

# Documentación trabajo de Python

## Seminario02

Sistemas Distribuidos

Universidad de Cádiz  
Grado en Ingeniería Informática

Curso 2014/2015

### Colaboradores :

- *Braulio Pareja Manzanares*
- *Carlos Campos Fe*
- *Juan Francisco García Pereira*
- *Luis Miguel Páez Molina*
- *Jose Antonio Muñoz Fuentes*

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Planificación</b>	<b>3</b>
<b>3. Requisitos</b>	<b>3</b>
<b>4. Código del programa</b>	<b>4</b>
4.1 Servidor .....	4
4.2 Cliente .....	6
<b>5. ¿Cómo se usa ?</b>	<b>8</b>

# 1. Introducción:

Hemos realizado un programa que consiste en compartir una carpeta entre un servidor y varios clientes de modo que los clientes tendrán accesos local en su equipo a todos los archivos del servidor y viceversa, así se actualizarán todos los archivos para que estén en las dos carpetas. Hemos utilizado Zeromq para la conexión entre los equipos y la transferencia de ficheros.

# 2. Planificación:

- Idea del proyecto: planteamos un proyecto que utilizara zeromq. Consultamos en la web ideas para realizar uno y al final nos surgió la idea de compartir una carpeta entre un servidor y varios clientes.

- Planteamiento: planteamos el proyecto viendo si era factible realizarlo mediante zeromq y que otros métodos nos hacía falta (listado de nombres de archivos, conexión udp, etc).

- Implementación del programa: una vez con todos los conocimientos que nos hacían falta, implementamos el programa a base de sprints funcionales de menor a mayor complejidad, realizando el programa en varios ficheros y en dos equipos diferentes (cliente y servidor)

- Pruebas y errores: una vez implementado, hicimos varias pruebas diferentes y solucionamos los errores que nos iban apareciendo.

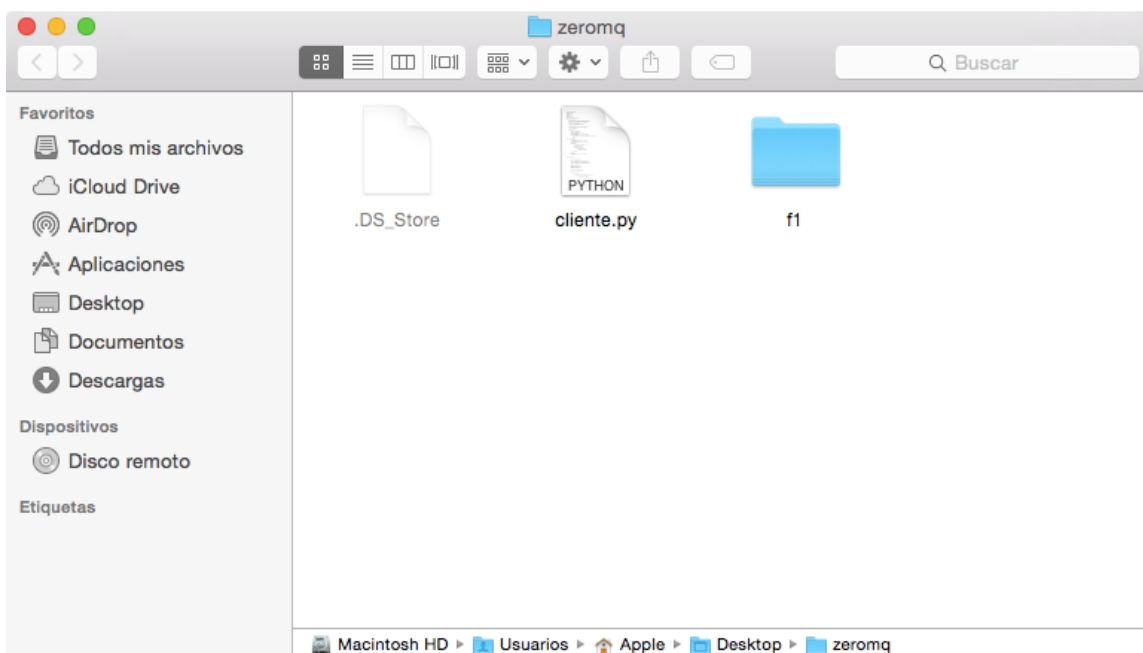
- Documentación: una vez depurado todo el programa, realizamos la documentación.

# 3. Requisitos:

- Librerías importadas: os, sys, json, zmq, time, socket.

- IP del servidor

- Carpeta con nombre “f1” en la raíz del fichero cliente.py y en la raíz del fichero servidor.py que es donde se guardaran los archivos compartidos.



## 4. Código del programa

### 4.1 Servidor

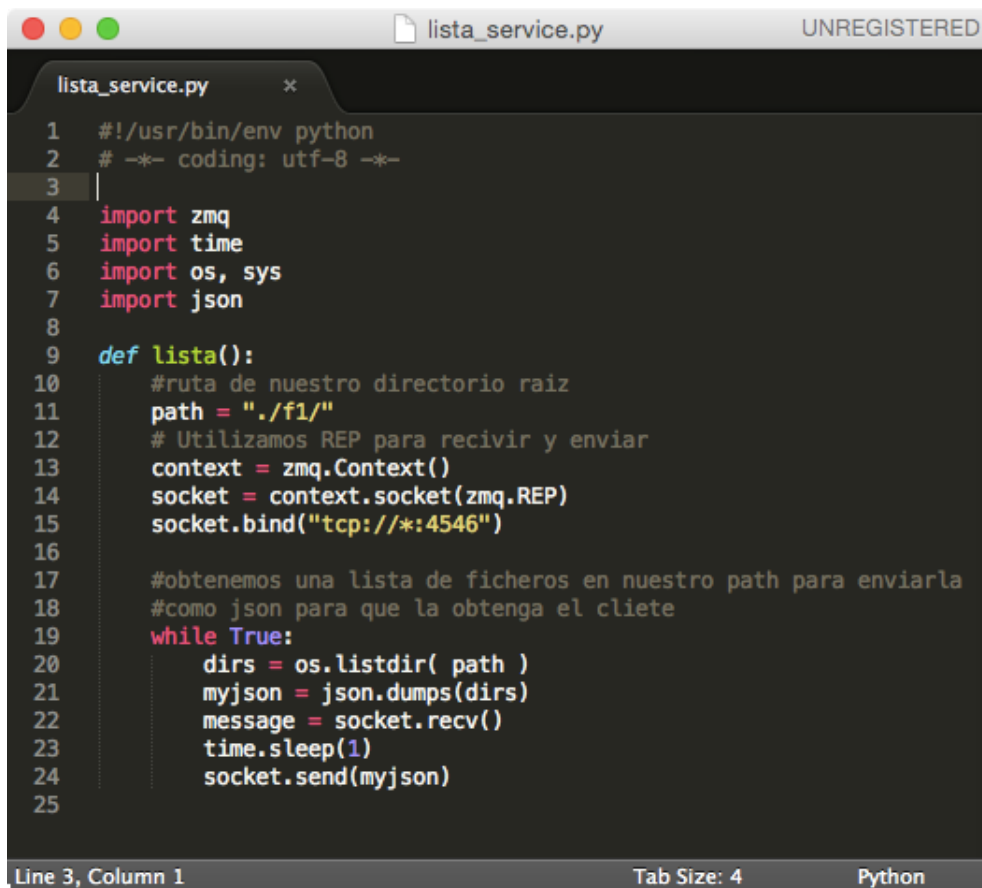


The screenshot shows a code editor window titled 'servidor.py' with a tab size of 4 and Python syntax highlighting. The code defines a 'main' function that creates three threads: 's1' for file service, 's2' for list service, and 's3' for reception service. Each thread is started and then the function ends.

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  #importamos los ficheros generados para ejecutar con varios hilos a la vez
5  import file_service
6  import lista_service
7  import recep_service
8  from threading import Thread
9
10 def main():
11     s1=Thread(target=file_service.server)
12     s2=Thread(target=lista_service.lista)
13     s3=Thread(target=recep_service.escucha)
14     s1.start()
15     s2.start()
16     s3.start()
17
```

Line 1, Column 22      Tab Size: 4      Python

Código main del servidor



The screenshot shows a code editor window titled 'lista\_service.py' with a tab size of 4 and Python syntax highlighting. The code defines a 'lista' function that uses 'zmq' to receive a request, gets a list of files from a specified path, and sends the list back to the client as JSON.

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  import zmq
5  import time
6  import os, sys
7  import json
8
9  def lista():
10     #ruta de nuestro directorio raiz
11     path = "./f1/"
12     # Utilizamos REP para recibir y enviar
13     context = zmq.Context()
14     socket = context.socket(zmq.REP)
15     socket.bind("tcp://*:4546")
16
17     #obtenemos una lista de ficheros en nuestro path para enviarla
18     #como json para que la obtenga el cliete
19     while True:
20         dirs = os.listdir( path )
21         myjson = json.dumps(dirs)
22         message = socket.recv()
23         time.sleep(1)
24         socket.send(myjson)
25
```

Line 3, Column 1      Tab Size: 4      Python

Código que obtiene una lista con los nombre de los archivos de la carpeta del servidor y la envía al cliente

```
recep_service.py UNREGISTERED

recep_service.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  import os
5  import sys
6  from socket import socket, error
7  def escucha():
8
9      s = socket()
10     # Escuchar peticiones en el puerto 4547.
11     s.bind(('', 4547))
12     while True:
13         s.listen(0)
14         conn, addr = s.accept()
15         path = conn.recv(1024)
16         dest = open("f1/"+path, 'w+')
17         while True:
18             try:
19                 # Recibir datos del cliente.
20
21                 input_data = conn.recv(1024)
22             except error:
23                 print("Error de lectura.")
24                 break
25             else:
26                 if input_data:
27                     # Almacenar datos.
28                     dest.write(input_data)
29                 else:
30                     break
31         print("El archivo se ha recibido correctamente.")
32         dest.close()
33
```

Line 1, Column 1 Tab Size: 4 Python

Recibe archivos del cliente y los guarda en su carpeta

```
file_service.py UNREGISTERED

file_service.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  import zmq
5  import os
6
7  def server():
8
9      # Utilizamos REP para recibir y enviar
10     context = zmq.Context(1)
11     sock = context.socket(zmq.REP)
12     sock.bind('tcp://*:4548')
13
14     # bucle de espera
15     while True:
16         # recepcion de nombre fichero
17         msg = sock.recv()
18         # Verificamos si esta disponible
19         if not os.path.isfile(msg):
20             sock.send('')
21             continue
22         # Abrimos el fichero para leer
23         fn = open(msg, 'rb')
24         stream = True
25         # empezamos a leer para enviar
26         while stream:
27             # Lectura bit a bit
28             stream = fn.read(128)
29             if stream:
30                 # enviamos datos por bloques
31                 sock.send(stream, zmq.SNDMORE)
32             else:
33                 # contenido vacio terminamos
34                 sock.send(stream)
35
```

Line 1, Column 1 Spaces: 4 Python

Envia archivos al cliente

## 4.2 Cliente



```
1  #!/usr/bin/python2
2  # -*- coding: utf-8 -*-
3  import time
4  import zmq
5  import os
6  import sys
7  import json
8  from socket import socket
9
10 global dirip    #variable global que guarda la direccion ip del servidor
11
12 #Funcion que comprueba si el parametro recibido es una direccion IP
13 def validate_ip(s):
14     a = s.split('.')
15     if len(a) != 4:
16         return False
17     for x in a:
18         if not x.isdigit():
19             return False
20         i = int(x)
21         if i < 0 or i > 255:
22             return False
23     return True
24
25 #Se comprueba si se han introducido parametros por linea de comando
26 if(len(sys.argv) > 1):
27     dirip=sys.argv[1] #Se asigna el parametro recibido por
28     #linea de comando a la variable global que dirip
29     if not validate_ip(dirip): #Se llama a la funcion validate_ip para
30     #comprobar que el parametro recibido corresponde con una direccion IP
31     print "Ip no válida"
32     sys.exit() #Finaliza la ejecucion del programa
33 else:
34     print "Introduce la direccion IP del servidor"
35     sys.exit() #Finaliza la ejecucion del programa
36
37 #Funcion que obtiene un archivo del servidor a partir del nombre del fichero
38 def get_file(path):
39     global dirip
40     # Open up the file we are going to write to
41     dest = open("f1/"+os.path.basename(path), 'w+')
42     # Set up the zeromq context and socket
43     context = zmq.Context()
44     socket = context.socket(zmq.REQ)
45
46     socket.connect('tcp://'+dirip+'4548')
47     # send the desired file to the server
48     socket.send(path)
49
```

Line 1, Column 1      Tab Size: 4      Python

Codigo del cliente (parte 1)

```
cliente.py
50 while True:
51     # Start grabbing data
52     data = socket.recv()
53     # Write the chunk to the file
54     dest.write(data)
55     if not socket.getsockopt(zmq.RCVMORE):
56         # If there is not more data to send, then break
57         break
58
59 #Funcion que envia un fichero al servidor a partir del nombre
60 def put_file(path):
61     global dirip
62     s = socket()
63     s.connect((dirip, 4547)) # conexion con el servidor por el puerto 4547
64     s.send(path) #Se envia el nombre del fichero al servidor
65
66     while True:
67         f = open("./f1/"+path, "rb")
68         content = f.read()
69         while content:
70             # Enviar contenido.
71             s.send(content)
72             content = f.read()
73         break
74
75     s.close() #cierre de conexion
76     f.close() #cierre de fichero
77     print("Sincronizacion finalizada")
78
79 #Funcion principal que se encarga de sincronizar los ficheros
80 def ficheros():
81
82     global dirip
83     context = zmq.Context()
84
85     print "Conectando con el servidor"
86     socket = context.socket(zmq.REQ)
87     socket.connect("tcp://" + dirip + ":4546") # conexion con el servidor
88                                           #por el puerto 4546
89
90     socket.send("...")
91     message = socket.recv() # se recibe la lista de ficheros
92                               #que se encuentran en el servidor
93     socket.send("...")
94     message = socket.recv()
95
96     data= json.loads(message) #se guarda la lista de ficheros
97                               # contenida en un json en una lista
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
```

Line 122, Column 38

Tab Size: 4

Python

Código del cliente (parte 2)

```
cliente.py
99 #se recorre la lista
100 for result in data:
101
102     if not os.path.exists("./f1/"+result): #se comprueban
103         #si los ficheros existen en el directorio f1 si no existen se añaden
104
105         print "Sincronizando..."
106
107         get_file(str(result)) # se llama a la
108                               #funcion get_file para obtener los
109                               #ficheros que faltan en nuestro directori f1
110
111
112
113     dirs = os.listdir( "./f1/" ) #se obtiene la lista de ficheros
114                               #contenidos en nuestro directorio f1
115     #se recorre la lista
116     for result in dirs:
117         if not result in data: #se comprueba si los ficheros de
118                               #nuestro directorio f1 se encuentran en el servidor y si
119                               #no se encuentran de añaden
120
121         print "Sincronizando con el servidor"
122         put_file(str(result)) #se llama a la funcion put_file para
123                               #añadir al servidor los ficheros que le faltan
124
125
126 if __name__ == '__main__':
127     #bucle que se encarga de que cada 10 segundos se sincronicen los ficheros
128     while True:
129         ficheros()
130         time.sleep(10)
131
132
133
```

Line 122, Column 38

Tab Size: 4

Python

Código del cliente (parte 3)

