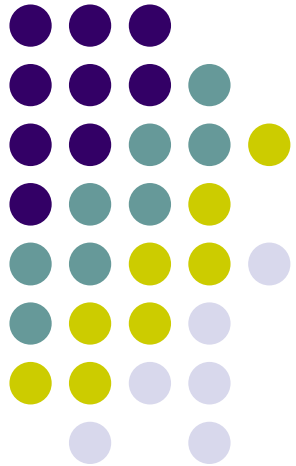
Programação Orientada a Objetos (POO)

TEsP Tecnologias Informáticas (TI)

UAlg ISE
UNIVERSIDADE DO ALGARVE
INSTITUTO SUPERIOR DE ENGENHARIA



Serviços WEB

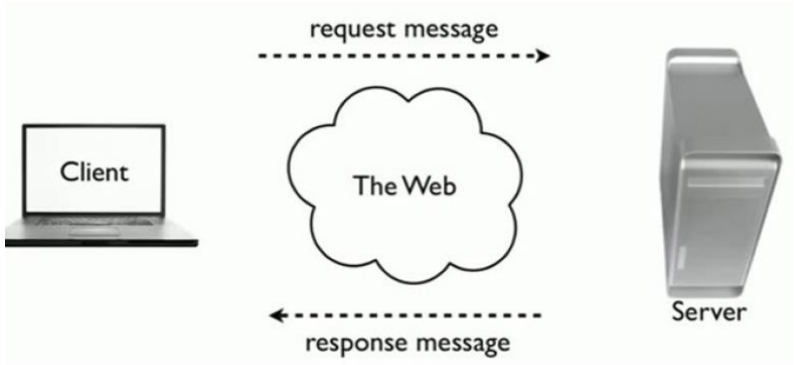
RESUMO

- Introdução à programação de aplicações web.
- Serviços
- WS RESTful.
- Flask.

Introdução a aplicações WEB

- Na prática um Web Service é constituído por um ou mais métodos que podem ser acedidos ou invocados por outras aplicações (móveis, para desktop ou para a web), utilizando tecnologias e protocolos Web.

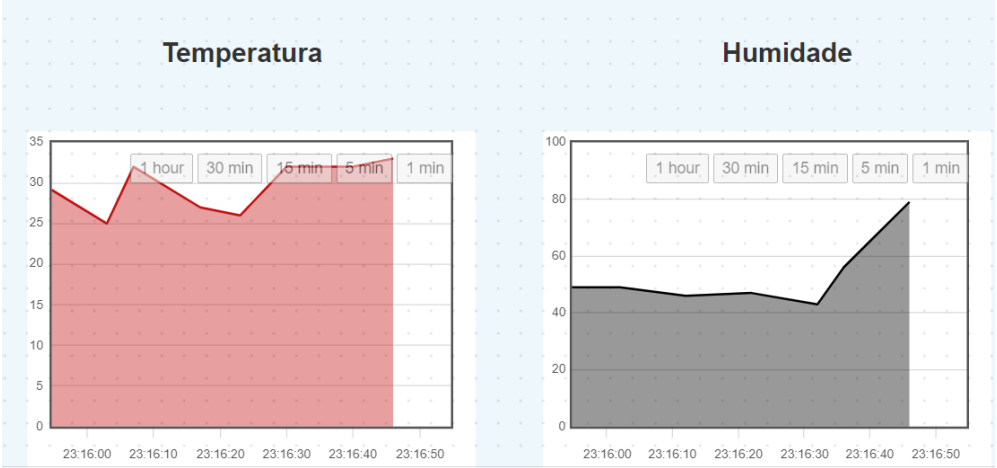
- Os Web Services (ilustração abaixo) são muito úteis quando se tenciona desenvolver serviços e aplicações de grande escala/distribuídas



Exemplo:

- Imagine possuir um Raspberry/Arduino que tem acoplado um sensor de temperatura que faz medições em intervalos de 5 minutos. Além disso, o Raspberry também pode fazer uma leitura mediante um pedido pontual.

- **Objetivo**: a partir de qualquer dispositivo e, independentemente do sistema operativo, conseguir aceder a essa informação de uma forma transparente.



Acesso online: http://193.136.227.157/emoncms/dashboard/edit&id=69

- ***Exemplo***:

  - **Solução:**
    - No Raspberry constrói-se e disponibiliza-se um Web Service (não interessa em que linguagem de programação foi programado) que tem duas funções:
      - informar a temperatura atual
      - informar a média da temperatura para o dia em causa. Os clientes podem ser uma app para Android, para iOS, uma simples página web, uma aplicação para Desktop, etc.
  - Feito o pedido do cliente, o serviço Web processa e envia os dados solicitados sendo que as comunicações têm como base o protocolo HTTP/HTTPS para proceder à transferência/transporte de informação.

VANTAGENS:

- **Funcionam** nos mais **diversos sistemas operativos**, nos mais diversos tipos de hardware e são bastante flexíveis
- Cada aplicação pode ter a sua **própria linguagem** de programação, que é traduzida para uma linguagem universal, um formato intermediário como XML, JSON, CSV, etc.
- **Simplicidade** na interoperabilidade de sistemas
- Simples de implementar, com a vantagem de poderem ser construídos em várias linguagens de programação

VANTAGENS (cont):

- Segurança, uma vez que não há uma acesso direto à informação (por exemplo a informação que está na base de dados).

- Redução de custos, um Web Service pode servir vários tipos de aplicações/serviços (reutilização de código)

- **O acesso à informação é realizado através de uma API.**

  - **Exemplos de API públicas (algumas requerem subscrição):**
    - https://rapidapi.com/
      - https://rapidapi.com/collection/list-of-free-apis
    - https://apilist.fun/
      - https://apilist.fun/api/order-pizza-api
    - https://github.com/javieraviles/covidAPI

WS Restful

- A Representational State Transfer (REST) é um estilo de arquitetura que define um conjunto de restrições e propriedades baseados em HTTP usadas para a criação de serviços Web.

- Os Web Services RESTful são identificados por um URI (Uniform Resource Identifier)

- Web Services que obedecem ao estilo arquitetural REST, ou web services RESTful, fornecem interoperabilidade entre sistemas de computadores na Internet.

- Num Web service RESTful, as solicitações feitas ao URI de um recurso provocará uma resposta com uma carga útil (payload) formatada em HTML, XML, JSON ou algum outro formato.

WS Restful

- Utilizam um protocolo cliente/servidor sem estado (*stateless*): cada mensagem HTTP contém toda a informação necessária para compreender o pedido.
  - Como resultado, nem o cliente e nem o servidor necessitam gravar nenhum estado das comunicações entre mensagens.

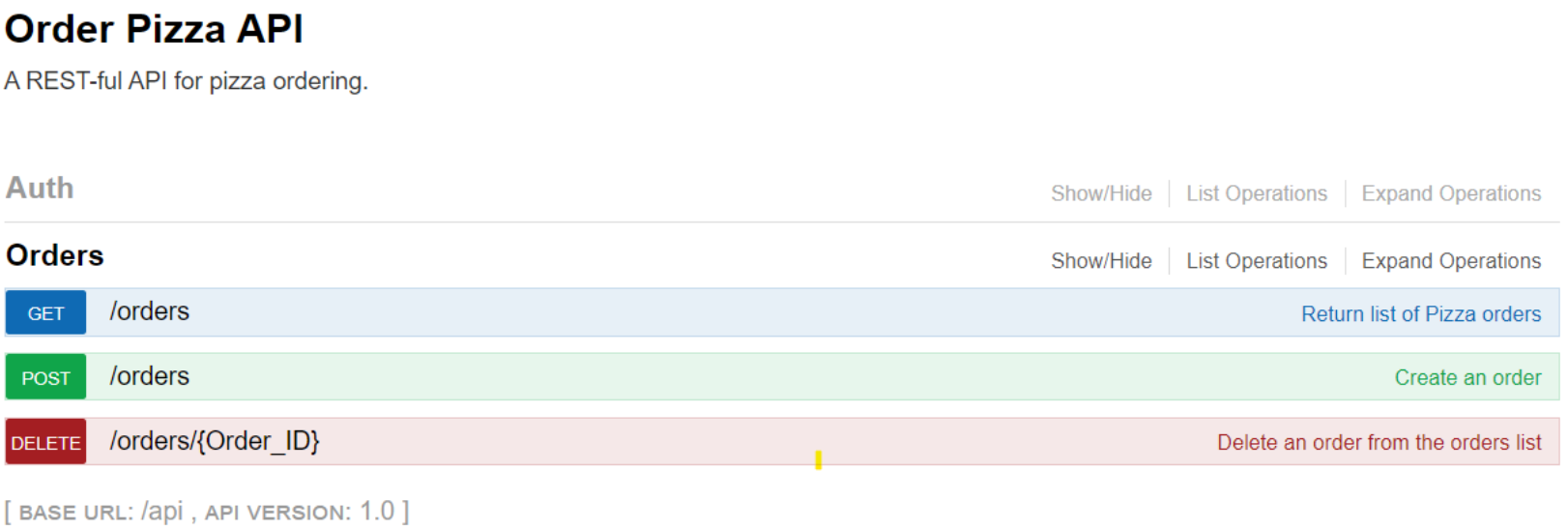
- Um conjunto de operações bem definidas que se aplicam a todos os recursos de informação:
  - HTTP em si define um pequeno conjunto de operações, as mais importantes/comuns são **POST, GET, PUT e DELETE.**

WS Restful

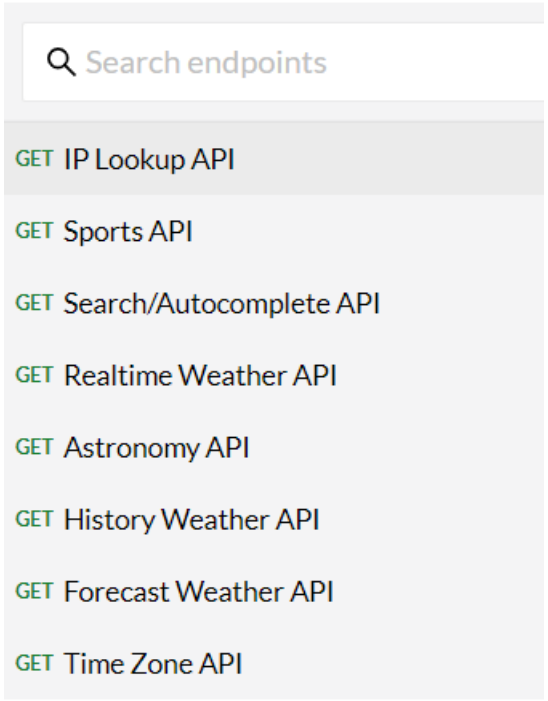
- Relativamente aos métodos HTTP mais usados numa arquitetura REST, destaque para:
  - GET - Acesso apenas de leitura a um recurso
  - PUT - Atualizar ou criar um novo recurso
  - DELETE - Remover um recurso
  - POST - Para criar um novo recurso

WS Restful

- **Exemplo de aplicação (listagem, encomenda e remoção de pedidos de pizza)**
  - https://order-pizza-api.herokuapp.com/api/ui/?ref=apilist.fun#/Orders



- Aceda a https://rapidapi.com/weatherapi/api/weatherapi-com/
- Se necessário faça registo (gratuito)



Opções de API

Código Python para utilizar serviço RESTful:

Determinar a localização de IP de PC

```
import requests

url = "https://weatherapi-com.p.rapidapi.com/ip.json"

querystring = {"q":"193.136.227.157"}

headers = {
   'x-rapidapi-host': "weatherapi-com.p.rapidapi.com",
   'x-rapidapi-key': "ec9b161860msh650e03cd3a28230p1af29ejsncefed8e6f5e9"
   }

response = requests.request("GET", url, headers=headers, params=querystring)

resposta=response.json()
print(f'O PC com IP {resposta["ip"]} encontra-se na cidade {resposta["city"]}')
```

```
O PC com IP 193.136.227.157 encontra-se na cidade Faro

Process finished with exit code 0
```

Utilização do serviço Weatherapi-com para previsão de temperatura

```
import requests

url = "https://weatherapi-com.p.rapidapi.com/forecast.json"

N_dias = 3
querystring = {"q":"Faro","days":str(N_dias)}

headers = {
    'x-rapidapi-host': "weatherapi-com.p.rapidapi.com",
    'x-rapidapi-key': "ec9b161860msh650e03cd3a28230p1af29ejsncefed8e6f5e9"
    }

response = requests.request("GET", url, headers=headers, params=querystring)

resposta=response.json()
print(f'''Temperatura atualizada (ultima medição) para Faro: {resposta["current"]["temp_c"]} ºC,
Condições atmosféricas: {resposta["current"]["condition"]["text"]}''')
previsoes = resposta["forecast"]["forecastday"]
for dia in previsoes:
    print(f'Dia: {dia["date"]} minima prevista: {dia["day"]["mintemp_c"]} maxima prevista: {dia["day"]["maxtemp_c"]}')
```

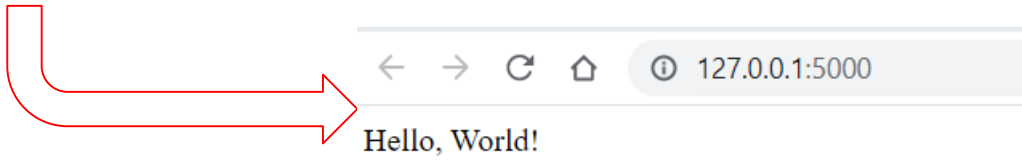
E depois para o nosso sistema com Arduino…

```
import requests

id_sensores=range(800,801+1)
api_key = "6a6c38b568ef38532db0e29930e5bdf9"  #api_key emoncms para READONLY

url = f'http://193.136.227.157/emoncms/feed/timevalue.json'

headers={'Authorization': f'Bearer {api_key}'}

querystring = {"id":{id_sensores[0]},"apikey":{api_key}}
response = requests.request("GET", url, headers=headers, params=querystring)
resposta=response.json()
print(f'Valor da Temperatura: {resposta["value"]}')

querystring = {"id":{id_sensores[1]},"apikey":{api_key}}
response = requests.request("GET", url, headers=headers, params=querystring)
resposta=response.json()
print(f'Valor da Humidade: {resposta["value"]}')
```

Web Service Gateway Interface – Flask

```
#flask1.py
from flask import Flask

app = Flask(__name__)
 # nome do módulo, local onde flask procura recursos
@app.route("/")  #indica a flask que url ativa a função

def hello_world():
    return "<p>Hello, World!</p>"
```

```
(venv) C:\Users\Cristiano\PycharmProjects\aulas_POO\venv>set FLASK_APP=flask1

(venv) C:\Users\Cristiano\PycharmProjects\aulas_POO\venv>set FLASK_ENV=development

(venv) C:\Users\Cristiano\PycharmProjects\aulas_POO\venv>flask run
 * Serving Flask app 'flask1' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 267-678-956
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



A continuar…

- **… no Pycharm**
- **Pasta Flask-Api:**
  - run_app.py: ficheiro de arranque da aplicação
  - test_API: ficheiro de teste dos métodos

- Exemplos em flask seguem o projeto em : https://flask.palletsprojects.com/en/2.0.x/quickstart/

- O Servidor built-in do flask não está preparado para produção. Se pretender soluções para o implementar num servidor WSGI (Web Service Gateway Service):
  - https://flask.palletsprojects.com/en/2.0.x/deploying/

- **Nota:** para efetuar os testes de ligação ao WSGI pode ser usado o Yet Another Rest Client instalado como extensão no chrome em alternativa ao ficheiro test_API.py