método cria a tabela das escolas na base de dados e adiciona-lhe os dados com as estatísticas das instituições.

def graficoInst(mother, lista)

Este método não chega a ser utilizado na aplicação no entanto cria um gráfico com as estatísticas de entrada de alunos por instituição, servindo portanto de exemplo para todos os gráficos seguintes que são executados na aplicação.

def detectUpper(x):

Método responsável por criar uma string apenas com as letras maiúsculas da string x. Útil para os gráficos das instituições devido ao enorme tamanho do nome das instituições não caber nas legendas dos gráficos correspondentes às estatísticas das mesmas.

- ### Código interface.py

Neste ficheiro é criada a interface gráfica e os eventos que o clique dos botões despoleta. Isto é criado graças ao import do wx que permite a criação de todos os gráficos. Criamos as métodos que executam os métodos do ficheiro trabalho.py que também é importado com a variável t.

```
class MyFrame(wx.Frame):
```

Esta classe é criada pelo wxglade e contém todos os elementos da interface gráfica

```
def __init__(self, *args, **kwds):
```

Método construtor da interface gráfica, aqui são adicionados todos os widgets (botões, separadores...)visíveis pelo utilizador

```
def criar_db(self, event):
        """
            Event responsible for creating the DB
        """
        self.tr.passagem_de_dados('cna131fresultados.xls',
'trabalho')
        event.Skip()
```

Um dos métodos que executa um método do código do ficheiro trabalho.py. Quando este evento acontece é criada a base de dados e feita a passagem dos dados do ficheiro excel para lá.

```
def __do_layout(self):
```

Neste método são organizados todos os widgets. É aqui que está o código que posiciona cada widget no seu local.

- ## *Resultados Experimentais*
  - O protocolo experimental utilizado foi o teste de cada método à medida que era criado um método novo, exceto em casos onde o código era tão simples ou tão parecido com métodos anteriores que não se verificava tal exigência.
  - Tivemos alguns problemas, muitas vezes os resultados não foram os pretendidos, mas em termos de resultados finais não correspondem às expetativas o fato de o código `if 'conexao in globals():` não funcionar da maneira pretendida, visto que nos permitiria conetar a uma base de dados já existente. Também a labels dos gráficos com os nomes dos distritos não corresponderam ao esperado não tendo sido possível retirar os parêntises e reconhecer os carateres especiais no resultado final.

- # Conclusões

O código criado respeita os requisitos pretendidos, apesar do codigo não ser um exemplo de organização, podem-lhe ser feitas alterações para que tenha uma estrutura mais amigável. As labels dos gráficos com as estatísticas dos distritos não correspondem às expetativas, podendo serem efetuadas alterações no código que retirem os carateres a mais e corrijam a formatação.

Podem ser feitas alterações no código de forma a que seja possível criar as estatísticas, fazer a passagem destas para o ficheiro ou criar os gráficos das estatísticas acedendo a uma base de dados já existente. Não implementámos isto no código porque deixámos essa função para o fim e não conseguimos confirmar criar um código que confirmasse a existência de uma variável global. Podiamos criar várias variáveis no código que abrissem e fechassem a base de dados vezes sem conta, no entanto visto que o código perderia qualidade, decidimos não o fazer e explicá-lo aqui, visto que seria de simples implementação.

O objectivo deste trabalho, como foi dito na introdução, é a nossa aprendizagem não da linguagem de programação Python mas antes familiriazar-nos com outra linguagem de programação, para que nos possamos comprender a logica inerente das linguagens de porgramação, e dar-nos a entender que mesmo que o paradigma principal seja diferente, elas funcionam segundo um molde de ideias fixas.

Com este trabalho aprendemos a extender o nosso conhecimento sobre funções e operações, extendendo e corrigindo ideias imcompletas presentes no nosso metodo de programação.

- # Bibliografia

JH, DD, EF, MD and the matplotlib development team (10, Out. 2013), api example code: barchart_demo.py, disponível: http://matplotlib.org/examples/api/barchart_demo.html

Documentação da biblioteca padrão de Python 2.7.6, disponível em: http://docs.python.org/2/library/sqlite3.html

Steve Tjoa (1, Fev 2010), Resposta na comunidade StackOverflow, disponível em: http://stackoverflow.com/questions/2177504/individually-labeled-bars-for-bar-graphs-in-matplotlib-python

Cristian Ciupitu(22, Set 2010), Resposta na comunidade StackOberflow, disponível em: http://stackoverflow.com/questions/3710263/how-do-i-create-a-csv-file-from-database-in-python

Zag (26, Out. 2010), Resposta na comunidade StackOverflow disponível em: http://stackoverflow.com/questions/3425320/sqlite3-programmingerror-you-must-not-use-8-bit-bytestrings-unless-you-use-a-te

- # Anexos

  - ## Interface

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# autor: Pedro Figueiredo e Alexandre Leitão
# data: 30 de Setembro de 2013
# trabalho de Linguagens de Programação
# generated by wxGlade 0.6.5 on Thu Nov 28 19:38:25 2013

import wx
import trabalho as t
# begin wxGlade: extracode
# end wxGlade
"""
This is the interface class, once you start the program by initializing
this class
"""

class MyFrame(wx.Frame):
    """
        Starts the the program interface
        @param wx.Frame The window of the program
    """

    def __init__(self, *args, **kwds):
        """
            Constructor method of the wx interface
        """
        # begin wxGlade: MyFrame.__init__
        self.tr =t.Trabalho()
        kwds["style"] = wx.DEFAULT_FRAME_STYLE
        wx.Frame.__init__(self, *args, **kwds)

        self.notebook_1 = wx.Notebook(self, -1, style=0)# Creates the section where all the panes will be deposited

        self.notebook_1_pane_db = wx.Panel(self.notebook_1, -1)# Creates the first pane which contains 3 buttons responsible for creating the DB, CSVs and imports data from xls
        self.label_1 = wx.StaticText(self.notebook_1_pane_db, -1, u"Clicando no botão seguinte pode fazer a passagem dos dados da folha excel para uma base de dados")

        self.button_criar_db = wx.Button(self.notebook_1_pane_db, -1, "Criar Base de dados")
        self.Bind(wx.EVT_BUTTON, self.criar_db, self.button_criar_db)

        self.label_2 = wx.StaticText(self.notebook_1_pane_db, -1, u"Clicando no botão seguinte cria duas tabelas. Uma com as estatísticas das escolas e outra com as estatísticas dos distritos.")

        self.button_estatisticas = wx.Button(self.notebook_1_pane_db, -1, u"Criação de tabelas para a estatística")
        self.Bind(wx.EVT_BUTTON, self.criar_estatisticas, self.button_estatisticas)
        self.label_csv = wx.StaticText(self.notebook_1_pane_db, -1, u"Clicando no botão seguinte passa os dados da estatística para um ficheiro CSV")
        self.button_csv = wx.Button(self.notebook_1_pane_db, -1, u"Criar ficheiro CSV")
        self.Bind(wx.EVT_BUTTON, self.passagem_csv, self.button_csv)

        self.notebook_1_pane_graficos_distritos = wx.Panel(self.notebook_1, -1)
        self.label_3 = wx.StaticText(self.notebook_1_pane_graficos_distritos, -1, u"Criação das estatísticas dos institutos do Norte")
```

```
        self.button_entradas_norte = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "entradas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_entradas_norte, self.button_entradas_norte)

        self.button_vagas_norte = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "vagas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_vagas_norte, self.button_vagas_norte)

        self.button_permilagem_norte = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "permilagem")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_permilagem_norte, self.button_permilagem_norte)

        self.static_line_1 = wx.StaticLine(self.notebook_1_pane_graficos_distritos, -1, style=wx.LI_VERTICAL)
        self.label_4 = wx.StaticText(self.notebook_1_pane_graficos_distritos, -1, u"Criação das estatísticas dos
institutos do Centro")

        self.button_entradas_centro = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "entradas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_entradas_centro, self.button_entradas_centro)

        self.button_vagas_centro = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "vagas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_vagas_centro, self.button_vagas_centro)

        self.button_permilagem_centro = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "permilagem")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_permilagem_centro, self.button_permilagem_centro)

        self.static_line_2 = wx.StaticLine(self.notebook_1_pane_graficos_distritos, -1, style=wx.LI_VERTICAL)
        self.label_5 = wx.StaticText(self.notebook_1_pane_graficos_distritos, -1, u"Criação das estatísticas do Sul")

        self.button_entradas_sul = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "entradas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_entradas_sul, self.button_entradas_sul)

        self.button_vagas_sul = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "vagas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_vagas_sul, self.button_vagas_sul)

        self.button_permilagem_sul = wx.Button(self.notebook_1_pane_graficos_distritos, -1, "permilagem")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_permilagem_Sul, self.button_permilagem_sul)

        self.notebook_1_pane_graficos_intituicoes = wx.Panel(self.notebook_1, -1)
        self.label_6 = wx.StaticText(self.notebook_1_pane_graficos_intituicoes, -1, u"Criação do gráfico das
escolas")

        self.button_entradas_escolas = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "entradas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_entradas_escolas, self.button_entradas_escolas)

        self.button_vagas_escolas = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "vagas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_vagas_escolas, self.button_vagas_escolas)

        self.button_percentagem_escolas = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "percentagem")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_percentagem_escolas, self.button_percentagem_escolas)

        self.static_line_3 = wx.StaticLine(self.notebook_1_pane_graficos_intituicoes, -1, style=wx.LI_VERTICAL)
        self.label_7 = wx.StaticText(self.notebook_1_pane_graficos_intituicoes, -1, u"Criação do gráfico dos
politécnicos")

        self.button_entradas_politecnicos = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "entradas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_entradas_institutos, self.button_entradas_politecnicos)

        self.button_vagas_politecnicos = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "vagas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_vagas_institutos, self.button_vagas_politecnicos)

        self.button_percentagem_politecnicos = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1,
"percentagem")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_percentagem_institutos,
self.button_percentagem_politecnicos)
```

```
        self.static_line_4 = wx.StaticLine(self.notebook_1_pane_graficos_intituicoes, -1, style=wx.LI_VERTICAL)
        self.label_8 = wx.StaticText(self.notebook_1_pane_graficos_intituicoes, -1, u"criação do gráfico das
universidades")

        self.button_entradas_universidades = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "entradas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_entradas_universidades, self.button_entradas_universidades)

        self.button_vagas_universidades = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1, "vagas")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_vagas_universidades, self.button_vagas_universidades)

        self.button_percentagem_universidades = wx.Button(self.notebook_1_pane_graficos_intituicoes, -1,
"percentagem")
        self.Bind(wx.EVT_BUTTON, self.criar_grafico_percentagem_universidades,
self.button_percentagem_universidades)

        self.__set_properties()
        self.__do_layout()

        # end wxGlade
        pass
#início de funções eventos dos botões.....................................................................
    def criar_db(self, event):
        """
            Event responsible for creating the DB
        """
        self.tr.passagem_de_dados('cna131fresultados.xls', 'trabalho')
        event.Skip()
        pass

    def criar_estatisticas(self, event):
        """
            Event responsible for creating the estatistics in which the program is going to base himself
        """
        self.tr.estatistica1('trabalho')
        self.tr.estatistica2('trabalho')
        event.Skip()
        pass

    def criar_grafico_entradas_norte(self, event):
        """
            Event Responsible for creating the graph which shows the number of entries in the north
        """
        self.tr.criacaoGraficoEntradasNorte('trabalho')
        event.Skip()
        pass

    def criar_grafico_entradas_centro(self, event):
        """
           Event Responsible for creating the graph which shows the number of entries in the center
        """
        self.tr.criacaoGraficoEntradasCentro('trabalho')
        event.Skip()
        pass

    def criar_grafico_entradas_sul(self, event):
        """
           Event Responsible for creating the graph which shows the number of entries in the south
        """
        self.tr.criacaoGraficoEntradasSul('trabalho')
        event.Skip()
        pass

    def criar_grafico_vagas_norte(self, event):
```

```
        """
            Event Responsible for creating the graph which shows the number of vacancies in the north
        """
        self.tr.criacaoGraficoVagasNorte('trabalho')
        event.Skip()
        pass

    def criar_grafico_vagas_centro(self, event):
        """
            Event Responsible for creating the graph which shows the number of vacancies in the center
        """
        self.tr.criacaoGraficoVagasCentro('trabalho')
        event.Skip()
        pass

    def criar_grafico_vagas_sul(self, event):
        """
            Event Responsible for creating the graph which shows the number of vacancies in the south
        """
        self.tr.criacaoGraficoVagasSul('trabalho')
        event.Skip()
        pass

    def criar_grafico_permilagem_norte(self,event):
        """
            Event Responsible for creating the graph which shows per thousand of vacancies ocuppied in the north
        """
        self.tr.criacaoGraficoPermilagemNorte('trabalho')
        event.Skip()
        pass

    def criar_grafico_permilagem_centro(self,event):
        """
            Event Responsible for creating the graph which shows per thousand of vacancies ocuppied in the center
        """
        self.tr.criacaoGraficoPermilagemCentro('trabalho')
        event.Skip()
        pass

    def criar_grafico_permilagem_Sul(self,event):
        """
            Event Responsible for creating the graph which shows per thousand of vacancies ocuppied in the south
        """
        self.tr.criacaoGraficoPermilagemSul('trabalho')
        event.Skip()
        pass

    def criar_grafico_entradas_escolas(self,event):
        """
             Event resposible for creating the graph which links the number of entries in each school
        """
        self.tr.criacaoGraficoEscolasEntradas('trabalho')
        event.Skip()
        pass

    def criar_grafico_vagas_escolas(self,event):
        """
             Event resposible for creating the graph which links the number of vacancies in each school
        """
        self.tr.criacaoGraficoEscolasVagas('trabalho')
        event.Skip()
        pass
```

```
def criar_grafico_percentagem_escolas(self,event):
    """
        Event Responsible for creating the graph which shows percentage of vacancies ocuppied in each school
    """
    self.tr.criacaoGraficoEscolasPercentagem('trabalho')
    event.Skip()
    pass

def criar_grafico_entradas_institutos(self,event):
    """
        Event resposible for creating the graph which links the number of entries in each institute
    """
    self.tr.criacaoGraficoInstitutosEntradas('trabalho')
    event.Skip()
    pass

def criar_grafico_vagas_institutos(self,event):
    """
        Event resposible for creating the graph which links the number of vacancies in each institute
    """
    self.tr.criacaoGraficoInstitutosVagas('trabalho')
    event.Skip()
    pass

def criar_grafico_percentagem_institutos(self,event):
    """
        Event resposible for creating the graph which shows percentage of vacancies ocuppied in each institute
    """
    self.tr.criacaoGraficoInstitutosPercentagem('trabalho')
    event.Skip()
    pass

def criar_grafico_entradas_universidades(self,event):
    """
        Event resposible for creating the graph which links the number of entries in each university
    """
    self.tr.criacaoGraficoUniversidadesEntradas('trabalho')
    event.Skip()
    pass

def criar_grafico_vagas_universidades(self,event):
    """
        Event resposible for creating the graph which links the number of vacancies in each university
    """
    self.tr.criacaoGraficoUniversidadesVagas('trabalho')
    event.Skip()
    pass

def criar_grafico_percentagem_universidades(self,event):
    """
        Event resposible for creating the graph which shows percentage of vacancies ocuppied in each university
    """
    self.tr.criacaoGraficoUniversidadesPercentagem('trabalho')
    event.Skip()
    pass

def passagem_csv(self,event):
    """
        Event responsible for converting the database tables into csv files
    """
    self.tr.estatisticasCSV('trabalho')
    pass
```

```
#fim de funções eventos dos botões
    def __set_properties(self):
        """
            This method sets important characteristics of the program interface frame
        """
        # begin wxGlade: MyFrame.__set_properties
        self.SetTitle(u"Trabalho prático")
        self.notebook_1_pane_graficos_distritos.SetMinSize((1302, 121))
        pass
        # end wxGlade

    def __do_layout(self):
        """
            Sets the layout of the various elements within the frame
        """
        # begin wxGlade: MyFrame.__do_layout
        sizer_1 = wx.BoxSizer(wx.VERTICAL)
        sizer_18 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_21 = wx.BoxSizer(wx.VERTICAL)
        sizer_20 = wx.BoxSizer(wx.VERTICAL)
        sizer_19 = wx.BoxSizer(wx.VERTICAL)
        sizer_7 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_8 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_10 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_11 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_13 = wx.BoxSizer(wx.VERTICAL)
        sizer_12 = wx.BoxSizer(wx.VERTICAL)
        sizer_9 = wx.BoxSizer(wx.VERTICAL)
        sizer_2 = wx.BoxSizer(wx.VERTICAL)
        sizer_6 = wx.BoxSizer(wx.VERTICAL)
        sizer_2.Add(self.label_1, 0, 0, 0)
        sizer_6.Add(self.button_criar_db, 0, 0, 0)
        sizer_6.Add(self.label_2, 0, 0, 0)
        sizer_6.Add(self.button_estatisticas, 0, 0, 0)
        sizer_6.Add(self.label_csv)
        sizer_6.Add(self.button_csv)
        sizer_2.Add(sizer_6, 1, wx.EXPAND, 0)
        self.notebook_1_pane_db.SetSizer(sizer_2)
        sizer_9.Add(self.label_3, 0, 0, 0)
        sizer_9.Add(self.button_entradas_norte, 0, 0, 0)
        sizer_9.Add(self.button_vagas_norte, 0, 0, 0)
        sizer_9.Add(self.button_permilagem_norte, 0, 0, 0)
        sizer_8.Add(sizer_9, 1, wx.EXPAND, 0)
        sizer_10.Add(self.static_line_1, 0, wx.EXPAND, 0)
        sizer_12.Add(self.label_4, 0, 0, 0)
        sizer_12.Add(self.button_entradas_centro, 0, 0, 0)
        sizer_12.Add(self.button_vagas_centro, 0, 0, 0)
        sizer_12.Add(self.button_permilagem_centro, 0, 0, 0)
        sizer_11.Add(sizer_12, 1, wx.EXPAND, 0)
        sizer_11.Add(self.static_line_2, 0, wx.EXPAND, 0)
        sizer_13.Add(self.label_5, 0, 0, 0)
        sizer_13.Add(self.button_entradas_sul, 0, 0, 0)
        sizer_13.Add(self.button_vagas_sul, 0, 0, 0)
        sizer_13.Add(self.button_permilagem_sul, 0, 0, 0)
        sizer_11.Add(sizer_13, 1, wx.EXPAND, 0)
        sizer_10.Add(sizer_11, 1, wx.EXPAND, 0)
        sizer_8.Add(sizer_10, 1, wx.EXPAND, 0)
        sizer_7.Add(sizer_8, 1, wx.EXPAND, 0)
        self.notebook_1_pane_graficos_distritos.SetSizer(sizer_7)
        sizer_19.Add(self.label_6, 0, 0, 0)
        sizer_19.Add(self.button_entradas_escolas, 0, 0, 0)
        sizer_19.Add(self.button_vagas_escolas, 0, 0, 0)
        sizer_19.Add(self.button_percentagem_escolas, 0, 0, 0)
```

```
        sizer_18.Add(sizer_19, 1, wx.EXPAND, 0)
        sizer_18.Add(self.static_line_3, 0, wx.EXPAND, 0)
        sizer_20.Add(self.label_7, 0, 0, 0)
        sizer_20.Add(self.button_entradas_politecnicos, 0, 0, 0)
        sizer_20.Add(self.button_vagas_politecnicos, 0, 0, 0)
        sizer_20.Add(self.button_percentagem_politecnicos, 0, 0, 0)
        sizer_18.Add(sizer_20, 1, wx.EXPAND, 0)
        sizer_18.Add(self.static_line_4, 0, wx.EXPAND, 0)
        sizer_21.Add(self.label_8, 0, 0, 0)
        sizer_21.Add(self.button_entradas_universidades, 0, 0, 0)
        sizer_21.Add(self.button_vagas_universidades, 0, 0, 0)
        sizer_21.Add(self.button_percentagem_universidades, 0, 0, 0)
        sizer_18.Add(sizer_21, 1, wx.EXPAND, 0)
        self.notebook_1_pane_graficos_intituicoes.SetSizer(sizer_18)
        self.notebook_1.AddPage(self.notebook_1_pane_db, "base de dados")
        self.notebook_1.AddPage(self.notebook_1_pane_graficos_distritos, u"gráficos dos distritos")
        self.notebook_1.AddPage(self.notebook_1_pane_graficos_intituicoes, u"gráficos das instituições")
        sizer_1.Add(self.notebook_1, 1, wx.EXPAND, 0)
        self.SetSizer(sizer_1)
        sizer_1.Fit(self)
        self.Layout()
        # end wxGlade
        pass

# end of class MyFrame
if __name__ == "__main__":
    app = wx.PySimpleApp(0)
    wx.InitAllImageHandlers()
    frame_1 = MyFrame(None, -1, "")
    app.SetTopWindow(frame_1)
    frame_1.Show()
    app.MainLoop()
    pass
```

- ## Trabalho.py

```
# -*- coding: utf-8 -*-
# autor: Pedro Figueiredo e Alexandre Leitão
# data: 30 de Setembro de 2013
# trabalho de Linguagens de Programação

import xlrd
from xlrd import open_workbook
import sqlite3
import script as sd
import csv

"""
Class which makes all the the CVS, DataBase and excell operations and
the
nedded queries for graphics
"""

class Trabalho:

    def __init__(self):

        pass
```

```
    def criar_base_de_dados(self, ficheiro):
        """
            Creates the database
            @param ficheiro gives the name of the database
        """
        #criação da base de dados com o ficheiro de nome basededados.db e
        #guardá-la na variável conexao
        self.conexao = sqlite3.connect(ficheiro)
        #criação da instância cursor que permite mandar queries para a base
        #de dados
        self.c = self.conexao.cursor()
        self.conexao.text_factory = str
        self.c.execute('drop table if exists resultados_cna')
        #query para criação da tabela na base de dados
        self.c.execute('''create table if not exists resultados_cna
                    (cInstituicao int, cCurso int,
                    instituicao text, curso text, grau text,
                    vagasIniciais int, colocados int,
                    notaMaisBaixa double, vagasSobrantes int)''')


    #leitura de dados de um ficheiro excel
    def ler_ficheiro_excel(self, ficheiro):
        """
            Reads one given excell file and stores the desired sheet
            @param ficheiro the name of the excell file
        """
        #guarda o ficheiro de excel na variável wb
        self.wb = open_workbook(ficheiro)
        #guarda a folha pretendida na variavel folha
        self.folha = self.wb.sheet_by_index(0)
        pass

    def passagem_de_dados(self, ficheiro_excel, ficheiro_base_de_dados):
        """
            Passes the data from the excell file to the the database
            @param ficheiro_excel name of the excell file
            @param ficheiro_base_de_dados name of the database
        """
        self.ler_ficheiro_excel(ficheiro_excel)
        self.criar_base_de_dados(ficheiro_base_de_dados)
        pass

        #Selecção da informação e introducção da mesma na base de dados
        for i in range(3, self.folha.nrows - 2):
            #query que envia uma linha excel para a base de dados
            self.c.execute("insert into resultados_cna
values(?,?,?,?,?,?,?,?,?)",
                        (self.folha.cell(i,0).value,
                         self.folha.cell(i,1).value,
                         self.folha.cell(i,2).value,
                         self.folha.cell(i,3).value,
                         self.folha.cell(i,4).value,
                         self.folha.cell(i,5).value,
                         self.folha.cell(i,6).value,
                         self.folha.cell(i,7).value,
                         self.folha.cell(i,8).value))
            pass
        #Executa na base de dados os queries do cursor
        self.conexao.commit()
        pass

    def estatistica1(self, ficheiro_base_de_dados):
```

```
        """
            Creates the query which will give the data nedded for the creation
of
            schools(escolas) table
            @param ficheiro_base_de_dados name of the database
            @return return the
        """
        self.c.execute("""select * from resultados_cna""")
        sd.tabela_escolas(self.c.fetchall())
        pass

    def estatistica2(self, ficheiro_base_de_dados):
        """
            Creates the query which will give the data nedded for the creation
of
            districts(distritos) table from the school table
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from escolas''')
        sd.tabela_distritos(self.c.fetchall())
        pass

    def estatisticasCSV(self, ficheiro_base_de_dados):
        """
            Creates the csv files which will hold the estatistics for schools
            and districts
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from escolas''')
        csvWriter = csv.writer(open('institutos.csv','wt'))
        csvWriter.writerows(self.c)
        del csvWriter
        self.c.execute('''select * from distritos''')
        csvWriter = csv.writer(open('distritos.csv','wt'))
        csvWriter.writerows(self.c)
        del csvWriter
        pass

    def criacaoGraficoEntradasNorte(self, ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of entries in the north
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDEN(self, self.c.fetchall())
        pass

    def criacaoGraficoEntradasCentro(self, ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of entries in the center
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDEC(self, self.c.fetchall())
        pass

    def criacaoGraficoEntradasSul(self, ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of entries in the south
            @param ficheiro_base_de_dados name of the database
```

```
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDES(self, self.c.fetchall())
        pass

    def criacaoGraficoVagasNorte(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of vacancies in the
north
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDVN(self, self.c.fetchall())
        pass

    def criacaoGraficoVagasCentro(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of vacancies in the
center
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDVC(self, self.c.fetchall())
        pass

    def criacaoGraficoVagasSul(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of vacancies in the
south
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDVS(self, self.c.fetchall())
        pass

    def criacaoGraficoPermilagemNorte(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the per thousand of ocuppied
            vacancies in the north
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDPN(self, self.c.fetchall())
        pass

    def criacaoGraficoPermilagemCentro(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the per thousand of ocuppied
            vacancies in the center
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('''select * from distritos''')
        sd.graficoDPC(self, self.c.fetchall())
        pass

    def criacaoGraficoPermilagemSul(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
```

```
            to make the graph which gives the per thousand of ocuppied
            vacancies in the south
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('"select * from distritos"')
        sd.graficoDPS(self, self.c.fetchall())
        pass

    def criacaoGraficoEscolasEntradas(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of entries in schools
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('"select*from escolas"')
        sd.graficoEscolasEntradas(self, self.c.fetchall())
        pass

    def criacaoGraficoEscolasVagas(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of vacancies in schools
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('"select*from escolas"')
        sd.graficoEscolasVagas(self, self.c.fetchall())
        pass

    def criacaoGraficoEscolasPercentagem(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the per thousand of
            ocupied vacancies in schools
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('"select*from escolas"')
        sd.graficoEscolasPercentagem(self, self.c.fetchall())
        pass

    def criacaoGraficoInstitutosEntradas(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of entries in institutes
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('"select*from escolas"')
        sd.graficoInstitutosEntradas(self, self.c.fetchall())
        pass

    def criacaoGraficoInstitutosVagas(self,ficheiro_base_de_dados):
        """
            Creates the query which will give the nedded information
            to make the graph which gives the number of vacancies in
institutes
            @param ficheiro_base_de_dados name of the database
        """
        self.c.execute('"select*from escolas"')
        sd.graficoInstitutosVagas(self, self.c.fetchall())
        pass

    def
criacaoGraficoInstitutosPercentagem(self,ficheiro_base_de_dados):
        """
```

```
        Creates the query which will give the nedded information
        to make the graph which gives the per thousand of
        ocupied vacancies in institutes
        @param ficheiro_base_de_dados name of the database
    """
    self.c.execute('"select*from escolas"')
    sd.graficoInstitutosPercentagem(self, self.c.fetchall())
    pass

  def
criacaoGraficoUniversidadesEntradas(self,ficheiro_base_de_dados):
    """
        Creates the query which will give the nedded information
        to make the graph which gives the number of entries in
        universities
        @param ficheiro_base_de_dados name of the database
    """
    self.c.execute('"select*from escolas"')
    sd.graficoUniversidadesEntradas(self, self.c.fetchall())
    pass

  def criacaoGraficoUniversidadesVagas(self,ficheiro_base_de_dados):
    """
        Creates the query which will give the nedded information
        to make the graph which gives the number of vacancies in
        universities
        @param ficheiro_base_de_dados name of the database
    """
    self.c.execute('"select*from escolas"')
    sd.graficoUniversidadesVagas(self, self.c.fetchall())
    pass

  def
criacaoGraficoUniversidadesPercentagem(self,ficheiro_base_de_dados):
    """
        Creates the query which will give the nedded information
        to make the graph which gives the per thousand of
        ocupied vacancies in univesities
        @param ficheiro_base_de_dados name of the database
    """
    self.c.execute('"select*from escolas"')
    sd.graficoUniversidadesPercentagem(self, self.c.fetchall())
    pass
```

- ## Script.py

```
# -*- coding: utf-8 -*-
# autor: Pedro Figueiredo e Alexandre Leitão
# data: 02 de Dezembro de 2013
# trabalho de Linguagens de Programação
import sqlite3
import matplotlib.pyplot as graf
import numpy as np

"""
    This script is the logic segment of this program, in a MVC
    comparison this would be the section most similar to the model
"""
def tabela_distritos(lista):
    """
        This function creates the district table on the database and
```

```
        inserts the values into it
        @param lista the list with districts on the resultados_rna table
"""
mapa_distritos = { 'Lisboa' : 'Lisboa' }
conn = sqlite3.connect('trabalho')
conn.text_factory = str
cursor = conn.cursor()
cursor.execute("drop table if exists distritos")
cursor.execute('''create table distritos
            (distrito, entradas, vagas, permilagem)
            ''')
conn.commit()
distritos = ['Aveiro','Beja','Braga','Bragança',
'Castelo Branco', 'Coimbra', 'Évora','Faro','Guarda','Leiria',
'Lisboa','Portalegre','Porto','Santarém','Setúbal',
'Viana do Castelo','Vila Real', 'Viseu']
escola = ""
entradas = 0
vagas = 0
data = []
for distrito in distritos:
    for row in lista:
        if distrito in row[0]:
            if distrito != 'Braga':
                entradas += row[1]
                vagas += row[2]
            pass
        if distrito == 'Faro':
            if 'Algarve' in row[0]:
                entradas += row[1]
                vagas  += row[2]
                pass
            pass
        if distrito == 'Vila Real':
            if     'Beira Interior' in row[0]:
                entradas += row[1]
                vagas  += row[2]
                pass
            if     'Alto Douro' in row[0]:
                entradas += row[1]
                vagas  += row[2]
                pass
            pass
        if distrito == 'Braga':
            if     'Minho' in row[0]:
                entradas += row[1]
                vagas  += row[2]
                pass
            if 'cávado' in row[0]:
                entradas += row[1]
                vagas  += row[2]
                pass
            pass
        if distrito == 'Santarém':
            if 'Tomar' in row[0]:
                entradas += row[1]
                vagas  += row[2]
                pass
            pass
        if distrito == 'Lisboa':
            if 'ISCTE' in row[0]:
                entradas += row[1]
                vagas  += row[2]
```

```
                        pass
                    if 'Infante D. Henrique' in row[0]:
                        entradas += row[1]
                        vagas  += row[2]
                        pass
                    if 'Hotelaria e Turismo' in row[0]:
                        entradas += row[1]
                        vagas  += row[2]
                        pass
                    pass
            data.append([distrito,entradas,vagas, (entradas * 1000.0/
(entradas+vagas))])
            entradas = 0
            vagas = 0
            pass
        for x in data:
            cursor.execute('insert into distritos values(?,?,?,?)',(x[0],x[1],x[2], x[3]))
            pass
        conn.commit()



def tabela_escolas(lista):
        """
            This function creates the schools table on the database and
            inserts the values into it
            @param lista the list with schools on the resultados_rna table
        """
        conn = sqlite3.connect('trabalho')
        conn.text_factory = str
        cursor = conn.cursor()
        cursor.execute("drop table if exists escolas")
        cursor.execute('''create table escolas(instituição , entradas, vagas,
                percentagem)''')
        conn.commit()
        entry = 0
        position = 0
        totalEntradas = 0
        r = ""
        data = []
        totalDeAlunos = 0
        for row in lista:
            totalDeAlunos += row[6]
            pass
        relacao = 0.0

        for row in lista:
            if(r == row[2].split(' - ')[0]):
                entry += row[6]
                position += row[8]
                relacao += (row[6] * 100.0)/totalDeAlunos
                pass
            else:
                data.append([r, entry, position, relacao])
                r = row[2].split(' - ')[0]
                entry = row[6]
                position = row[8]
                relacao = ((row[6] *100.0)/totalDeAlunos) + 0.0
                pass
            totalEntradas += entry
            pass
        contador = 0
```

```
        for dados1 in data:
            for dados2 in data:
                if dados1 != dados2:
                    if dados1[0] == dados2[0]:
                        dados1[1] += dados2[1]
                        dados1[2] += dados2[2]
                        dados1[3] += dados2[3]
                        data.pop(contador)
                        pass
                    pass
                contador += 1
                pass
            contador = 0
            pass

        '''for institution in data:
            if institution[0] == row[2].split(' - ')[0]:
                institution[1] += entry
                institution[2] += position'''
        data.pop(0)
        for x in data:
            cursor.execute('insert into escolas values(?,?,?,?);',(x[0], x[1], x[2],
x[3]))
            pass
        conn.commit()
        pass
#Constrói um gráfico que demonstra o nº de alunos colocados por
#instituição

def graficoInst(mother, lista):
        """
            Builds a graphic wich demonstrates the number of students
            positioned in each instituicion
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayEntradas = []
        for row in lista:
            arrayInstituicao.append(row[0])
            arrayEntradas.append(row[1])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayInstituicao))
        ax.set_xlabel(u'instituições')
        ax.set_ylabel('entradas')
        ax.set_title(u'Entradas em instituições')
        ax.set_xticklabels(zip(arrayInstituicao), rotation = 90)
        ax.bar(x, arrayEntradas)
        graf.show()
        pass

#Constrói um gráfico que demonstra o nº de vagas sobrantes por
#instituição

def graficoIV(mother, lista):
        """
            Builds a graphic wich demonstrates the number of vacancies
            in each instituicion
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayVagas = []
        for row in lista:
```

```
            arrayInstituicao.append(row[0])
            arrayVagas.append(row[2])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayInstituicao) - 1)
        ax.set_xlabel(u'instituições')
        ax.set_ylabel('vagas')
        ax.set_title(u'Vagas em instituições')
        ax.set_xticklabels(zip(arrayInstituicao), rotation = 90)
        graf.bar(x, arrayVagas)
        graf.show()
        print arrayVagas
        pass

# Constrói o Gráfico das estatísticas que mostra a percentagem de alunos
# numa instituição ao total de alunos

def graficoIP(mother, lista):
        """
            Builds a graphic wich demonstrates the per thousand of ocuppied
            vacancies in each instituicion
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayPercentagem = []
        for row in lista:
            arrayInstituicao.append(row[0])
            arrayPercentagem.append(row[3])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayInstituicao) - 1)
        ax.set_xlabel(u'instituições')
        ax.set_ylabel('Percentagem')
        ax.set_title(u'Percentagem de alunos em relação ao total')
        ax.set_xticklabels(zip(arrayInstituicao), rotation = 90)
        graf.bar(x, arrayPercentagem)
        graf.show()
        pass
# Constrói um gráfico que mostra os alunos colocados por distrito
def graficoDA(mother,lista):
        """
            Builds a graphic wich demonstrates the number of students
            positioned in each district
            @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayAluno = [],[]
        for row in lista:
            arrayDistrito.append(row[0])
            arrayAluno.append(row[1])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayDistrito))
        ax.set_xlabel('distritos')
        ax.set_ylabel('alunos')
        ax.set_title('Alunos colocados por distrito')
        ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
        ax.bar(x, arrayAluno)
        graf.show()
        pass
#Constrói um gráfico que mostra as vagas sobrantes po distrito
def graficoDV(mother,lista):
        """
            Builds a graphic wich demonstrates the number of vacancies
```

```
            in each district
            @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayVaga = [],[]
        for row in lista:
            arrayDistrito.append(row[0])
            arrayVaga.append(row[2])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayVDistrito))
        ax.set_xlabel('distritos')
        ax.set_ylabel('vagas')
        ax.set_title('Vagas sobrantes por distrito')
        ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
        ax.bar(x, arrayVaga)
        graf.show()
        pass
# Constrói um gráfico que mostra em permilagem os alunos que
# entraram em relação àqueles que concorreram por distrito
def graficoDP(mother,lista):
        """
            Builds a graphic wich demonstrates the per thousand of ocuppied
            vacancies in each district
            @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayPermilagem = [],[]
        for row in lista:
            arrayDistrito.append(row[0])
            arrayPermilagem.append(row[3])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayDistrito))
        ax.set_xlabel('distritos')
        ax.set_ylabel('permilagem')
        ax.set_title('Permilagem de alunos colocados por distrito')
        ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
        ax.bar(x, arrayPermilagem)
        graf.show()
        pass

def graficoDEN(mother,lista):
        """
            Builds a graphic wich demonstrates the number entrys
            in the north
            @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayEntradas = [],[]
        for row in lista:
            if ((row[0] == 'Braga') | (row[0] == 'Vila Real') |
            (row[0] == 'Aveiro') | (row[0] == 'Bragança') | (row[0] == 'Coimbra')|
            (row[0] == 'Guarda') | (row[0] == 'Porto') | (row[0] == 'Viseu')):
                arrayDistrito.append(row[0])
                arrayEntradas.append(row[1])
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayDistrito))
        ax.set_xlabel('distritos')
        ax.set_ylabel('Entradas')
        ax.set_title('Alunos colocados por distrito no Norte')
        ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
        ax.bar(x, arrayEntradas)
        graf.show()
        pass
```

```
def graficoDEC(mother,lista):
    """
        Builds a graphic wich demonstrates the number entrys
        in the center
        @param lista list with the number of students in the institutes
    """
    arrayDistrito, arrayEntradas = [],[]
    print 'olá'
    for row in lista:
        if ((row[0] == 'Lisboa') | (row[0] == 'Leiria') |
        (row[0] == 'Santarém') | (row[0] == 'Castelo Branco') |
        (row[0] == 'Setúbal') | (row[0] == 'Portalegre')):
            arrayDistrito.append(row[0])
            arrayEntradas.append(row[1])
        pass
    figs, ax = graf.subplots()
    x = np.arange(len(arrayDistrito))
    ax.set_xlabel('distritos')
    ax.set_ylabel('Entradas')
    ax.set_title('Alunos colocados por distrito no Centro')
    ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
    ax.bar(x, arrayEntradas)
    graf.show()
    pass

def graficoDES(mother,lista):
    """
        Builds a graphic wich demonstrates the number entrys
        in the south
        @param lista list with the number of students in the institutes
    """
    arrayDistrito, arrayEntradas = [],[]
    for row in lista:
        if ((row[0] == 'Faro') | (row[0] == 'Évora') |
        (row[0] == 'Beja')):
            arrayDistrito.append(row[0])
            arrayEntradas.append(row[1])
        pass
    x = np.arange(len(arrayDistrito))
    f = graf.figure()
    ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
    ax.bar(x, arrayEntradas, align='center')
    ax.set_xticks(x)
    ax.set_xticklabels(zip(arrayDistrito))
    graf.show()
    pass

def graficoDVN(mother,lista):
    """
        Builds a graphic wich demonstrates the number of vacancies
        in the north
        @param lista list with the number of students in the institutes
    """
    arrayDistrito, arrayVagas = [],[]
    for row in lista:
        if ((row[0] == 'Braga') | (row[0] == 'Vila Real') |
        (row[0] == 'Aveiro') | (row[0] == 'Bragança') | (row[0] == 'Coimbra')|
        (row[0] == 'Guarda') | (row[0] == 'Porto') | (row[0] == 'Viseu')):
            arrayDistrito.append(row[0])
            arrayVagas.append(row[2])
        pass
    figs, ax = graf.subplots()
```

```
        x = np.arange(len(arrayDistrito))
        ax.set_xlabel('distritos')
        ax.set_ylabel('Entradas')
        ax.set_title('Vagas sobrantes por distrito no Sul')
        ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
        ax.bar(x, arrayVagas)
        graf.show()
        pass

def graficoDVC(mother,lista):
        """
                Builds a graphic wich demonstrates the number of vacancies
                in the center
                @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayVagas = [],[]
        print 'olá'
        for row in lista:
                if ((row[0] == 'Lisboa') | (row[0] == 'Leiria') |
                (row[0] == 'Santarém') | (row[0] == 'Castelo Branco') |
                (row[0] == 'Setúbal') | (row[0] == 'Portalegre')):
                        arrayDistrito.append(row[0])
                        arrayVagas.append(row[2])
                pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayDistrito))
        ax.set_xlabel('distritos')
        ax.set_ylabel('Entradas')
        ax.set_title('Alunos colocados por distrito no Centro')
        ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
        ax.bar(x, arrayVagas)
        graf.show()
        pass

def graficoDVS(mother,lista):
        """
                Builds a graphic wich demonstrates the number of vacancies
                in the south
                @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayVagas = [],[]
        for row in lista:
                if ((row[0] == 'Faro') | (row[0] == 'Évora') |
                (row[0] == 'Beja')):
                        arrayDistrito.append(row[0])
                        arrayVagas.append(row[2])
                pass
        x = np.arange(len(arrayDistrito))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayVagas, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(zip(arrayDistrito))
        graf.show()
        pass

def graficoDPN(mother,lista):
        """
                Builds a graphic wich demonstrates the per thousand
                of ocuppied vacancies in the north
                @param lista list with the number of students in the institutes
        """
        arrayDistrito, arrayPermilagem = [],[]
```

```
    for row in lista:
        if ((row[0] == 'Braga') | (row[0] == 'Vila Real') |
        (row[0] == 'Aveiro') | (row[0] == 'Bragança') | (row[0] == 'Coimbra')|
        (row[0] == 'Guarda') | (row[0] == 'Porto') | (row[0] == 'Viseu')):
            arrayDistrito.append(row[0])
            arrayPermilagem.append(row[3])
        pass
    figs, ax = graf.subplots()
    x = np.arange(len(arrayDistrito))
    ax.set_xlabel('distritos')
    ax.set_ylabel('Entradas')
    ax.set_title('Permilagem de alunos colocados por distrito no Norte')
    ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
    ax.bar(x, arrayPermilagem)
    graf.show()
    pass

def graficoDPC(mother,lista):
    """
        Builds a graphic wich demonstrates the per thousand
        of ocuppied vacancies in the center
        @param lista list with the number of students in the institutes
    """
    arrayDistrito, arrayPermilagem = [],[]
    for row in lista:
        if ((row[0] == 'Lisboa') | (row[0] == 'Leiria') |
        (row[0] == 'Santarém') | (row[0] == 'Castelo Branco') |
        (row[0] == 'Setúbal') | (row[0] == 'Portalegre')):
            arrayDistrito.append(row[0])
            arrayPermilagem.append(row[3])
        pass
    figs, ax = graf.subplots()
    x = np.arange(len(arrayDistrito))
    ax.set_xlabel('distritos')
    ax.set_ylabel('Entradas')
    ax.set_title('Permilagem de alunos colocados por distrito no Centro')
    ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
    ax.bar(x, arrayPermilagem)
    graf.show()
    pass

def graficoDPS(mother,lista):
    """
        Builds a graphic wich demonstrates the per thousand
        of ocuppied vacancies in the south
        @param lista list with the number of students in the institutes
    """
    arrayDistrito, arrayPermilagem = [],[]
    for row in lista:
        if ((row[0] == 'Faro') | (row[0] == 'Évora') |
        (row[0] == 'Beja')):
            arrayDistrito.append(row[0])
            arrayPermilagem.append(row[3])
        pass
    #figs, ax = graf.subplots()

    #ax.set_xlabel('distritos')
    #ax.set_ylabel('Entradas')
    #ax.set_title('Permilagem de alunos colocados por distrito no Sul')
    #ax.set_xticklabels(zip(arrayDistrito), rotation = 90)
    #ax.bar(x, arrayPermilagem)
    x = np.arange(len(arrayDistrito))
    f = graf.figure()
```

```
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayPermilagem, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(zip(arrayDistrito))
        graf.show()
        pass

def graficoEscolasEntradas(mother, lista):
        """
                Builds a graphic wich demonstrates the number of entrys
                in each school
                @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayEntradas = []
        for row in lista:
                if 'Escola' in row[0]:
                        arrayInstituicao.append(detectUpper(row[0]))
                        arrayEntradas.append(row[1])
                        pass
                pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayInstituicao))
        ax.set_xlabel(u'instituições')
        ax.set_ylabel('entradas')
        ax.set_title(u'Entradas em escolas')
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        ax.bar(x, arrayEntradas)
        graf.show()
        pass

def graficoEscolasVagas(mother, lista):
        """
                Builds a graphic wich demonstrates the number of vacancies
                in each school
                @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayVagas = []
        for row in lista:
                if 'Escola' in row[0]:
                        if row[2] != 0:
                                arrayInstituicao.append(detectUpper(row[0]))
                                arrayVagas.append(row[2])
                                pass
                        else:
                                arrayInstituicao.append(detectUpper(row[0]))
                                arrayVagas.append(1)
                                pass
                pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayInstituicao))
        ax.set_xlabel(u'instituições')
        ax.set_ylabel('entradas')
        ax.set_title(u'vagas em escolas')
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        print arrayVagas
        ax.bar(x, arrayVagas)
        graf.show()
        pass

def graficoEscolasPercentagem(mother, lista):
        """
```

```
            Builds a graphic wich demonstrates the per thousand of ocuppied
            vacancies in each school
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayPercentagem = []
        for row in lista:
            if 'Escola' in row[0]:
                arrayInstituicao.append(detectUpper(row[0]))
                arrayPercentagem.append(row[3])
                pass
            pass
        figs, ax = graf.subplots()
        x = np.arange(len(arrayInstituicao))
        ax.set_xlabel(u'instituições')
        ax.set_ylabel('entradas')
        ax.set_title(u'Percentagem de alunos em relação ao total')
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        ax.bar(x, arrayPercentagem)
        graf.show()
        pass

def graficoInstitutosEntradas(mother, lista):
        """
            Builds a graphic wich demonstrates the number of entrys
            in each institute
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayEntradas = []
        for row in lista:
            if (('Instituto' in row[0])|('ISCTE' in row[0])):
                arrayInstituicao.append(detectUpper(row[0]))
                arrayEntradas.append(row[1])
                pass
            pass
        x = np.arange(len(arrayInstituicao))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayEntradas, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        graf.show()
        pass

def graficoInstitutosVagas(mother, lista):
        """
            Builds a graphic wich demonstrates the number of vacancies
            in each institute
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayVagas = []
        for row in lista:
            if (('Instituto' in row[0])|('ISCTE' in row[0])):
                arrayInstituicao.append(detectUpper(row[0]))
                arrayVagas.append(row[2])
                pass
            pass
        x = np.arange(len(arrayInstituicao))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayVagas, align='center')
```

```
        ax.set_xticks(x)
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        graf.show()
        pass

def graficoInstitutosPercentagem(mother, lista):
        """
                Builds a graphic wich demonstrates the per thousand of ocuppied
                vacancies in each institute
                @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayPercentagem = []
        for row in lista:
                if (('Instituto' in row[0])|('ISCTE' in row[0])):
                        arrayInstituicao.append(detectUpper(row[0]))
                        arrayPercentagem.append(row[3])
                        pass
                pass
        x = np.arange(len(arrayInstituicao))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayPercentagem, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        graf.show()
        pass

def graficoUniversidadesEntradas(mother,lista):
        """
                Builds a graphic wich demonstrates the number of students
                positioned in each university
                @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayEntradas = []
        for row in lista:
                if (('Universidade') in row[0]):
                        arrayInstituicao.append(detectUpper(row[0]))
                        arrayEntradas.append(row[1])
                        pass
                pass
        x = np.arange(len(arrayInstituicao))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayEntradas, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        graf.show()
        pass

def graficoUniversidadesVagas(mother,lista):
        """
                Builds a graphic wich demonstrates the number of vacancies
                in each university
                @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayVagas = []
        for row in lista:
                if (('Universidade') in row[0]):
                        arrayInstituicao.append(detectUpper(row[0]))
                        arrayVagas.append(row[2])
```

```
                pass
            pass
        x = np.arange(len(arrayInstituicao))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayVagas, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        graf.show()
        pass

def graficoUniversidadesPercentagem(mother,lista):
        """
            Builds a graphic wich demonstrates the per thousand of ocuppied
            vacancies in each university
            @param lista list with the number of students in the institutes
        """
        arrayInstituicao = []
        arrayPercentagem = []
        for row in lista:
            if (('Universidade') in row[0]):
                arrayInstituicao.append(detectUpper(row[0]))
                arrayPercentagem.append(row[3])
                pass
            pass
        x = np.arange(len(arrayInstituicao))
        f = graf.figure()
        ax = f.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.bar(x, arrayPercentagem, align='center')
        ax.set_xticks(x)
        ax.set_xticklabels(tuple(arrayInstituicao), rotation = 90)
        graf.show()
        pass

def detectUpper(x):
        b = ''
        for letter in x:
            if letter.isupper() == True:
                b += letter
                pass
            pass
        return b
        pass
```