Resolución Práctica 3.5.2: Implementación de un cliente POP3-SMTP particular

Fichero *“pop-smtp-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"
PORT_POP = 2110

# Configuración servidor SMTP local de la asignatura SAR
SERV_SMTP = SERV_POP
PORT_SMTP = 2025

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

class ComSMTP:
      Helo, Ehlo, STLS, From, To, Data, Quit = ("HELO", "EHLO", "STARTTLS",
"MAIL FROM:", "RCPT TO:", "DATA", "QUIT")

# Código OK (respuesta positiva) a cada comando SMTP
CodOKSMTP = dict()
CodOKSMTP[ "connect" ] = CodOKSMTP[ ComSMTP.STLS ] = "220"
CodOKSMTP[ ComSMTP.Helo ] = CodOKSMTP[ ComSMTP.Ehlo ] =
CodOKSMTP[ ComSMTP.From ] = CodOKSMTP[ ComSMTP.To ] = "250"
CodOKSMTP[ ComSMTP.Data ] = "354"
CodOKSMTP[ ComSMTP.Quit ] = "221"

# Campos 'Internet Message Format' (IMF) [RFC5322]
class FieldIMF:
      Date, From, To, Subject = ("Date:", "From:", "To:", "Subject:")

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 recvEHLOmultiline( s ):
	msg = recvline( s ).decode( "ascii" )
	if isSMTPerror( msg, ComSMTP.Ehlo ):
		exit( 1 )
	print( msg )
	mline = []
	while msg.startswith( "250" ) and not msg.startswith( "250 " ):
		try:
			msg = recvline( s ).decode( "ascii" )
		except Exception as e:
			print( "Error: {}".format( e ) )
			#input( "RETURN sakatu!" )
			continue
		else:
			if msg.startswith( "250" ):
				mline.append( msg )
			else:
				print( "Error: {}".format( msg ) )
	return mline

def isSMTPerror( msg, comando = "connect" ):
	if not msg.startswith( CodOKSMTP[ comando ] ):
		print( "ERROR! {}".format( msg ))
		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()

####################################
# Enviar mensaje SMTP servidor local
####################################
serv_smtp = (SERV_SMTP, PORT_SMTP)

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

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

# Client Initiation
msg = "{} {}\r\n".format( ComSMTP.Ehlo, 'cliente SAR' )
s.sendall( msg.encode( "ascii" ) )
mline = recvEHLOmultiline( s )
for l in mline:
      print( l )

# Mail Transaction
## FROM
dir_origen = 'szasar@ehu.eus'
msg = "{} <{}>\r\n".format( ComSMTP.From, dir_origen )
s.sendall( msg.encode( "ascii" ) )
msg = recvline( s ).decode( "ascii" )
if isSMTPerror( msg, ComSMTP.From ):
      exit( 1 )

## TO
dir_destino = CUENTA + '@diffiss.eus'
msg = "{} <{}>\r\n".format( ComSMTP.To, dir_destino )
s.sendall( msg.encode( "ascii" ) )
msg = recvline( s ).decode( "ascii" )
if isSMTPerror( msg, ComSMTP.To ):
      exit( 1 )

## DATA
msg = "{}\r\n".format( ComSMTP.Data )
s.sendall( msg.encode( "ascii" ) )
msg = recvline( s ).decode( "ascii" )
if isSMTPerror( msg, ComSMTP.Data ):
      exit( 1 )

## Cuerpo mensaje
```

```
	lineas = []
	## Header section
	## Origination Date Field
	lineas.append( "{} {}".format( FieldIMF.Date, time.ctime() ))
	## Originator Fields
	lineas.append( "{} Asignatura SAR de UPV-EHU <{}>".format( FieldIMF.From,
dir_origen )  )
	## Destination Address Fields
	lineas.append( "{} Estudiante {} <{}>".format( FieldIMF.To, CUENTA,
dir_destino )  )
	## Informational Fields
	lineas.append( "{} {}".format( FieldIMF.Subject, "SAR: Resultado consulta
servidor POP" )  )
	# empty line
	lineas.append( "" )
	## Body
	lineas.append( "Número mensajes por aula virtual de eGela" )
	lineas.append( "---------------------------------------" )
	for asig, cont in lcont.items():
		lineas.append( "{}: {}".format( asig, cont ) )
	lineas.append( "Agur\r\nAsignatura SAR" )

	for l in lineas:
		s.sendall( "{}\r\n".format( l ).encode() )

	## Fin cuerpo mensaje
	msg = ".\r\n"
	s.sendall( msg.encode( "ascii" ) )
	msg = recvline( s ).decode( "ascii" )
	if isSMTPerror( msg, ComSMTP.Helo ):
		exit( 1 )
	else:
		print( msg )

	# Cerrar sesión de usuario SMTP
	msg = "{}\r\n".format( ComSMTP.Quit)
	s.sendall( msg.encode( "ascii" ) )
	msg = recvline( s ).decode( "ascii" )
	if isSMTPerror( msg, ComSMTP.Quit ):
		exit( 1 )
	else:
		print( msg )
	# Cerrar conexión con servidor SMTP
	s.close()
```