Relatório Projeto de Programação

Python

2º Semestre

Duarte Pinto Rodrigues

Gestão de Sistemas de Informação, 1º ano

Duarte Pinto Rodrigues
A22004018

Introdução

Este projeto foi desenvolvido no contexto da cadeira de Programação II, onde o professor responsável pela cadeira disponibilizou o seguinte enunciado:

"O Instituto Politécnico decidiu desenvolver um novo sistema de gestão de turmas e solicitou à equipa de Gestão de Sistemas de Informação para programar o software que irá ser usado para fazer esta gestão.
Foram entregues os requisitos técnicos que se pretendem ver ser desenvolvidos nesta aplicação, os alu-nos têm o objetivo de realizarem uma aplicação com uma linguagem de programação dinâmica: Python.
A escrita de código adicional na linguagem de programação C/C++ com o eventual uso de bibliotecas adicionais é possível, desde que não ultrapasse 20% do código total do projeto.
Uma porção do programa funcionará nos computadores clientes, e outra parte do código irá funcionar num servidor, que contém também uma base de dados, seguindo então o seguinte esquema:

O utilizador apenas pode interagir com o código que se encontra do lado do cliente, e todo o código executado no servidor deverá de funcionar de forma "passiva", que reage à informação que chega do código do lado do cliente. Deverá ser possível então que cliente e servidor possam trocar informação.
Outro dos requisitos estipulados foi ter o SQLite como Sistema de Gestão de Base de Dados no servidor."

Este trabalho é o resultado do desenvolvimento de toda a proposta para este enunciado.

Duarte Pinto Rodrigues
A22004018

Índice de ilustrações

Código do Projeto

Ficheiro 'Classes.py':

Onde se define as classes de alunos, disciplinas e professores.

Classe Utente

```
class Utente():
    nome = ""
    idade = 0
    morada = ""
```

Figura 1 - Classe 'Utente'

A primeira classe criada tem como objetivo inicializar certas características que são partilhadas entre alunos e professores, sendo essas o nome, a morada(variáveis de tipo *String*) e idade(variável de tipo *Int*)

Subclasse Aluno

```
class Aluno(Utente):
    numero_aluno = 0 ## são 8 dígitos
    cc = 0

    def __init__ (self, numero_aluno, nome, idade, morada, cc):
        self.numero_aluno = numero_aluno
        self.nome = nome
        self.idade = idade
        self.morada = morada
        self.cc = cc
        pass

    def mostrarNumeroAluno(self):
        return self.numero_aluno
        pass

    def mostrarNomeAluno(self):
        return self.nome
        pass
```

Figura 2 - Subclasse 'Aluno'

Neste excerto de código é criada a subclasse 'Aluno', com os campos 'numero_aluno' e 'cc', que são os campos que distinguirão o aluno de um professor, considerando que ambos partilham.

O código contém também uma função '__init__' que é um método reservado para inicializar os atributos da subclasse.

Para além da função de inicialização, existem ainda duas funções de seu nome 'mostrarNumeroAluno' e 'mostrarNomeAluno'. Cada uma dessas funções devolve o nome e o número do aluno(respetivamente) quando são chamadas.

Duarte Pinto Rodrigues
A22004018

Subclasse Professor

```
class Professor(Utente):
    numero_prof = 0 ## são 5 dígitos
    categoria_prof = ""
    anos_exp = 0

    def __init__ (self, numero_prof, nome, idade, morada, categoria_prof, anos_exp):
        self.numero_prof = numero_prof
        self.nome = nome
        self.idade = idade
        self.morada = morada
        self.categoria_prof = categoria_prof
        self.anos_exp = anos_exp
        pass

    def mostrarNumeroProf(self):
        return self.numero_prof
        pass

    def mostrarNomeProf(self):
        return self.nome
        pass
```

Figura 3 - Subclasse 'Professor'

À semelhança da subclasse anterior, a subclasse ‘Professor’ possuí uma função de inicialização, e outras duas para devolver o número de professor(‘mostrarNumeroProf’) e o nome do mesmo(‘mostrarNomeProf’).

Apesar das duas subclasses presentes neste projeto serem bastante similares, o que as diferencia são os campos que cada uma utiliza, neste caso, a subclasse ‘Professor’ usa os campos ‘numero_prof’, ‘anos_prof’(tipo int) e ‘categoria_prof’(tipo String)

Classe Disciplina

```
class Disciplina():
    numero_aluno = 0
    #nome_aluno = ""
    nome_aluno = list()
    nome_disc = ""
    nome_prof = ""
    numero_prof = 0

    def __init__ (self,  nome_disc, nome_prof, numero_prof):
        self.nome_disc = nome_disc
        self.nome_prof = nome_prof
        self.numero_prof = numero_prof
        self.nome_aluno = list()
        pass

    def mostrarNomeDisc(self):
        return(self.nome_disc)
        pass

    def RegistarProf(self, nome_disc, nome_prof, numero_prof):
        self.nome_disc = nome_disc
        self.nome_prof = nome_prof
        self.numero_prof = numero_prof
        pass

    def InscreverAluno(self, aluno):
         self.nome_aluno.append(aluno)
         pass

    def mostrarInscFiltrada(self):
        for aluno in self.nome_aluno:
            print("|", aluno.mostrarNumeroAluno(),
                  "|", aluno.mostrarNomeAluno(),
                  "|", aluno.mostrarNomeDisc(),
                  "|", aluno.mostrarNomeProf(),
                  "|", aluno.mostrarNumeroProf())
        pass
```

Figura 4 - Classe ''Disciplina

A classe Disciplina apenas requer que o utilizador insira o nome desejado no ficheiro main.

O código tem uma função __init__, como todas as outras, para definir os atributos da classe.

O código tem uma função para retornar o nome da disciplina('mostrarNomeDisc'), outra para registar um professor numa disciplina('RegistarProf'), outra para inscrever um aluno numa disciplina('InscreveAluno') e outra função para mostrar uma lista com os nomes dos alunos e os respetivos números que estavam inscritos numa disciplina específica.

(Tive uma dificuldade nesta parte do código, principalmente a desenvolver uma parte do código onde fossem devolvidos os dados dos alunos numa determinada disciplina, porém com a ajuda do professor, esta foi desenvolvida com sucesso.)

Ficheiro 'Main.py':

Os maiores propósitos do ficheiro 'Main.py' eram:

- 'Juntar' todas as funções do ficheiro 'Modulos.py'
- Estabelecer uma ligação ao servidor criado no ficheiro 'Servidor.py' e enviar o número da opção introduzida pelo utilizador.

Para que o programa no ficheiro 'Main.py' funcione, este necessita a execução prévia do ficheiro 'Servidor.py'.

Em quase todo o código do ficheiro, existe uma estrutura de código recorrente, que consiste em chamar a função equivalente à da opção escolhida pelo utilizador(o número da função escolhida é enviada para o servidor para mais funções serem executadas), sendo esta função proveniente do ficheiro 'Modulos.py' e uma secção de código que pede ao utilizador pelo número da opção para voltar ou não de volta para o menu principal.

```
while escolha != 0:
    Modulos.menu()
    escolha = int(input("Insira o número correspondente à sua escolha: "))
    s.send(str(escolha).encode('utf8'))

    if escolha == 1:
        print("Criar Disciplina")
        Modulos.adicionarDisciplinas(s, lista_disciplinas)

        volta = 50
        volta = int (input("Pretende voltar ao menu?(1-Sim / 0-Não):"))

        if(volta==0):
            os.system("cls")
            print("Obrigado por utilizar a plataforma")
        elif(volta==1):
            os.system("cls")
            print("")
        else:
            print("Valor inválido")
            volta = int (input("Pretende voltar ao menu?(1-Sim/0-Não):"))
        ##fim da tarefa 1 - Criar Disciplina - Funciona
```

Figura 5 - Exemplo 1: processo de chamada de uma função do ficheiro 'Modulos.py'

Através do IP 127.0.0.1 e a porta 5000, é possível estabelecer uma ligação com o servidor criado no mesmo IP e com a mesma porta, para este servidor vão ser enviadas inúmeras informações para outro tipo de funções.

```
ip = "127.0.0.1"
porto = 5000
escolha = 50

s.connect((ip, porto))
```

Figura 6 - IP, porto e o comando de conexão

Para além da conexão, ainda são declaradas as listas onde vão ser gravados em memória os dados dos alunos, professores e disciplinas

```
lista_alunos = list()
lista_professores = list()
lista_disciplinas = list()
```

Figura 7 - Listas

Ficheiro 'Modulos.py':

Função 'menu'

Uma função relativamente simples, a função 'menu' é uma série de prints que vai ser mostrada ao utilizador para apresentar as opções ao mesmo, ainda contendo quatro prints com cardinais por uma questão puramente estética.

```
def menu():
    print("")
    print("##############################################")
    print("# Bem-Vindo/a                                #")
    print("##############################################")
    print("# 1) Criar Disciplina                        #")
    print("# 2) Listar Disciplina                       #")
    print("# 3) Eliminar Disciplina                     #")
    print("# 4) Criar Aluno                             #")
    print("# 5) Inscrever Aluno                         #")
    print("# 6) Eliminar Aluno                          #")
    print("# 7) Listar Alunos                           #")
    print("# 8) Listar Alunos Inscritos Numa Disciplina #")
    print("# 9) Criar Professor                         #")
    print("# 10) Adicionar Professor a Disciplina       #")
    print("# 11) Importar Alunos de um Ficheiro         #")
    print("##############################################")
    print("# 0) Sair                                    #")
    print("##############################################")
    pass
```

Figura 8 - Função 'menu'

Função 'adicionarDisciplina'

```
def adicionarDisciplinas(s, lista_disciplinas):
    nomeDisciplina = input("Indique o Nome da Diciplina:")
    s.send(nomeDisciplina.encode())
    nomeProf = ""
    numeroProf = ""

    nova_disc = Classes.Disciplina(nomeDisciplina, nomeProf, numeroProf)
    lista_disciplinas.append(nova_disc)

    print("Disciplina Registada!")
    pass
```

Figura 9 - Função 'adicionarDisciplina'

A função 'adicionarDisciplinas' permite ao utilizador introduzir o nome de uma disciplina e, após introduzir, envia-o para o servidor e envia o nome, em conjunto com o nome do professor e o respetivo número em branco para a lista e guarda todos esses dados em memória.

Função 'listarDisciplina'

Esta função cria um ciclo *for* e por cada disciplina na lista 'lista_disciplina', o ciclo imprime um número, um hífen e lista o nome da disciplina, se houver mais que uma disciplina, o número que antecede o nome da disciplina aumenta.

O processo de mostrar o nome da disciplina provém de uma função da Classe 'Disciplina' e tem o nome 'mostrarNomeDisc'.

```
def listarDisciplinas(lista_disciplinas):
    print("Lista de disciplinas registadas: \n")
    contador = 1
    for disciplinas in lista_disciplinas:
        print(contador, " - ", disciplinas.mostrarNomeDisc())
        contador = contador + 1
    pass
```

Figura 10 - Função 'listarDisciplina'

Função 'eliminarDisciplina'

A função 'eliminarDisciplinas' cria o mesmo processo que a função anterior e pede o input do utilizador para o número de uma das disciplinas pois estão numeradas, depois de receber o input, o código entra num processo onde se o número estiver presente na lista, o pedaço de código '*lista_disciplinas.pop(ind)*' é executado, apagando da lista o conteúdo, neste caso, o nome da lista, que se encontrava nessa posição. Se o que o utilizador inseriu não estiver presente na lista, é devolvido o print "Ocorreu um erro.".

(Para que a opção do utilizador seja a mesma que a posição em memória na lista que o utilizador escolheu, à variável é lhe subtraída 1 valor)

```
def eliminarDisciplinas(lista_disciplinas):
    listarDisciplinas(lista_disciplinas)
    ind = int (input("Indique a posicao da disciplina que deseja eliminar: "))
    ind = ind - 1
    try:
        lista_disciplinas.pop(ind)
    except:
        print("Ocorreu um erro.")

    listarDisciplinas(lista_disciplinas)
    pass
```

Figura 11 - Função 'eliminarDisciplina'

Duarte Pinto Rodrigues
A22004018

Função ‘adicionarAluno’

Esta função pede vários dados ao utilizador, o nome do aluno, a idade, a morada e número do cc. Para além disso ainda gera automaticamente um número de aluno.

Também vai enviar o nome do aluno e o número do mesmo para o servidor.

Todos estes dados vão ser guardados numa variável do tipo Classe.Aluno e depois são guardados na lista, ‘lista_alunos’.

```
def adicionarAluno(s, lista_alunos):
    numero_aluno = randint(22000000, 22009999)
    s.send(str(numero_aluno).encode('utf8'))
    nome_aluno = input("Insira o nome do/a aluno/a: ")
    s.send(nome_aluno.encode())
    idade_aluno = int(input("Insira a idade do/a aluno/a: "))
    morada_aluno = input("Insira a morada do/a aluno/a: ")
    cc_aluno = int(input("Insira o numero do CC do/a aluno/a: "))

    novo_aluno = Classes.Aluno(numero_aluno,
                               nome_aluno,
                               idade_aluno,
                               morada_aluno,
                               cc_aluno)
    lista_alunos.append(novo_aluno)
    pass
```

Figura 12 - Função 'adicionarAluno’

Duarte Pinto Rodrigues
A22004018

Função 'inscreverAlunos'

Esta função permite ao utilizador escolher o aluno que quiser da lista 'lista_alunos' e a disciplina que quiser da lista 'lista_disciplinas' e inscreve o aluno escolhido na disciplina escolhida.

Todo esse registo fica guardado numa lista que está declarada na classe 'Disciplina' no ficheiro 'Classe.py.'

Portanto, de certa forma, dentro da lista 'lista disciplinas', há uma lista de alunos por cada disciplina registada, e é dentro dessa lista de alunos que o utilizador regista nesta função.

(Algo que não consegui desenvolver por falta de conhecimento foi impedir que o utilizador inscrevesse um aluno numa disciplina que ainda não tivesse um professor atribuído)

```
def inscreverAluno(lista_alunos, lista_disciplinas, s):
    listarAlunos(lista_alunos)
    indaln = int(input("Insira o índice do aluno que pretende inscrever:"))
    indaln = indaln - 1
    listarDisciplinas(lista_disciplinas)
    inddisc = int(input("Insira o índice da disciplina em que pretende inscreve
    inddisc = inddisc - 1

    aluno = lista_alunos[indaln]
    lista_disciplinas[inddisc].InscreverAluno(aluno)

    pass
```

Figura 13 - Função 'inscreverAluno'

Função 'eliminarAluno'

Tal como a função 'eliminarDisciplina', esta função permite ao utilizador escolher um aluno da lista 'lista_alunos' e indicar a posição dele/a e eliminá-lo/a da lista, juntamente com os seus dados.

```
def eliminarAlunos(lista_alunos):
    listarAlunos(lista_alunos)
    ind = int (input("Indique a posicao do/a aluno/a que deseja eliminar: "))
    ind = ind - 1

    try:
        lista_alunos.pop(ind)

    except:
        print("Ocorreu um erro.")

    listarAlunos(lista_alunos)
    pass
```

Figura 14 - Função 'eliminarAluno'

Duarte Pinto Rodrigues
A22004018

Função 'listarAlunos'

Tal como a função 'listarDisciplina', esta função vai listar todos os alunos já registados com o seguinte formato:

Posição na lista – Número do aluno – Nome do aluno

A função irá mostrar dados neste formato enquanto houver itens na lista 'lista_alunos'.

```
def listarAlunos(lista_alunos):
    print("Lista de alunos/as registados/as: \n")
    contador = 1
    for alunos in lista_alunos:
        print(contador, " - ",
              alunos.mostrarNumeroAluno(), "-",
              alunos.mostrarNomeAluno())
        contador = contador + 1
    pass
```

Figura 15 - Função 'listarAlunos'

Função 'listarAlunosdeDisciplinas'

Esta função permite ao utilizador indicar quais são os alunos que quer ver numa lista consoante à escolha do mesmo.

Ao lhe ser mostrado uma lista com o mesmo formato do da função 'listarDisciplina' o utilizador escolhe a disciplina e é mostrada uma lista com o seguinte formato:

|Número do aluno|Nome do aluno|Disciplina|Nome do Professor|Número do Professor|

```
def listarAlunosdeDisciplinas(lista_disciplinas):
    listarDisciplinas(lista_disciplinas)
    inddisc = int(input("Insira o indice da disciplina em que pretende ver os
    inddisc = inddisc - 1

    lista_disciplinas[inddisc].mostrarInscFiltrada()
    pass
```

Figura 16 - Função 'listarAlunosdeDisciplina'

Duarte Pinto Rodrigues
A22004018

Função 'adicionarProfessor'

Esta função pede vários dados ao utilizador, o nome do aluno, a idade, a morada, a categoria de professor e os anos de experiência. Para além disso ainda gera automaticamente um número de professor.

Também vai enviar o nome do professor e o número do mesmo para o servidor.

Todos estes dados vão ser guardados numa variável do tipo Classe.Professor e depois são guardados na lista, 'lista_professor'.

```
def adicionarProf(s, lista_profs):
    numero_prof = randint(60000, 65000)
    s.send(str(numero_prof).encode('utf8'))
    nome_prof = input("Insira o nome do/a professor/a: ")
    s.send(nome_prof.encode())
    idade_prof = int(input("Insira a idade do/a professor/a: "))
    morada_prof = input("Insira a morada do/a professor/a: ")
    categoria_prof = input("Insira a categoria do/a professor/a: ")
    anos_exp = int(input("Insira os anos de experiencia do/a professor/a: "))

    novo_prof = Classes.Professor(numero_prof,
                                  nome_prof,
                                  idade_prof,
                                  morada_prof,
                                  categoria_prof,
                                  anos_exp)
    lista_profs.append(novo_prof)
    pass
```

Figura 17 - Função 'adicionarProf'

Função 'listarProfs'

Esta função cria um ciclo *for* e por cada disciplina na lista 'lista_professores', o ciclo imprime um número, um hífen e lista o nome da disciplina, se houver mais que uma disciplina, o número que antecede o nome da disciplina aumenta.

Os processos de mostrar o nome e número do professor provêm de uma função da Subclasse 'Professor' e têm o nome 'mostrarNomeProf' e 'mostrarNumeroProf'.

Tal como a função 'listarDisciplina', esta função vai listar todos os alunos já registados com o seguinte formato:

Posição na lista – Número do aluno – Nome do aluno

```
def listarProfs(lista_profs):
    print("Lista de professores/as registados/as: \n")
    contador = 1
    for profs in lista_profs:
        print(contador," - ",
              profs.mostrarNumeroProf() ,"-",
              profs.mostrarNomeProf())
        contador = contador + 1
    pass
```

Figura 18 – Função 'listarProfs'

Duarte Pinto Rodrigues
A22004018

Função ‘adProfADisciplina’

Nesta função, são apresentadas duas listas a lista ’lista_professores’ e a lista ‘lista_disciplinas’ com o mesmo formato referido anteriormente e duas opções para o utilizador selecionar.

O professor que o utilizador selecionar irá ser colocado da lista ’lista_professores’ para a lista ‘lista_disciplina’ no campo que foi selecionado.

```
def adProfADisciplina(lista_disciplinas, lista_professores):
    listarDisciplinas(lista_disciplinas)
    inddisc = int(input("Indique o numero da disciplina que quer inscrever o pr
    inddisc = inddisc - 1
    listarProfs(lista_professores)
    indprof = int(input("Indique o numero do/a profesor/a para inscrever na dis
    indprof = indprof - 1

    nome_disc = lista_disciplinas[inddisc].nome_disc
    nome_prof = lista_professores[indprof].nome
    num_prof = lista_professores[indprof].numero_prof

    lista_disciplinas[inddisc].RegistarProf(nome_disc, nome_prof, num_prof)
    print("Inscrito com sucesso")
    pass
```

Figura 19 - Função 'adProfADisciplina'

Duarte Pinto Rodrigues
A22004018

Função 'lerFicheiro'

Esta função vai importar todos os dados que se situam na Folha1 do ficheiro "ExcelProjeto.xlsx" executando um ciclo for para correr todas as linhas das colunas com nome:

- nome_aluno
- Idade_aluno
- morada_aluno
- cc_aluno
- numero_aluno

Depois de serem importados, os dados são guardados na lista 'lista_alunos' e essa mesma lista é mostrada ao utilizador

(Quando esta função estava a ser desenvolvida, foi necessário instalar um conjunto de *libraries* e reinstalar o IDLE *Spyder*)

```
def lerFicheiro(lista_alunos):
    f = pandas.read_excel('ExcelProjeto.xlsx' , sheet_name='Folha1')

    for i in f.index:
        nome_aluno = f['nome_aluno'][i]
        idade_aluno= f['idade_aluno'][i]
        morada_aluno = f['morada_aluno'][i]
        cc_aluno =  f['cc_aluno'][i]
        numero_aluno = f['numero_aluno'][i]

        novo_aluno = Classes.Aluno(numero_aluno,
                                   nome_aluno,
                                   idade_aluno,
                                   morada_aluno,
                                   cc_aluno)
        lista_alunos.append(novo_aluno)
        print("Ficheiro importado e registos registados em memória")
        pass
    listarAlunos(lista_alunos)
    pass
```


Duarte Pinto Rodrigues
A22004018

Ficheiro 'Servidor.py'

Este ficheiro serve para receber a informação dos ficheiros 'Modulos.py' e 'Main.py' e com essa informação, faz uma ação específica, porém todas elas envolvem enviar informações que foram registadas em funções do ficheiro 'Modulos.py' e enviá-las para uma base de dados.

A base de dados criada para este projeto, o ficheiro 'ProjetoPython.db' vai receber os dados vindos do ficheiro 'Servidor.py' e vai guardá-los em três tabelas diferentes, a tabela 'Alunos', 'Disciplinas' e 'Professores'

Estes dados são enviados para a base de dados através de código em SQL que está estruturado dentro do python de forma a evitar um processo de SQL injection.

```
import socket
import sqlite3

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

dbNome = "ProjetoPython.db"

global conn
conn = sqlite3.connect(dbNome)

ip = "127.0.0.1"
porto = 5000
s.bind((ip, porto ))
s.listen(5)
```

Figura 20 - Estruturar o ficheiro 'Servidor.py'

```
while True:
    escolha = c.recv(1024)
    x = escolha.decode()
    choice = int(x)

    print ("A escolha é ", x)

    ##Ações das escolhas

    if choice == 1:
        nome_disc = c.recv(1024)
        disciplina = nome_disc.decode()

        print("Disciplina a inserir: ", disciplina)

        sql = """
        INSERT INTO Disciplinas(NOME_DISC, CODIGO_PROF, NOME_PROF)
        VALUES (?, ?, ?);
        """

        bd = conn.cursor()

        dados_disc = (disciplina, "", "")
        try:
            bd.execute(sql, dados_disc)
            conn.commit()
        except:
            print("O registo já existe")
```

Figura 21 - Exemplo de código SQL embutido no ficheiro de python 'Servidor.py'

Conclusões

Com a concretização deste projeto, acredito que não só aprendi e apurei os meus conhecimentos sobre a linguagem de python, mas também apurei as minhas habilidades no que toca a resolução de problemas em programação e entender a percetibilidade de uma máquina quando executa o código que lhe é dado.

Acredito que a parte mais difícil do projeto em geral foi adaptar-me a esta nova linguagem que apesar de oferecer uma grande capacidade de desenvolvimento, trabalhar com uma língua como o python pode vir a tornar-se confuso depois de trabalhar tanto tempo com uma linguagem como a linguagem C.

A mudança de uma linguagem cuja complexidade necessária para desenvolver uma quantia de código cria um estilo de pensamento que pode considerar-se desnecessário quando em comparação ao python, o pensamento lógico é afetado de certa forma, e, falo por mim, que se acaba sempre por desenvolver mais código do que necessário.

Apesar das dificuldades, apreciei o desafio proposto pois acredito que tudo o que aprendi com ele vai ser uma mais valia para o futuro.

Referencias

Para este trabalho, foram utilizados os slides disponibilizados pelo professor da cadeira e as aulas de dúvidas dadas pelos mesmos