Escola Profissional
BENTO DE JESUS CARAÇA
DELEGAÇÃO DE LISBOA

PROJETO DE ESPECIFICAÇÃO TÉCNICO DE UM SISTEMA DE COMUNICÁÇÃO

Projeto final M9 de PSI

Professor Breno Sousa

Nome de aluno

Nº de aluno

_____/_____/_____

O Relatório encontra se em condições para ser apresentado

Ciclo de formação 2024/2026

Ano letivo 2024/2025

Yasin Alizadah

Marcos pires

Rafael Carvalho

Escola Profissional
BENTO DE JESUS CARAÇA
DELEGAÇÃO DE LISBOA

INDICE

Introdução

Neste relatório, tenho como objetivo demonstrar o nosso código de python que fizemos durantes as aulas de PSI, para fazer um sistema simples para gerenciar o empréstimo de livros da sua biblioteca.

A biblioteca precisa das seguintes partes:

- Classes: livro, aluno, biblioteca
- Livro: título, autor, código do livro, disponibilidade (emprestado ou não)
- Aluno: nome, matrícula, lista de livros emprestados
- Biblioteca: catálogo de livros, cadastro de alunos

Com as seguintes funcionalidades:

- Cadastrar novos livros e alunos.
- Consultar livros disponíveis.
- Fazer empréstimo de livros (associar livro a aluno).
- Devolver livros.
- Mostrar os livros emprestados por um aluno.

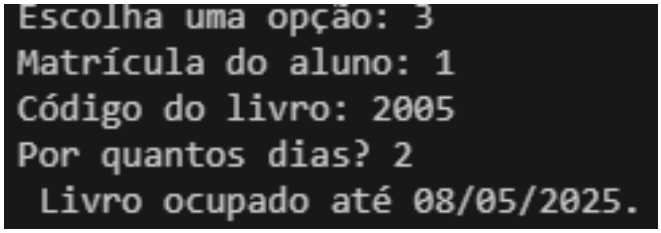
Introdução à explicação do código

Nas seguintes páginas iremos explicar a estrutura do nosso código de forma clara, que foi desenvolvida ao longo das aulas de PSI, incluindo todas as classes, todas as funções, todos os objetos e o nosso menu.

```
from datetime import datetime, timedelta  # Importa bibliotecas para manipular datas
```

No início do código, usamos o modulo "datetime" para trabalhar com datas como objetos de data, mesmo não sendo um tipo de data próprio.

Neste caso o modulo de "datatime" irá anotar a data exata na qual cada ação foi realizada, como por exemplo indicar até que dia específico um aluno estará sobre a posse de um livro emprestado, como se pode verificar na imagem abaixo após realizar a função de emprestar um livro.



Desenvolvimento da explicação do código (pt 1)

O método __init__ é o método construtor da classe. Ele é chamado automaticamente ao criar um novo objeto da classe Livro. O título, autor e código são os dados passados na criação do livro.

Aqui introduzimos uma classe de livro, a linha self.disponivel = true verifica se o livro em questão está disponível para fazer o empréstimo para os alunos, O método __str__ define como o objeto Livro será representado como texto. Se disponível for True, mostra Disponível ou Emprestado, e retorna uma string formatada com código, título, autor e status.

```
class Livro:
    def __init__(self, titulo, autor, codigo):
        self.titulo = titulo
        self.autor = autor
        self.codigo = codigo
        self.disponivel = True

    def __str__(self):
        status = "Disponível" if self.disponivel else "Emprestado"
        return f"{self.codigo}: {self.titulo}, {self.autor} ({status})"
```

O método __init__ é o método construtor da classe como já referido anteriormente, este método é chamado automaticamente ao criar um novo objeto da classe Aluno. O nome e matrícula são os dados passados na criação do aluno. A linha começada pelo método __str__ serve para personalizar como o objeto será mostrado como texto, quando se usa print (objeto). Quando se imprime um objeto dessa classe ele retorna a string formatada com a matrícula e o nome. A função f"{self.matricula}: {self.nome}" cria uma string com esses dois atributos.

```
class Aluno:
    def __init__(self, nome, matricula):
        self.nome = nome
        self.matricula = matricula
        self.livros_emprestados = {}

    def __str__(self):
        return f"{self.matricula}: {self.nome}"
```

Continuação da explicação do código (pt 2)

O método __init__ é o método construtor da classe, este método é chamado automaticamente ao criar um novo objeto da classe Biblioteca.

```
class Biblioteca:
    def __init__(self):
        self.catalogo = {}
        self.alunos = {}
```

Esta parte do código abaixo é um método chamado "cadastrar_livro", da classe livro. Nós definimos o método "cadastrar_livro" com os parâmetros: título (nome do livro), autor (quem escreveu) e código (código do livro).

O "self" indica que este método pertence a classe livro, e ele acessa os atributos e métodos dessa classe.

Verifica se o código do livro já está presente no dicionário "self.catalogo". Se já estiver, o sistema informa que o livro está registado e sai do método com "return".

Se o código ainda não estiver no catálogo, cria uma nova instância da classe "Livro" e a adiciona ao catálogo, usando o codigo como chave.

```
def cadastrar_livro(self, titulo, autor, codigo):
    if codigo in self.catalogo:
        print("Livro já registrado.")
        return
    self.catalogo[codigo] = Livro(titulo, autor, codigo)
    print("Livro registrado com sucesso!")
```

Define um método chamado cadastrar_aluno que pertence a uma classe (por isso tem o self como primeiro parâmetro).

Esse método recebe dois argumentos além do self: nome e matrícula, que são informações do aluno a ser cadastrado.

if matrícula in self.alunos, verifica se a matrícula passada já existe dentro do dicionário self.alunos. self.alunos é um dicionário que armazena os alunos cadastrados, usando a matrícula como chave. Se a matrícula já existir (ou seja, o aluno já está cadastrado), imprime uma mensagem informando isso.

Depois, o return interrompe a execução do método para evitar que o mesmo aluno seja cadastrado duas vezes. Se a matrícula não estiver no dicionário, cria um novo objeto Aluno com o nome e matrícula fornecidos.

Esse novo objeto Aluno é então armazenado no dicionário self.alunos, usando a matrícula como chave.

```
def cadastrar_aluno(self, nome, matricula):
    if matricula in self.alunos:
        print(" Aluno já registrado.")
        return
    self.alunos[matricula] = Aluno(nome, matricula)
    print(" Aluno registrado com sucesso!")
```

Continuação da explicação do código (pt 4)

Este método emprestar_livro permite registar o empréstimo de um livro a um aluno. Primeiro, verifica se o aluno está cadastrado no sistema (usando o número da matrícula) e se o livro (identificado pelo código) existe no catálogo.

Se algum deles não estiver, uma mensagem de erro é exibida e o processo é interrompido.

Em seguida, verifica se o livro está disponível, se não estiver, informa que já está emprestado.

Caso tudo esteja em ordem, o livro é marcado como indisponível (disponível = False) e a data de entrega é calculada somando os dias informados à data atual.

Por fim, essa data é regista no dicionário de livros emprestados do aluno, associada ao código do livro, e uma mensagem com a data de devolução prevista é mostrada ao utilizador.

```
def emprestar_livro(self, matricula, codigo, dias):
    if matricula not in self.alunos:
        print("O aluno não se encontra no sistema.")
        return
    if codigo not in self.catalogo:
        print("O Livro não se encontra no sistema.")
        return

    livro = self.catalogo[codigo]
    aluno = self.alunos[matricula]

    if not livro.disponivel:
        print("O Livro já se encontra ocupado.")
        return

    livro.disponivel = False
    data_entrega = datetime.now() + timedelta(days=dias)
    aluno.livros_emprestados[codigo] = data_entrega
    print(f" Livro ocupado até {data_entrega.strftime('%d/%m/%Y')}.")
```

Continuação da explicação do código (pt 5)

Este código define o método devolver livro, que gerência a devolução de um livro emprestado por um aluno em um sistema de biblioteca.

Primeiro, ele verifica se a matrícula informada está registada no sistema (self.alunos); se não estiver, exibe uma mensagem e encerra a função.

Em seguida, verifica se o aluno com essa matrícula possui o livro identificado pelo código fornecido, se não o tiver, também exibe uma mensagem e retorna.

Caso contrário, o método marca o livro como disponível (livro.disponivel = True), remove o livro da lista de livros emprestados do aluno (del aluno.livros_emprestados[codigo]) e imprime uma confirmação de devolução.

```
def devolver_livro(self, matricula, codigo):
    if matricula not in self.alunos:
        print("O aluno não se encontra no sistema.")
        return
    aluno = self.alunos[matricula]
    if codigo not in aluno.livros_emprestados:
        print("Este aluno não está na posse do livro.")
        return

    livro = self.catalogo[codigo]
    livro.disponivel = True
    del aluno.livros_emprestados[codigo]
    print("Livro devolvido com sucesso!")
```

Continuação da explicação do código (pt 6)

A seguinte parte do código define métodos para listar informações importantes, como livros e alunos. O método listar_livros_disponiveis percorre o dicionário catálogo e imprime apenas os livros cujo atributo disponível é verdadeiro, ou seja, que não estão emprestados.

O método listar_catalogo imprime todos os livros do catálogo, independentemente de estarem disponíveis ou não, ambos usam self.catalogo.values() para acessar os livros.

Os outros dois métodos focam nos alunos, o método listar_alunos imprime todos os alunos registados no sistema acessando self.alunos.values(),

Por fim, listar_livros_por_aluno verifica se a matrícula informada está no sistema e, se estiver, lista os livros que o aluno pediu emprestado, mostrando o título e a data prevista para devolução formatada. Este método acessa um dicionário livros_emprestados que relaciona códigos de livros às datas de devolução, e usa o código para recuperar o livro correspondente no catálogo.

```
def listar_livros_disponiveis(self):
    print("\n Livros Disponíveis:")
    for livro in self.catalogo.values():
        if livro.disponivel:
            print(livro)

def listar_catalogo(self):
    print("\n Catálogo de Livros:")
    for livro in self.catalogo.values():
        print(livro)

def listar_alunos(self):
    print("\n Alunos Registrados:")
    for aluno in self.alunos.values():
        print(aluno)

def listar_livros_por_aluno(self, matricula):
    if matricula not in self.alunos:
        print("O Aluno não se encontra no sistema.")
        return
    aluno = self.alunos[matricula]
    print(f"\n Livros com {aluno.nome}:")
    for codigo, data in aluno.livros_emprestados.items():
        livro = self.catalogo[codigo]
        print(f"{livro.titulo} - Devolver até {data.strftime('%d/%m/%Y')}")
```

Esta parte do código define a função menu(), que cria uma instância da classe Biblioteca e mostra de forma repetitiva um menu com opções da biblioteca da escola.

O menu apresenta opções para registar alunos e livros, emprestar e devolver livros, além de visualizar informações como livros disponíveis, livros emprestados por um aluno, alunos registados e o catálogo completo. O loop while True garante que o menu continue sendo exibido até que o usuário escolha a opção "0" (Sair).

```
def menu():
    biblioteca = Biblioteca()

    while True:
        print("\n BIBLIOTECA ESCOLAR ")
        print("1. Registrar Aluno")
        print("2. Registrar Livro")
        print("3. Emprestar Livro")
        print("4. Devolver Livro")
        print("5. Ver Livros Disponíveis")
        print("6. Ver Livros de um Aluno")
        print("7. Ver Alunos Registrados")
        print("8. Ver Catálogo de Livros")
        print("0. Sair")

        opcao = input("Escolha uma opção: ")
```

Fim da explicação do código

A parte final do código corresponde ao menu de gestão da biblioteca.

Através da variável opcao, o utilizador pode escolher uma das opções disponíveis, em que cada número representa uma operação diferente, como cadastrar alunos ("1"), cadastrar livros ("2"), emprestar ("3") ou devolver livros ("4"), ver livros disponiveis (“5”), ver livros de um aluno (“6”), ver alunos registrados (“7”), ver catálogo de livros (“8”) e sair (“0”). Quando o utilizador introduz uma opção, o programa solicita os dados necessários e chama os métodos correspondentes da biblioteca, input() é utilizado para recolher informações do utilizador, e em alguns casos, o número de dias de empréstimo, o valor é convertido para inteiro com int().

Por fim, se a opção introduzida não corresponder a nenhuma das disponíveis, o programa apresenta uma mensagem de erro.

```
if opcao == "1":
    nome = input("Nome do aluno: ")
    matricula = input("Matrícula: ")
    biblioteca.cadastrar_aluno(nome, matricula)

elif opcao == "2":
    titulo = input("Título do livro: ")
    autor = input("Autor do livro: ")
    codigo = input("Código do livro: ")
    biblioteca.cadastrar_livro(titulo, autor, codigo)

elif opcao == "3":
    matricula = input("Matrícula do aluno: ")
    codigo = input("Código do livro: ")
    dias = int(input("Por quantos dias? "))
    biblioteca.emprestar_livro(matricula, codigo, dias)

elif opcao == "4":
    matricula = input("Matrícula do aluno: ")
    codigo = input("Código do livro: ")
    biblioteca.devolver_livro(matricula, codigo)

elif opcao == "5":
    biblioteca.listar_livros_disponiveis()
```

```
        elif opcao == "6":
            matricula = input("Matrícula do aluno: ")
            biblioteca.listar_livros_por_aluno(matricula)

        elif opcao == "7":
            biblioteca.listar_alunos()

        elif opcao == "8":
            biblioteca.listar_catalogo()

        elif opcao == "0":
            print("Saindo do sistema...")
            break

        else:
            print("Opção não disponivel!")

menu()
```

Conclusão

Em conclusão, este trabalho apresenta um sistema simples mas funcional de gestão de biblioteca, desenvolvido em Python, que permite realizar operações fundamentais como o registo de alunos e livros, a gestão de empréstimos e devoluções.

Através de um menu interativo o programa oferece uma interface acessível ao utilizador, demonstrando a aplicação da programação a objetos e estruturas de controlo.

Este projeto contribui para o desenvolvimento de competências em lógica de programação e organização de dados, sendo uma base sólida para futuras melhorias e expansões do sistema.