
**Projeto Final Módulo 15 (Sistema de Gestão de clientes)**

Iúri Branco Casas Novas, Pedro Fontana Moretti

Curso Técnico de Gestão e Programação de Sistemas Informáticos

Escola Profissional Bento de Jesus Caraça, Delegação do Barreiro

P.S.I.: Programação e Sistemas de Informação

Professor Diogo Oliveira

5 de fevereiro de 2025

## Introdução

O projeto foi desenvolvido no âmbito da disciplina de PSI (Programação de Sistemas de Informação), a pedido do professor Diogo Oliveira. O principal objetivo foi permitir que os conhecimentos adquiridos nos módulos anteriores fossem aplicados pelos alunos.

Além disso, o trabalho deveria seguir os princípios do modelo CRUD (*Create, Read, Update, Delete*), garantindo que uma estrutura organizada e orientada fosse mantida, tal como nos exercícios realizados nas aulas anteriores.

## Planeamento do Projeto Antes da Implementação

Para começarmos, precisamos de um tema, após fazermos um *brainstorming*, definimos que o tema seria um programa para gerir os clientes de um ginásio, seguindo o modelo CRUD. O início do trabalho não foi feito pelo *Visual Studio Code*, mas sim pelo Bloco de Notas, para planear como cada função iria interagir entre si e como o utilizador poderia interagir com o programa. O objetivo foi garantir que fosse possível registar clientes, atualizar dados, obter registos e apagar clientes, respeitando assim o modelo CRUD *(Create, Read, Update, Delete)*.

Na primeira hora, foi discutido como o programa seria estruturado antes de ser desenvolvido no *Visual Studio Code*.

## Implementação do Código

Após a discussão sobre a estrutura do código, passou-se então para a implementação, começando pela criação das pastas *(create, read, update, delete)* e dos ficheiros numa estrutura de repositório no *GitHub*. Foi criado um ficheiro *main.py*, onde todas as funções seriam executadas, e o ficheiro *README.md*, onde mais tarde seria escrita uma explicação sobre como o sistema se iria estruturar. Este ficheiro serviria para que futuros "colaboradores" (ou, no nosso caso, o professor) pudessem perceber o código sem precisar de ler linha por linha, bastando para isso consultar o *README.md*.

Foi criado, então, o create.py, onde foi utilizada a query *CREATE TABLE IF NOT EXISTS* para criar a tabela, que no programa recebeu o nome de clientes, com as colunas necessárias, sendo o id autoincrementável. As colunas incluíram nome, idade, peso, altura e imc.

Posteriormente, foi também criado na mesma pasta "*create*" o *insert.py*. Neste ficheiro, foi implementada a função inserir_cliente, na qual foi passada como parâmetro a conexão à base de dados, que mais tarde foi fornecida como argumento no ficheiro *main.py*. Dentro desta função, foram utilizadas cinco variáveis, cada uma com um input para que o utilizador preenchesse os valores correspondentes. Em seguida, foi executada a query *INSERT INTO* clientes, utilizando placeholders para os valores de nome, idade, peso, altura e imc, a fim de reduzir as hipóteses de SQL injections.

O valor de imc foi calculado diretamente dentro da função, fazendo uma simples conta e guardando o resultado na variável imc.

Na pasta Read, foram criados os ficheiros *select.py*, *grau.py* e *plano.py*, que mantêm a lógica do create, utilizando funções para garantir a reutilização do código em outras partes, como no ficheiro *main.py*. Na parte de seleção, foi implementada uma função do próprio Python *fetchall()*, seguida de um print para mostrar todos os dados da tabela na consola.

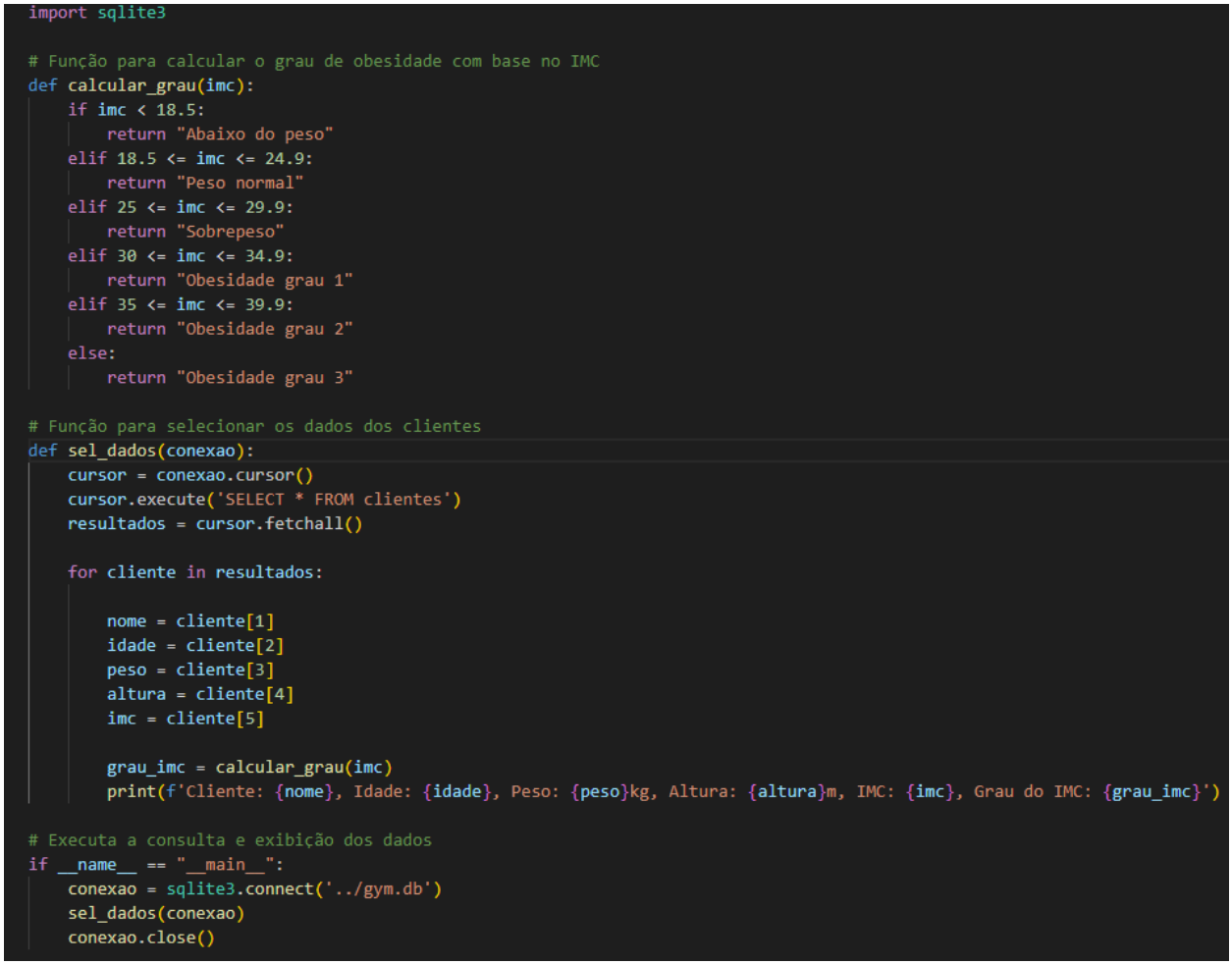
Também foi criado o ficheiro *grau.py*, onde foi realizada uma verificação de todos os IMC dos clientes e atribuída uma classe estrutural de massa corporal, para ser mais fácil criar planos específicos para as necessidades de cada cliente.

No ficheiro *plano.py*, foi utilizado o IMC e um sistema de verificação. Se o IMC fosse demasiado baixo, a pessoa receberia uma mensagem informando que "o treino tem de ser para ganhar massa", e o processo continuaria sucessivamente conforme as diferentes faixas de IMC.

## *Imagens*

**Figura 1**

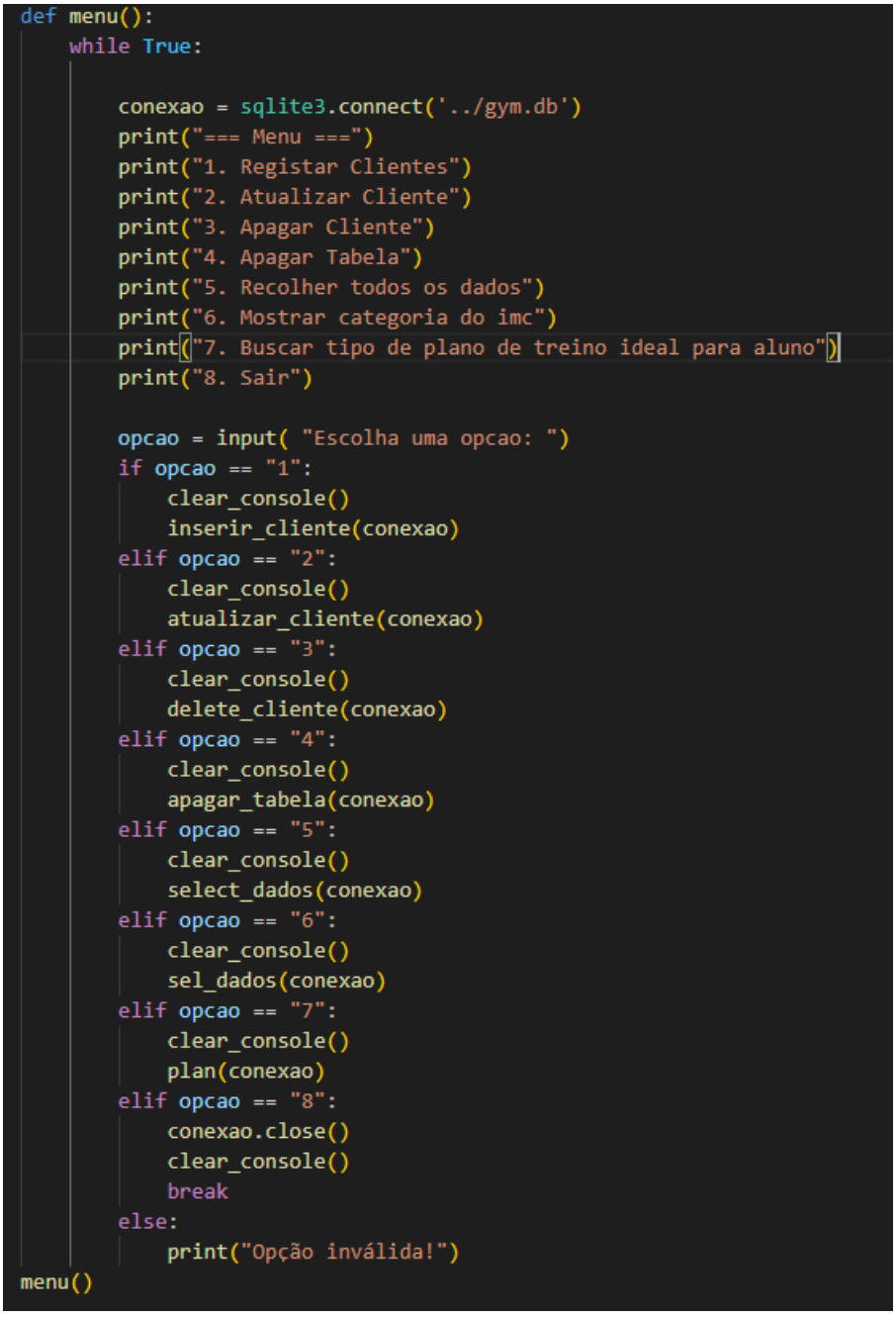
Código do ficheiro *grau.py*



```python
import sqlite3

# Função para calcular o grau de obesidade com base no IMC
def calcular_grau(imc):
    if imc < 18.5:
        return "Abaixo do peso"
    elif 18.5 <= imc <= 24.9:
        return "Peso normal"
    elif 25 <= imc <= 29.9:
        return "Sobrepeso"
    elif 30 <= imc <= 34.9:
        return "Obesidade grau 1"
    elif 35 <= imc <= 39.9:
        return "Obesidade grau 2"
    else:
        return "Obesidade grau 3"

# Função para selecionar os dados dos clientes
def sel_dados(conexao):
    cursor = conexao.cursor()
    cursor.execute('SELECT * FROM clientes')
    resultados = cursor.fetchall()

    for cliente in resultados:

        nome = cliente[1]
        idade = cliente[2]
        peso = cliente[3]
        altura = cliente[4]
        imc = cliente[5]

        grau_imc = calcular_grau(imc)
        print(f'Cliente: {nome}, Idade: {idade}, Peso: {peso}kg, Altura: {altura}m, IMC: {imc}, Grau do IMC: {grau_imc}')

# Executa a consulta e exibição dos dados
if __name__ == "__main__":
    conexao = sqlite3.connect('../gym.db')
    sel_dados(conexao)
    conexao.close()
```

**Figura 2**

Parte do ficheiro *main.py* com código do menu



```python
def menu():
    while True:

        conexao = sqlite3.connect('../gym.db')
        print("=== Menu ===")
        print("1. Registar Clientes")
        print("2. Atualizar Cliente")
        print("3. Apagar Cliente")
        print("4. Apagar Tabela")
        print("5. Recolher todos os dados")
        print("6. Mostrar categoria do imc")
        print("7. Buscar tipo de plano de treino ideal para aluno")
        print("8. Sair")

        opcao = input( "Escolha uma opcao: ")
        if opcao == "1":
            clear_console()
            inserir_cliente(conexao)
        elif opcao == "2":
            clear_console()
            atualizar_cliente(conexao)
        elif opcao == "3":
            clear_console()
            delete_cliente(conexao)
        elif opcao == "4":
            clear_console()
            apagar_tabela(conexao)
        elif opcao == "5":
            clear_console()
            select_dados(conexao)
        elif opcao == "6":
            clear_console()
            sel_dados(conexao)
        elif opcao == "7":
            clear_console()
            plan(conexao)
        elif opcao == "8":
            conexao.close()
            clear_console()
            break
        else:
            print("Opção inválida!")
menu()
```

***Texto corrido***

O nosso projeto foi desenvolvido para criar uma aplicação capaz de gerir um ginásio e os seus clientes. Para se registarem, os utilizadores devem inserir o seu nome, idade, altura e peso, dados que serão armazenados na base de dados *gym*.db. Além do registo, a aplicação permite apagar clientes, remover tabelas, atualizar registos, listar todos os dados, calcular a categoria do IMC e sugerir um plano de treino adequado com base nesse índice. Caso nenhuma destas opções seja selecionada, será apresentada uma mensagem de erro.

O programa está estruturado em vários ficheiros, todos com a extensão .py, utilizada para scripts em Python. Entre eles, temos:

*create.py*, responsável por criar a tabela e estabelecer a conexão com a base de dados;

*insert.py*, utilizado para registar novos clientes;

*delete.py*, que remove clientes da base de dados;

*drop.py*, para excluir uma tabela por completo;

*grau.py*, que calcula o grau do IMC;

*plano.py*, que define o plano de treino ideal com base no IMC do cliente;

*select.py*, que apresenta todos os clientes registados;

*main.py*, que funciona como o menu principal da aplicação.

## Conclusão

O desenvolvimento deste projeto permitiu-nos aprofundar os conhecimentos adquiridos ao longo do curso, especialmente na implementação do modelo CRUD em um ambiente prático. Além de aprimorar as nossas habilidades técnicas em Python e no uso de bases de dados, também desenvolvemos competências interpessoais, como o trabalho em equipe, a organização e a resolução de problemas.

A criação do sistema de gestão para um ginásio demonstrou a importância da estruturação do código e da reutilização de funções para otimizar o desempenho e a manutenção do programa. Além disso, percebemos como a programação pode ser aplicada para resolver problemas reais e facilitar a gestão de informações de forma eficiente.

Durante o desenvolvimento do projeto não tivemos dificuldade nenhuma, como já sabíamos como utilizar as operações CRUD que foi o pedido para o projeto, o único pormenor que discutimos por mais tempo foi o tema do projeto que acabou por ser um programa para gerir os clientes de um ginásio.

Por fim, este projeto reforçou a nossa capacidade de planeamento, implementação e validação de soluções.