Resolución Práctica 3.5: Implementación de un cliente POP3-SMTP particular

Fichero *“pop-smtp-tls_cli.py”*:

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

import socket, sys, time
import getpass
import ssl

# Mirar configuración servidor POP en cliente de correo habitual
SERV_POP = "mailin.ehu.eus"
PORT_POP = 995

# Mirar configuración servidor SMTP en cliente de correo habitual
SERV_SMTP = "smtp.ehu.eus"
PORT_SMTP = 25

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 ) )
                  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 seguro
```

```
      ####################################
      serv_pop = (SERV_POP, PORT_POP)

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

      context = ssl.create_default_context()

      ssl_sock = context.wrap_socket(s, server_hostname = SERV_POP )

      ssl_sock.connect( serv_pop )

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

      # Certificado del servidor POP3
      cert_pop = ssl_sock.getpeercert()
      print( "Certificado del servidor POP3:" )
      print( cert_pop )
      try:
            ssl.match_hostname( cert_pop, SERV_POP )
      except Exception as e:
            print( "Error: {}".format( e ) )
            exit( 1 )
      else:
            print( "Verificado hostname '{}' en certificado servidor
POP3!".format( SERV_POP ) )

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

      # The AUTHORIZATION State
      LDAP = input( "Introduce la cuenta LDAP asociada al correo: " )
      msg = "{} {}\r\n".format( ComPOP3.User, LDAP )
      ssl_sock.sendall( msg.encode( "ascii" ) )
      msg = recvline( ssl_sock ).decode( "ascii" )
      if isPOPerror( msg ):
            exit( 1 )

      CLAVE = getpass.getpass()
      msg = "{} {}\r\n".format( ComPOP3.Pass, CLAVE )
      ssl_sock.sendall( msg.encode( "ascii" ) )
      msg = recvline( ssl_sock ).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 )
      ssl_sock.sendall( msg.encode( "ascii" ) )
      msg = recvline( ssl_sock ).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_2022-23', 'SZA_2022-23']
# 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 )
      ssl_sock.sendall( msg.encode( "ascii" ) )
      mline = recvmultiline( ssl_sock )
      for l in mline:
            if "Subject:" in l:
                  for asign in lasign:
                        if asign + ':' in l:
                              lcont[ asign ] += 1
                              break
                  break
for asig, cont in lcont.items():
      print( "{}: {}".format( asig, cont ) )

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

##############################
# Enviar mensaje SMTP seguro
##############################
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 )

# STARTTLS Extension to SMTP
msg = "{}\r\n".format( "STARTTLS" )
s.sendall( msg.encode( "ascii" ) )
msg = recvline( s ).decode( "ascii" )
if isSMTPerror( msg, ComSMTP.STLS ):
      exit( 1 )
else:
      print( msg )

# TLS negotiation
context = ssl.create_default_context()
```

```
      ssl_sock = context.wrap_socket(s, server_hostname = SERV_SMTP )

      # Certificado del servidor SMTP
      cert_smtp = ssl_sock.getpeercert()
      print( "Certificado del servidor SMTP:" )
      print( cert_smtp )
      try:
            ssl.match_hostname( cert_smtp, SERV_SMTP )
      except Exception as e:
            print( "Error: {}".format( e ) )
            exit( 1 )
      else:
            print( "Verificado hostname '{}' en certificado servidor
SMTP!".format( SERV_SMTP ) )

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

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

      ## TO
      nom_correo = input( "Introduce el nombre de cuenta de tu dirección de
correo (antes de @ikasle.ehu.eus): " )
      dir_destino = nom_correo + '@ikasle.ehu.eus'
      msg = "{} <{}>\r\n".format( ComSMTP.To, dir_destino )
      ssl_sock.sendall( msg.encode( "ascii" ) )
      msg = recvline( ssl_sock ).decode( "ascii" )
      if isSMTPerror( msg, ComSMTP.To ):
            exit( 1 )

      ## DATA
      msg = "{}\r\n".format( ComSMTP.Data )
      ssl_sock.sendall( msg.encode( "ascii" ) )
      msg = recvline( ssl_sock ).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( "{} Estudiante SAR <{}>".format( FieldIMF.From,
dir_origen )  )
      ## Destination Address Fields
      lineas.append( "{} Estudiante UPV/EHU <{}>".format( FieldIMF.To,
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\nEstudiante SAR" )

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

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

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