# Relatório Projeto de Programação

Python

2º Semestre

Duarte Pinto Rodrigues

Gestão de Sistemas de Informação, 1º ano



# 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.

# Índice de ilustrações

[Figura 1 - Classe 'Utente' 5](#_Toc73271069)

[Figura 2 - Subclasse 'Aluno' 5](#_Toc73271070)

[Figura 3 - Subclasse 'Professor' 6](#_Toc73271071)

[Figura 4 - Classe ''Disciplina 7](#_Toc73271072)

[Figura 5 - Exemplo 1: processo de chamada de uma função do ficheiro 'Modulos.py' 8](#_Toc73271073)

[Figura 6 - IP, porto e o comando de conexão 8](file:///C:\Users\Duart\Downloads\Relatório%20Projeto%20de%20Programação.docx#_Toc73271074)

[Figura 7 - Listas 8](file:///C:\Users\Duart\Downloads\Relatório%20Projeto%20de%20Programação.docx#_Toc73271075)

[Figura 8 - Função 'menu' 9](#_Toc73271076)

[Figura 9 - Função 'adicionarDisciplina' 9](#_Toc73271077)

[Figura 10 - Função 'listarDisciplina' 10](#_Toc73271078)

[Figura 11 - Função 'eliminarDisciplina' 10](#_Toc73271079)

[Figura 12 - Função 'adicionarAluno’ 11](#_Toc73271080)

[Figura 13 - Função 'inscreverAluno' 12](#_Toc73271081)

[Figura 14 - Função 'eliminarAluno’ 12](#_Toc73271082)

[Figura 15 - Função 'listarAlunos' 13](#_Toc73271083)

[Figura 16 - Função 'listarAlunosdeDisciplina' 13](#_Toc73271084)

[Figura 17 - Função 'adicionarProf' 14](#_Toc73271085)

[Figura 18 – Função ‘listarProfs’ 14](#_Toc73271086)

[Figura 19 - Função 'adProfADisciplina' 15](#_Toc73271087)

[Figura 20 - Estruturar o ficheiro 'Servidor.py' 17](#_Toc73271088)

[Figura 21 - Exemplo de código SQL embutido no ficheiro de python 'Servidor.py' 17](#_Toc73271089)

# Código do Projeto

# Ficheiro ‘Classes.py’:

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

## Classe Utente

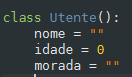


Figura 1 - Classe 'Utente'

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

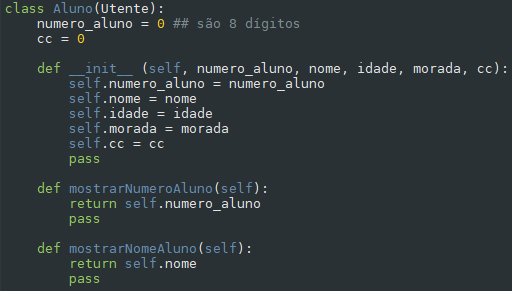


Figura 2 - Subclasse 'Aluno'

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.

## Subclasse Professor

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

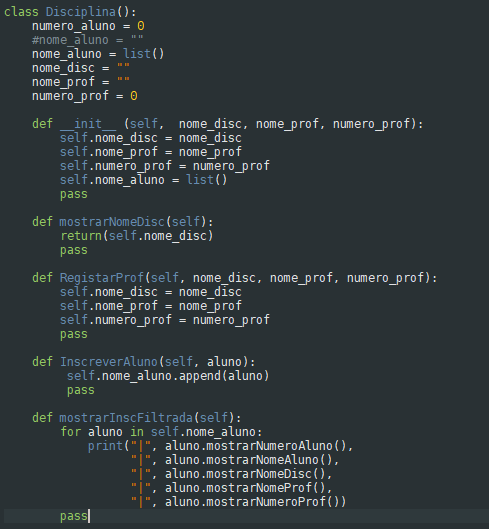


Figura 4 - Classe ''Disciplina

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.

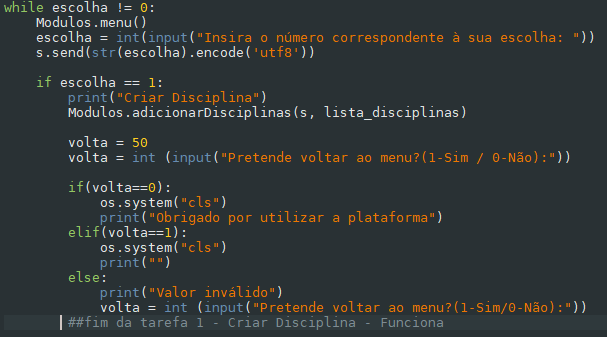


Figura 5 - Exemplo 1: processo de chamada de uma função do ficheiro 'Modulos.py'

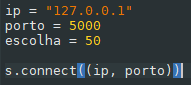
Através do IP 127.0.0.1 e a porta 5000, é possível estabelecer uma ligação com o servidor criado no mesmo IP e com a mesma porta, para este servidor vão ser enviadas inúmeras informações para outro tipo de funções.

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

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.

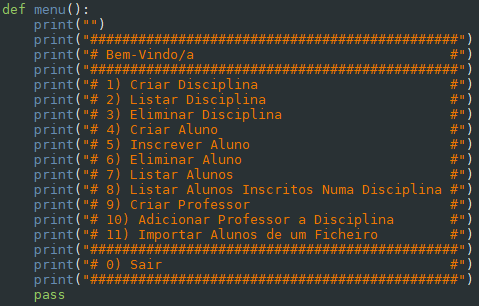


Figura 8 - Função 'menu'

## Função ‘adicionarDisciplina’

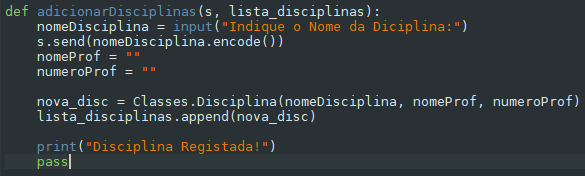


Figura 9 - Função 'adicionarDisciplina'

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’.

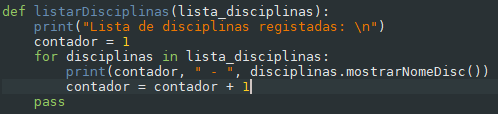


Figura 10 - Função 'listarDisciplina'

## 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)

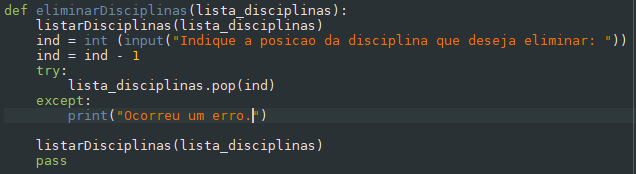


Figura 11 - Função 'eliminarDisciplina'

## 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’.

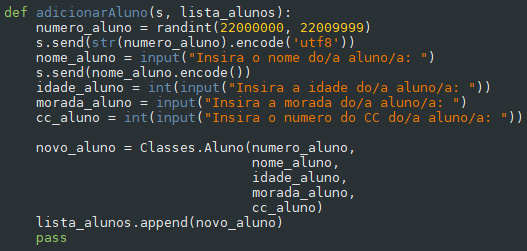


Figura 12 - Função 'adicionarAluno’

## 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)

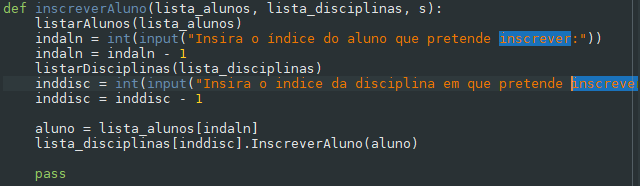


Figura 13 - Função 'inscreverAluno'

## 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.

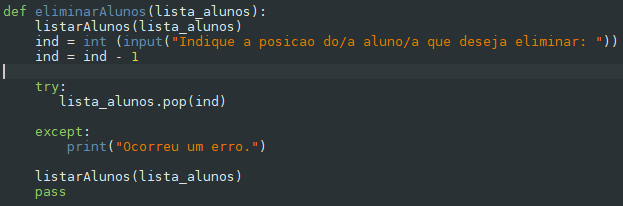


Figura 14 - Função 'eliminarAluno’

## 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’.

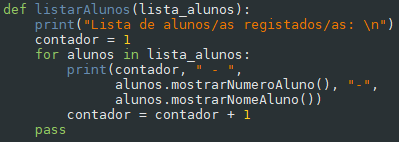


Figura 15 - Função 'listarAlunos'

## 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|

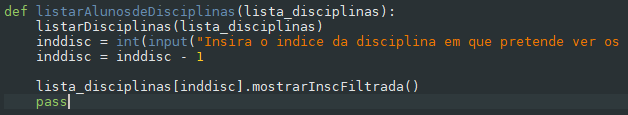


Figura 16 - Função 'listarAlunosdeDisciplina'

## 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’.

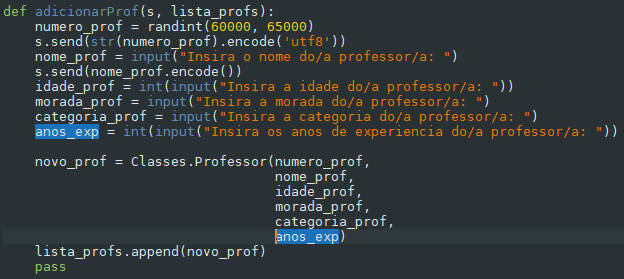


Figura 17 - Função 'adicionarProf'

## 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

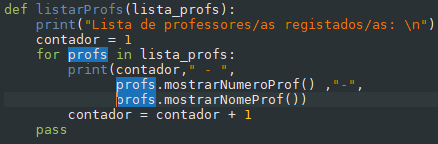


Figura 18 – Função ‘listarProfs’

## 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.

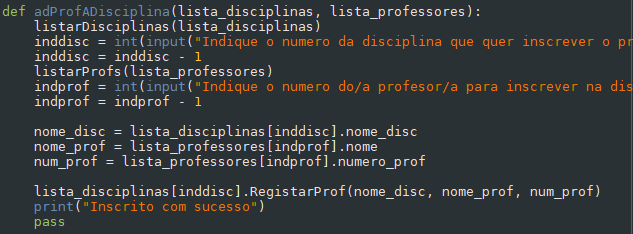


Figura 19 - Função 'adProfADisciplina'

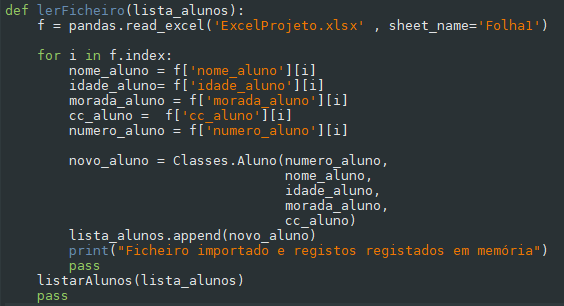
## Função ‘lerFicheiro’

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*)



# 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.

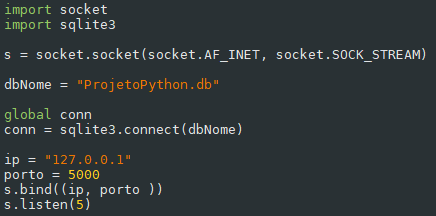


Figura 20 - Estruturar o ficheiro 'Servidor.py'

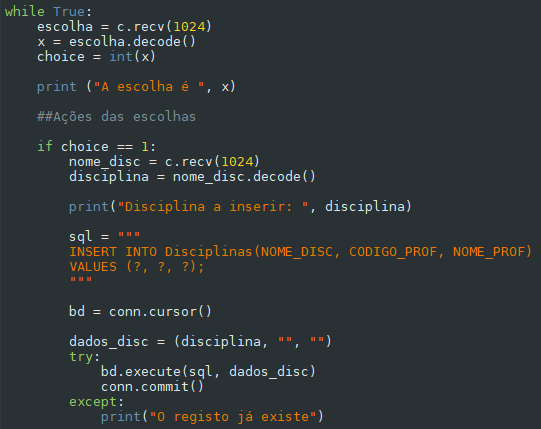


Figura 21 - Exemplo de código SQL embutido no ficheiro de python 'Servidor.py'

# 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