Guião de apoio 6
Aplicações no estilo arquitetural *REST* com Flask e *SQLite*

---

1. Introdução ao tema

Neste guião, exploraremos o estilo arquitetural *REST* que foi abordado nas últimas aulas. Será implementada uma aplicação através da *framework* de desenvolvimento Web *Flask* e do motor de base de dados *SQLite*.

2. Introdução ao *SQLite*

O motor de base de dados *SQLite* permite a utilização de uma base de dados baseada em SQL que não necessita de um processo servidor como outros sistemas de gestão de base de dados. Com o *SQLite* a base de dados pode ser criada num ficheiro em disco ou em memória, ficando acessível às aplicações numa dessas formas. A Listagem 1 exemplifica o uso de *SQLite*.

Listagem 1 – Exemplo de utilização de *SQLite* num programa Python.

```
import sqlite3
from os.path import isfile

def connect_db(dbname):
       db_is_created = isfile(dbname) # Existe ficheiro da base de dados?
       connection = sqlite3.connect('notas.db')
       cursor = connection.cursor()
       if not db_is_created:
              cursor.execute("CREATE TABLE notas (numero_aluno INTEGER, ano TEXT,\
                           cadeira TEXT, nota INTEGER);")
              connection.commit()
       return connection, cursor

um_registo =         (123, '2021/2022', 'AD', 20)
varios_registos=[    (1000,'2021/2022','AD',10),
                     (1000,'2021/2022','ITW',10),
                     (1001,'2021/2022','AD',17),
                     (1000,'2021/2022','ITW',17)]

if __name__ == '__main__':
       conn, cursor = connect_db('notas.db')
       cursor.execute('INSERT INTO notas VALUES (?, ?, ?, ?)', um_registo)
       conn.commit()
       cursor.executemany('INSERT INTO notas VALUES (?, ?, ?, ?)', varios_registos)
       conn.commit()
       cursor.execute('SELECT * FROM notas') # Fazer query e obter todos
       todos = cursor.fetchall()             # os resultados
       print ("Todos: ", todos)
       cursor.execute('SELECT * FROM notas') # Fazer query e obter um a um
       registo = cursor.fetchone()
       while registo:
              print ("Mais um: ", registo)
              registo = cursor.fetchone()
       cursor.execute('SELECT * FROM notas') # Fazer query e obter em grupos
       registos = cursor.fetchmany(size = 2)
       while registos:
              print ("Grupo: ",registos)
              registos = cursor.fetchmany(size = 2)
       conn.close()
```

3. Exemplo base com *Flask*

A Listagem 2 mostra um exemplo base escrito com recurso a Flask, de uma aplicação que pretende registar alunos bem como as notas que estes obtêm em disciplinas. O exemplo implementa a interação entre cliente e servidor, mas não persiste os dados de nenhuma forma. Para inserir alunos deverá ser feito um PUT na URL /aluno com a informação relevante (número, nome, idade). Nesse caso o servidor deverá responder com a localização (cláusula location) do novo recurso criado (aluno). Para obter informação sobre um aluno deverá ser feito um GET na URL /aluno/<id>, onde <id> é o número de aluno. O Servidor deverá responder com os dados referentes ao aluno pretendido.

Listagem 2 – Exemplo de utilização da *framework* Flask.

```
from flask import Flask, request, make_response

app = Flask(__name__)

@app.route('/aluno', methods = ["PUT"])
@app.route('/aluno/<int:id>', methods = ["GET"])
def aluno(id = None):
	if request.method == "GET":
		# Ler dados do aluno com id na base de dados
		r = make_response('Dados do aluno %d' % id)
		r.status_code = 200
		return r
	if request.method == "PUT":
		# Ler dados do aluno no pedido e inserir na base de dados
		# Em caso de sucesso responder com a localização do novo recurso
		r = make_response()
		r.headers['location'] = '/alunos/123' # 123 para exemplo
		return r

@app.route('/notas', methods = ["POST", "GET"])
def notas():
	if request.method == "POST":
		#ler dados no pedido e inserir na base de dados
		r = make_response()
		return r
	if request.method == "GET":
		#ler campos no pedido e fazer query de acordo
		r = make_response(request.data) #Devolve os dados no pedido
		return r

if __name__ == '__main__':
	app.run(debug = True)
```

Para lançar notas de um aluno deverá ser feito um POST na URL /notas com os dados relevantes. Para pesquisar notas deverá ser feito um GET na URL /notas com os dados da pesquisa que se pretende: por exemplo, ano='2022' e cadeira='AD', resultaria em todos os registos de AD em 2022/2023.

A Listagem 3 exemplifica um programa cliente que interage com o servidor acima.

Listagem 3 – Exemplo de um cliente para a Listagem 2 utilizando o módulo *requests*.

```
import requests
import json

r = requests.get('http://localhost:5000/aluno/25')
print (r.status_code)
print (r.content.decode())
print (r.headers)
print ('***')

dados = {'numero': 123, 'nome': 'Carabino Tiro Certo', 'idade': 18}
r = requests.put('http://localhost:5000/aluno', json = dados)
print (r.status_code)
print (r.content.decode())
print (r.headers)
print ('***')

notas = {'numero_aluno': 123, 'ano': '1988/1989', 'cadeira': 'AD', 'nota': 20}

r = requests.post('http://localhost:5000/notas', json = notas)
print (r.status_code)
print (r.content.decode())
print (r.headers)
print ('***')

pesquisa = {'ano': '1988/1989', 'cadeira': 'AD'}
r = requests.get('http://localhost:5000/notas', json = pesquisa)
print (r.status_code)
print (r.content.decode())
print (r.headers)
print ('***')
```

4. Exercícios

1. Copie o programa apresentado na Listagem 1 e execute-o. Apoiando-se na documentação sobre o módulo *Python* sqlite3 deverá entender o papel de algumas funções disponíveis nas classes disponibilizadas pelo módulo: *connect, cursor, commit, close, execute, executemany, fetchone, fetchmany, fetchall*. Para além destas deverá perceber a utilidade da função *executescript* dos objetos *cursor*.

2. Copie e execute os programas nas Listagem 2 e Listagem 3. Verifique como funciona a interação entre o cliente e o servidor e como se pode neste último aceder aos dados do pedido e construir uma resposta.

3. Altere os programas das Listagem 2 e Listagem 3 da forma que achar necessária para que toda a troca de dados seja feita em *JSON* e para que os dados sejam guardados numa base de dados em *SQLite*. Deverá alterar a base de dados apresentada na Listagem 1, pesquisar como integrar *SQLite* no *Flask*, e procurar a forma de ler *JSON* num pedido Flask.

5. Bibliografia e outro material de apoio

- Flask Quickstart: https://flask.palletsprojects.com/en/2.1.x/quickstart/
- Flask API: https://flask.palletsprojects.com/en/2.1.x/api/
- Módulo requests: http://docs.python-requests.org/en/master/
- SQLite: https://www.sqlite.org/docs.html