Resolución Práctica 3.5.1: Implementación de un cliente POP3 particular

Fichero *“pop-local_cli.py”*:

```
#!/usr/bin/env python3

import socket, sys, time
import getpass

# Configuración servidor POP local de la asignatura SAR
SERV_POP = "dif-mail.ehu.es"
#SERV_POP = "158.227.106.40"
PORT_POP = 2110

class ComPOP3:
	Capa, User, Pass, Stat, Top, Quit = ("CAPA", "USER", "PASS", "STAT", 
"TOP", "QUIT")

def recvline( s, removeEOL = True ):
	line = b''
	CRreceived = False
	while True:
		c = s.recv( 1 )
		if c == b'':
			raise EOFError( "Connection closed by the peer before 
receiving an EOL." )
		line += c
		if c == b'\r':
			CRreceived = True
		elif c == b'\n' and CRreceived:
			if removeEOL:
				return line[:-2]
			else:
				return line
		else:
			CRreceived = False

def recvmultiline( s ):
	msg = recvline( s ).decode( "ascii" )
	if isPOPerror( msg ):
		exit( 1 )
	mline = []
	while msg != ".":
		try:
			msg = recvline( s ).decode( "ascii" )
		except Exception as e:
			print( "Error: {}".format( e ) )
			continue
		else:
			if msg != ".":
				mline.append( msg )
	return mline

def isPOPerror( msg ):
	if msg.startswith( "-ERR" ):
		print( "ERROR! {}".format( msg[5:] ))
		return True
	else:
		return False
```

```
def int2bytes( n ):
      if n < 1 << 10:
            return str(n) + " B  "
      elif n < 1 << 20:
            return str(round( n / (1 << 10) ) ) + " KiB"
      elif n < 1 << 30:
            return str(round( n / (1 << 20) ) ) + " MiB"
      else:
            return str(round( n / (1 << 30) ) ) + " GiB"

# Programa principal
if __name__ == "__main__":
      if len( sys.argv ) != 1:
            print( "Uso: python3 {}".format( sys.argv[0] ) )
            exit( 1 )

      ############################################
      # Analizar buźon usuario POP3 servidor local
      ############################################
      serv_pop = (SERV_POP, PORT_POP)

      s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )

      s.connect( serv_pop )

      # Saludo del servidor POP3
      msg = recvline( s ).decode( "ascii" )
      if isPOPerror( msg ):
            exit( 1 )

      # Capacidades servidor POP3 (Opcional)
      msg = "{}\r\n".format( ComPOP3.Capa )
      s.sendall( msg.encode( "ascii" ) )
      mline = recvmultiline( s )
      for l in mline:
            print( l )

      # The AUTHORIZATION State
      CUENTA = input( "Introduce tu cuenta asociada al correo del servidor 
local: " )
      msg = "{} {}\r\n".format( ComPOP3.User, CUENTA )
      s.sendall( msg.encode( "ascii" ) )
      msg = recvline( s ).decode( "ascii" )
      if isPOPerror( msg ):
            exit( 1 )

      CLAVE = getpass.getpass()
      msg = "{} {}\r\n".format( ComPOP3.Pass, CLAVE )
      s.sendall( msg.encode( "ascii" ) )
      msg = recvline( s ).decode( "ascii" )
      print( msg )
      if isPOPerror( msg ):
            exit( 1 )
      else:
            print( "Usuario autenticado en servidor POP3." )

      # The TRANSACTION State
      msg = "{}\r\n".format( ComPOP3.Stat )
      print( ComPOP3.Stat )
      s.sendall( msg.encode( "ascii" ) )
      msg = recvline( s ).decode( "ascii" )
      if isPOPerror( msg ):
            exit( 1 )
      tokens = msg.split()
      print( 'Número de mensajes: {}, Tamaño del buzón: {}'.format( tokens[1],
int2bytes( int(tokens[2]) ) ))
      num_msgs = int( tokens[1] )

      # Lista de asignaturas
      lasign = ['SAR', 'SZA']
      # Lista de contadores
      lcont = dict()
      for asign in lasign:
            lcont[ asign ] = 0
      for i in range( num_msgs ):
            msg = "{} {} 0\r\n".format( ComPOP3.Top, i + 1 )
            s.sendall( msg.encode( "ascii" ) )
            mline = recvmultiline( s )
            for l in mline:
                  if "Subject:" in l:
                        for asign in lasign:
#                             if asign + ':' in l:
                              if asign in l:
                                    lcont[ asign ] += 1
                                    break
                        break
      for asig, cont in lcont.items():
            print( "{}: {}".format( asig, cont ) )

      # The UPDATE State
      # Cerrar sesión de usuario POP3
      msg = "{}\r\n".format( ComPOP3.Quit )
      s.sendall( msg.encode( "ascii" ) )
      msg = recvline( s ).decode( "ascii" )
      if isPOPerror( msg ):
            exit( 1 )
      else:
            print( msg )
      # Cerrar conexión con servidor POP3
      s.close()
```