

Práctica 3: Creación de API REST con Bottle

Autores:

Alejandro Serrano Fernández

Pedro Antonio Navas Luque

Prefacio:

Para que el código funcione correctamente, es necesario la instalación de bottle para Python 3. Dado a que hemos tenido unos cuantos problemas para su instalación, es necesario realizar los siguientes comandos en la terminal de GNU/Linux:

```
sudo apt-get install python3-pip
```

```
pip3 install bottle
```

Introducción:

Antes de comenzar con la memoria de esta práctica, recordar que los códigos de esta práctica están realizados bajo la versión 3 de Python. Esta práctica está realizada por los alumnos Alejandro Serrano Fernández y Pedro Antonio Navas Luque para la asignatura de Sistemas Distribuidos. En esta práctica trabajamos con el framework Bottle para ofrecer un servicio web de gestión de habitaciones de un hotel.

1. Funcionamiento del sistema

El sistema implementado estaría dispuesto de un cliente y un servidor (en nuestro caso se encuentran en el mismo equipo), en el que ambos interaccionan mediante el protocolo HTTP y el uso de JSON para el envío de datos entre ambos equipo. Cuando el cliente solicite algún dato sobre las habitaciones, hará una petición GET al servidor y éste le ofrecerá el archivo JSON correspondiente. Al contrario, cuando el cliente solicite una petición POST, éste le ofrecerá su JSON (con los datos a enviar) al servidor para hacer las modificaciones que sean pertinentes.

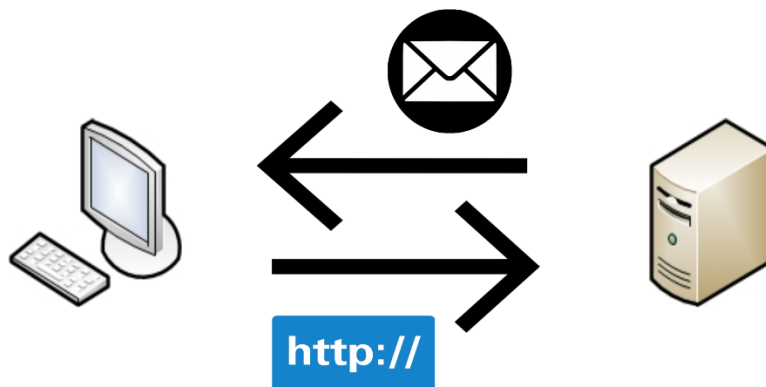
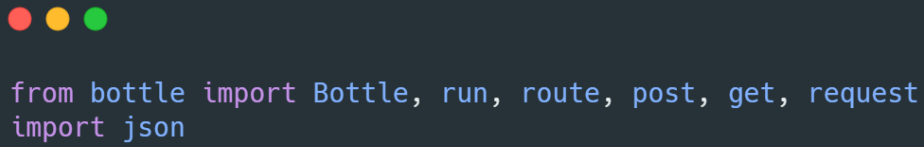


Figura 1: Concepto de interacción entre cliente y servidor

2. Implementación

Para el correcto funcionamiento del programa, importaremos en el código fuente del servidor, las librerías mostradas en la siguiente imagen:



```
from bottle import Bottle, run, route, post, get, request
import json
```

Por el contrario, para el código fuente del cliente tan solo importaríamos la librería “requests”, con la que haremos uso de las funciones requests.get() y requests.post()

2.1 Alta de una habitación



```
@post('/send')
def anadir():
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    lista_aux = {'Identificador': request.json.get('Identificador'),
                'num_plazas': request.json.get('num_plazas'), 'equipamiento': request.json.get('equipamiento'),
                'ocupada': request.json.get('ocupada')}

    encontrado = False

    d1 = json.dumps(habitacion) #Lo convierte a un diccionario de json
    d2 = json.loads(d1)
    for key in d2:
        if (key['Identificador'] == lista_aux['Identificador']):
            encontrado = True

    if(encontrado == False):
        habitacion.append(lista_aux)
        print("Añadido correctamente")

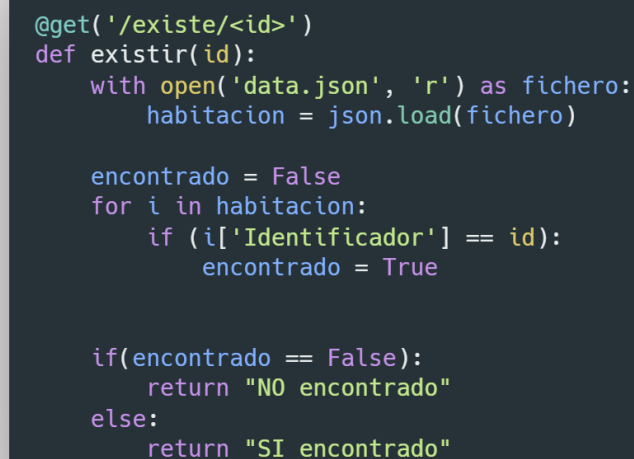
    else:
        print("Identificador repetido")

    with open('data.json', 'w') as fichero:
        json.dump(habitacion, fichero)
```

Figura 2: Código fuente del servidor para dar de alta una nueva habitación

Definimos una petición post con el que recibiremos los datos a introducir de la nueva habitación. Para ello primeramente abrimos nuestro archivo JSON y lo cargaremos en una variable local. Posteriormente recibimos los datos introducidos por el cliente con el método `request.json.get()`. Una vez lo recibimos, el servidor comprobará que el identificador de la habitación no está repetido y añadirá sus datos a la variable `habitaciones` para ser posteriormente sobrescrito en el archivo JSON.

En cuanto al código fuente del cliente, éste desea conocer si la habitación introducida está registrada o no en el sistema, para ello realiza un `request.get()` con el que obtendrá dicha información.



```
@get('/existe/<id>')
def existir(id):
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    encontrado = False
    for i in habitacion:
        if (i['Identificador'] == id):
            encontrado = True

    if(encontrado == False):
        return "NO encontrado"
    else:
        return "SI encontrado"
```

Figura 3: Petición GET para conocer si una habitación está ocupada o no

En el caso de que no lo esté, el cliente introducirá los datos correspondientes y serán enviados al servidor para que los almacene en un archivo JSON.

```

if (response == '1'):
    num_hab = input("Introduce identificador de la habitacion: ")
    req = requests.get(url = exists+num_hab)
    data = req.text

    if(data == "SI encontrado"):
        print("\n*****")
        print("\n\tLA HABITACION YA EXISTE\n")
        print("*****\n")

    else:

        plazas = input("Introduce plazas de la habitacion: ")
        equip = input("Introduce lista de equipamientos: ")
        ocup = input("¿Ocupada? (si/no): ")

        if(ocup != "si" and ocup != "no"):
            print("\n***Error, respuesta distinta a si o no***")

        else:
            lista = {'Identificador' : num_hab, 'num_plazas' : plazas, 'equipamiento': equip,
'ocupada': ocup}
            requests.post(page, json = lista)

            print("\n*****")
            print("\n\tAÑADIDA CORRECTAMENTE\n")
            print("*****\n")

```

Figura 4: Código fuente del cliente para dar de alta una nueva habitación

2.2 Modificar una habitación

Cuando el cliente selecciona la opción de modificar una habitación, éste, al igual que en el apartado anterior, desea saber si una habitación está ocupada o no antes de proceder a introducir más datos. Posteriormente, si la habitación se encuentra en el sistema, el cliente le solicitará al servidor los datos introducidos y éste se encargará de recibirlos, eliminar la habitación a modificar, y añadir sus nuevos datos. Una vez añadidos, los nuevos cambios quedarán registrado en el archivo JSON que contiene las habitaciones.

```

elif (response == '2'):

    num_hab = input("Introduce identificador de la habitacion: ")
    req = requests.get(url = exists+num_hab)
    data = req.text

    if(data == "NO encontrado"):
        print("\n*****")
        print("\n\tNO SE HA ENCONTRADO LA HABITACION\n")
        print("*****\n")

    else:
        plazas = input("Introduce plazas de la habitacion: ")
        equip = input("Introduce lista de equipamientos: ")
        ocup = input("¿Ocupada? (si/no): ")
        lista = {'num_plazas' : plazas, 'equipamiento': equip, 'ocupada': ocup}
        requests.post(modify+num_hab, json = lista)

        print("\n*****")
        print("\n\tSE HA ACTUALIZADO CORRECTAMENTE\n")
        print("*****\n")

```

Figura 5: Código fuente del cliente para realizar una modificación en una habitación

```

@post('/modify/<id>')
def modificar(id):
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    lista_aux = {'Identificador': id, 'num_plazas':request.json.get('num_plazas'),
'equipamiento':request.json.get('equipamiento'), 'ocupada':request.json.get('ocupada')}
    print(lista_aux['Identificador'])
    encontrado = False

    #d1 = json.dumps(habitacion) #Lo convierte a un diccionario de json
    #d2 = json.loads(d1)
    for i in habitacion:
        if (i['Identificador'] == lista_aux['Identificador']):
            habitacion.remove(i)
            encontrado = True

    if(encontrado == True):
        habitacion.append(lista_aux)
        print("Actualizado correctamente")

    else:
        print("NO se ha encontrado la habitacion")

    with open('data.json', 'w') as fichero:
        json.dump(habitacion,fichero)

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

2.3 Consultar la lista completa de habitaciones

```
elif (response == '3'):
    r = requests.get(url = consulta)
    message = r.json()

    print("\n\t~~~~~DATOS DE HABITACIONES~~~~~\n")

    for i in message['dict']:
        print("=====")
        print ("Id. habitación: "+i['Identificador'])
        print ("Num. plazas disponibles: "+i['num_plazas'])
        print ("Lista de equipamientos: "+i['equipamiento'])
        print ("Ocupada: "+i['ocupada'])
        print ("=====\\n")
```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

```
@get('/listado')
def mostrar_habitaciones():
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)
    return dict(dict = habitacion)
```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

2.4 Consultar una habitación mediante identificador

```

@get('/conshabit/<id>')
def consultar_habitacion(id):
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    encontrado = False
    for i in habitacion:
        if (i['Identificador'] == id):
            encontrado = True
            return dict (dict = i)

    if(encontrado == False):
        return dict (dict = {'Identificador': 'no encontrado'})

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

```

elif (response == '4'):
    resp = input("Introduce el identificador de la habitacion: ")
    consulta_aux = consultaaid+resp;
    r = requests.get(url = consulta_aux)
    message = r.json()

    if (message['dict']['Identificador'] == 'no encontrado'):
        print ( "\n=====")
        print ( "\n\t!IDENTIFICADOR NO ENCONTRADO!\n")
        print ( "=====\\n")

    else:
        print( " \\n=====")
        print ( "Id. habitación: "+message['dict']['Identificador'])
        print ( "Num. plazas disponibles: "+message['dict']['num_plazas'])
        print ( "Lista de equipamientos: "+message['dict']['equipamiento'])
        print ( "Ocupada: "+message['dict']['ocupada'])
        print ( "=====\\n")

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

2.5 Consultar habitaciones ocupadas o desocupadas

```
elif (response == '5'):

    print("¿Que desea: ?")
    resp = input("1. Ocupadas (Respoder si)\n2. No ocupadas (Respoder no)\n")

    if(resp != 'si' and resp != 'no'):
        print("\n***Error, respuesta diferente a si o no***")
    else:
        r = requests.get(url = habocupadas+resp)
        message = r.json()

        for i in message['dict']:
            if (i['Identificador'] == 'no encontrado'):
                print ("\n=====")
                print ("\n\ti NO SE HAN ENCONTRADO HABITACIONES !\n")
                print ("=====\n")
            else:
                print( "\n=====")
                print ("Id. habitación: "+i['Identificador'])
                print ("Num. plazas disponibles: "+i['num_plazas'])
                print ("Lista de equipamientos: "+i['equipamiento'])
                print ("Ocupada: "+i['ocupada'])
```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

```

@get('/listaocup/<decision>')
def mostrar_ocupadas(decision):
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    lista = []

    encontrado = False
    for i in habitacion:
        if (i['ocupada'] == decision):
            encontrado = True
            lista.append(i)

    print(lista)
    if(encontrado == False):
        return dict (dict = [{'Identificador': 'no encontrado'}])

    else:
        return dict (dict = lista)

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

2.6 Consultar números de habitaciones con las plazas deseadas

```

@get('/plazas/<num>/<num2>')
def plazas_habitacion(num,num2):
    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    lista = []

    encontrado = False
    for i in habitacion:
        if (int(i['num_plazas']) >= int(num) and int(i['num_plazas']) <= int(num2)):
            encontrado = True
            lista.append(i)

    print(lista)
    if(encontrado == False):
        return dict (dict = [{'Identificador': 'no encontrado'}])

    else:
        return dict (dict = lista)

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

```
elif (response == '6'):  
  
    print("Introduzca el intervalo de plazas")  
    resp1 = input("Limite inferior de plazas: ")  
    resp2 = input("Limite superior de plazas: ")  
  
    if(int(resp1) > int(resp2)):  
        print("Error, intervalo mal introducido")  
    else:  
        r = requests.get(url = plaza+resp1+"/"+resp2)  
        message = r.json()  
  
        print("\n")  
        for i in message['dict']:  
            if (i['Identificador'] == 'no encontrado'):  
                print ("\n=====")  
                print ("\n\ti NO SE HAN ENCONTRADO HABITACIONES !\n")  
                print ("=====\\n")  
            else:  
                print( "=====")  
                print ("Id. habitación: "+i['Identificador'])  
                print ("Num. plazas disponibles: "+i['num_plazas'])  
                print ("Lista de equipamientos: "+i['equipamiento'])  
                print ("Ocupada: "+i['ocupada'])  
                print ("=====\\n")
```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

2.7 Borrar una habitación

```

elif (response == '7'):
    num_hab = input("Introduce identificador de la habitacion: ")
    req = requests.get(url = exists+num_hab)
    data = req.text

    if(data == "NO encontrado"):
        print("\n*****")
        print("\n\tNO SE HA ENCONTRADO LA HABITACION\n")
        print("*****\n")
    else:
        requests.post(remover+num_hab)
        print("\n*****")
        print("\n\tSE HA BORRADO CORRECTAMENTE\n")
        print("*****\n")

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

```

@post('/remove/<id>')
def modificar(id):

    with open('data.json', 'r') as fichero:
        habitacion = json.load(fichero)

    encontrado = False

    for i in habitacion:
        if (i['Identificador'] == id):
            habitacion.remove(i)
            encontrado = True

    if(encontrado == True):
        print("Borrado correctamente")

    else:
        print("NO se ha encontrado la habitacion")

    with open('data.json', 'w') as fichero:
        json.dump(habitacion,fichero)

```

Figura 6: Código fuente del servidor para realizar una modificación en una habitación

3. Ejecución del programa

Para la ejecución del programa es necesario disponer de la librería Bottle, tal y como se indica en el prefacio de este documento. Disponemos de tres archivos, server.py (encargado de recibir y de enviar datos de las habitaciones al cliente), client.py (Dispone de una interfaz con la que interactuar o manipular los datos de las habitaciones) y de un archivo JSON (donde se almacenarán los datos de las habitaciones). Este último ha de estar en el mismo directorio que el archivo server.py. Para el correcto funcionamiento del programa es necesario que no se modifique ni se borre el archivo. JSON proporcionado.

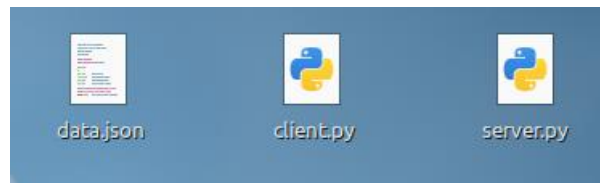


Figura x: Muestra de los ficheros necesarios para ejecutar el servicio

Para comenzar, primero ejecutaremos el archivo `server.py` con Python 3. Una vez ejecutado el servidor está preparado para recibir las distintas peticiones del cliente. El servidor quedaría abierto hasta que no se le indique lo contrario, con el comando `Ctrl + c`.

```
alejandro@alejandro-VirtualBox: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
alejandro@alejandro-VirtualBox:~/Escritorio$ python3 server.py
Bottle v0.12.18 server starting up (using WSGIRefServer())...
Listening on http://localhost:8081/
Hit Ctrl-C to quit.
```

Figura x: Muestra de la terminal del archivo `server.py`

Posteriormente, podemos iniciar el archivo `client.py` con el que interactuaremos con el servidor. Una vez iniciado se nos mostraría de la siguiente manera:

```
alejandro@alejandro-VirtualBox: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
alejandro@alejandro-VirtualBox:~/Escritorio$ python3 client.py
Elige la opcion que deseas realizar:
  1. Dar de alta una nueva habitacion
  2. Modificar los datos de una habitacion.
  3. Consultar la lista completa de habitaciones.
  4. Consultar una habitacion mediante identificador.
  5. Consultar la lista de habitaciones ocupadas o desocupadas
  6. Consultar el numero de habitaciones con las plazas deseadas
  7. Borrar una habitacion
  8. Salir
Introduce respuesta:
```

Figura x: Muestra de la terminal del archivo `client.py`

Finalmente, quedaría a disposición del usuario la elección de cualquiera de las opciones que se proponen.

Hay que destacar, que el programa del cliente está totalmente preparado por si en algún momento el usuario introduce una opción incorrecta, habitación incorrecta...