Faculty of Computers, Informatics and Microelectronics
Technical University of Moldova

Multimedia System
Laboratory work #4

---

e-mail Client

Authors:

Sezer AKSOY

Supervisor:

Alexandr Gavrisco

2018
Chişinau

Laboratory work #4

1. **General purpose**

Use application-level protocols (POP3/IMAP and SMTP) to implement an email client with basic functionality.

2. **Prerequisites**

- Read about OSI model (definition, basic info)
- Read about (de)serialization, parsing
- Client-Server architecture

3. **Task:**

How to implement this lab

The entrypoint of this lab is the base task. Each next task is based on previous one and has a corresponding grade. If you implement tasks for a certain grade, you have to prove by answering question related to implementation and lab's theme.

Base task (5 - 6)

There are some prerequisites:

- pick up an e-mail service (it should support IMAP/POP3 and SMTP clients),

Implement a simple mail client which offers the following features:

- Log in;
- Get number of unread messages;
- Get last N received messages (display subject, date, sender), ordered by date;
- Send a message. Next fields must be available - subject, recipient, CC, body;

Message Preview and MIME-types (7)

- Display additional info for messages list:
    - If there're attachments in the e-mail, display number of attached files;
    - If message is plain text, display a short preview of the message (e.g. first sentence);

- Allow to select and read a message (just display content)

IMAP, HTML and attachments (8)

- If you implemented previous task using POP3 protocol, migrate the app to IMAP;
- Allow to send e-mail with attachments (any file, at least 1 attachment);
- If an e-mail is in HTML format, either display it using a WebView or send as file and open in browser;
- Implement drafts (synchronized with the e-mail server)

Notifications and search (9)

- Check in background for new messages and display a notification (show pop-up and/or play a sound) when there's a new message;
- Add search feature (e.g. find messages which contain a in subject or sender).
- Add "remember me" option to the login screen (store passwords **securely**).

UX improvements (10)

- "Undo" button - allow to undo sending a message after user sent it (implementation is up to you);
- Implement pagination for messages (e.g. display 10 msg per page with next/prev buttons)
- Allow basic formatting (bold, italic etc) for a new email.
- If there's no internet connection, enqueue messages and send them later.

4. **Theory:**

Serialization is the process of turning an object in memory into a stream of bytes so you can do stuff like store it on disk or send it over the network.

Deserialization is the reverse process: turning a stream of bytes into an object in memory.

Parsing methods means that the method is taking input from a string and returning some other data type.

Definition of parse

The actual definition of "parse" in Wiktionary is "To split a file or other input into pieces of data that can be easily stored or manipulated." So we are splitting a string into parts then recognizing the parts to convert it into something simpler than a string.

Parsing an integer

An example would be the parseInt() function. It would take an input such as "123", which would be a string consisting of the char values 1, 2, and 3. Then it would convert this value to the integer 123, which is a simple number that can be stored and manipulated as an integer.

I can see why you might be confused by this simple example, though, since the string "123" doesn't have any obvious parts.

Parsing a date

Here is a better example involving parsing a date from a string:

```
1. SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
2. Date date = format.parse("2016-04-23");
```

This example shows a date format that actually has recognizable parts:

- yyyy is the year
- - is a literal dash
- MM is the month
- - is another literal dash
- dd is the day

Java is parsing the date string for you by breaking it down into a predefined template of parts and recognizing the parts. Then the the parse function outputs a date object, which is easier to store and manipulate than a date string.

5. Task Realization:

```
import imaplib
import email
import getpass
import re
import base64
import smtplib
import ssl
import sys
from colors import *
from email.mime.text import MIMEText

user_email = 'networkprogramming.utm.sa@gmail.com'
user_password = 'network2018'
gmail_connected = False
menu_count = -1
mail = imaplib.IMAP4_SSL('imap.gmail.com')

def encoded_words_to_text(encoded_words):
    try:
        encoded_word_regex =
r'=\?{1}(.+)\?{1}([B|Q])\?{1}(.+)\?{1}='
        charset, encoding, encoded_text =
re.match(encoded_word_regex, encoded_words).groups()
        if encoding is 'B':
```

```
            byte_string = base64.b64decode(encoded_text)
        elif encoding is 'Q':
            byte_string = quopri.decodestring(encoded_text)
        return byte_string.decode(charset)
    except Exception:
        return encoded_words
```

We are making secure login here with gmail imap4 ssl and we are encoding word to text from gmail style.

############## GMAIL LOGİN AND CONNECTION VIA IMAP################

```
def input_user_credentials():
    global user_email, user_password
    sys.stdout.write(BOLD + BLUE)
    print "######### PLEASE INTRODUCE YOUR LOGIN INFO #########"
    sys.stdout.write(GREEN)
    print "YOUR GMAİL ADDRESS : "
    user_email = raw_input()
    sys.stdout.write(CYAN)
    print "YOUR GMAİL PASSWORD : "
    user_password = getpass.getpass()
    connect_gmail_imap()

def connect_gmail_imap():
    global mail, gmail_connected
    mail.login(user_email, user_password)
    mail.select("INBOX")
    result2, new_messages = mail.search(None, '(UNSEEN)')
    gmail_connected = True
    print
    sys.stdout.write(RED)
    print "Hello, " + user_email
    sys.stdout.write(BLUE)
    print "You have  " + str(len(new_messages[0].split())),"unread
messages."
```

We are asking gmail address and password in order to login gmail account.

############## SEND AN EMAIL #################

```
def send_simple_mail():
    global mail
    print 'YOUR EMAIL MESSAGE: '
    msg = MIMEText(raw_input())
    print 'YOUR EMAIL SUBJECT: '
    msg['Subject'] = raw_input()
    print 'RECEIVER ADDRESS : '
```

```
    to_email = raw_input()
    msg['To'] = to_email
    print 'CC (IF YOU DO NOT WANT TO SEND PLEASE LEAVE IT ALONE) : '
    msg['CC'] = raw_input()
    msg['From'] = user_email

    server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
    server.login(user_email, user_password)
    server.sendmail(user_email, to_email, msg.as_string())
    server.quit()
```

We are sending simple email with this method, we are asking email content(message), subject and receiver and as a option choice we are asking CC info.

############## OPEN AN EMAIL ##################

```
def open_an_email():
    global mail
    esult, data = mail.uid('search', 'CHARSET', 'UTF-8', "ALL")
    list_of_ids = data[0].split()
    last_count_of_email = len(list_of_ids) - 1
    print 'HOW MANY EMAIL DO YOU WANT TO OPEN? : '
    mail_number = int(input()) - 1
    current_email_uid = list_of_ids[last_count_of_email -
mail_number]
    result2, message_fetch = mail.uid('fetch', current_email_uid,
'(RFC822)')
    raw_email = message_fetch[0][1]
    email_message = email.message_from_string(raw_email)
    print
    sys.stdout.write(CYAN)
    print 'SUBJECT: ' +
encoded_words_to_text(email_message['subject'])
    print 'SENDER: ' + email_message['from']
    print 'DATE: ' + email_message['Date']

    attachment_count = 0

    for part in email_message.walk():
        if part.get('Content-Disposition') is None:
```

```
            continue
        attachment_count += 1

    if attachment_count != 0:
        print str(attachment_count) + ' ATTACHMENT FOUND'

    if email_message.is_multipart():
        for part in email_message.get_payload():

            text = None

            if part.get_content_charset() is None:
                # We cannot know the character set, so return
decoded "something"
                text = part.get_payload(decode=True)
                continue

            charset = part.get_content_charset()

            if part.get_content_type() == 'text/plain':
                text = unicode(part.get_payload(decode=True),
str(charset), "ignore").encode('utf8', 'replace')
            if part.get_content_type() == 'text/html':
                html = unicode(part.get_payload(decode=True),
str(charset), "ignore").encode('utf8', 'replace')
            if text is not None:
                print
                print 'CONTENT: ' + text.strip()
                print
    else:
        text = unicode(email_message.get_payload(decode=True),
email_message.get_content_charset(), 'ignore').encode('utf8',
'replace')
        print 'CONTENT: ' + text.strip()
```

We are asking first how many email they want to open in the same moment for example if they input 2 then we are showing lasts 2 email order by date.
If there is any attachment with those emails we are also giving information about that.

############## GET LAST N EMAILS #################

```
def get_last_n_messages():
    global mail
    print 'HOW MANY EMAILS DO YOU WANT TO FETCH? : '
    n = int(input())
    result, data = mail.uid('search', 'CHARSET', 'UTF-8', "ALL")
    list_of_ids = data[0].split()
    last_count_of_email = len(list_of_ids) - 1
    for i in range(n):
        current_email_uid = list_of_ids[last_count_of_email - i]
        result2, message_fetch = mail.uid('fetch',
current_email_uid, '(RFC822)')
        raw_email = message_fetch[0][1]
```

```
        email_message = email.message_from_string(raw_email)
        print
        sys.stdout.write(RED)
        print 'SUBJECT: ' +
encoded_words_to_text(email_message['subject'])
        print 'SENDER: ' + email_message['from']
        print 'DATE: ' + email_message['Date']

        attachment_count = 0

        for part in email_message.walk():
            if part.get('Content-Disposition') is None:
                continue
            attachment_count += 1

        if attachment_count != 0:
            print str(attachment_count) + ' attachments found'

        if email_message.is_multipart():
            for part in email_message.get_payload():

                text = None

                if part.get_content_charset() is None:
                    # We cannot know the character set, so return
decoded "something"
                    text = part.get_payload(decode=True)
                    continue

                if part.get_content_type() == 'text/plain':
                    charset = part.get_content_charset()
                    text = unicode(part.get_payload(decode=True),
str(charset), "ignore").encode('utf8', 'replace')
                if text is not None:
                    print
                    print 'First sentence: ' +
text.strip().split('\n')[0]
                    print
```

We are asking here how many email they want to see and we are showing up lastest emails.

```
###################### MENU ##########################

while menu_count != 0:
    print
    if gmail_connected == False:
        sys.stdout.write(BOLD + BLUE)
        print "######### WELCOME TO GMAIL LOGIN #########"
        sys.stdout.write(RED)
        print "DO YOU WANT TO CONNECT TO YOUR GMAİL ACCOUNT?(YES=1 /
NO=0)"
```

```
elif gmail_connected == True:
    sys.stdout.write(REVERSE)
    print "2. TO GET LAST EMAILS"
    print "3. TO SEND AN EMAIL"
    print "4. TO READ EMAIL"
print
print
menu_count = int(input())

if gmail_connected == False:
    if menu_count == 1:
        input_user_credentials()
elif gmail_connected == True:
    if menu_count == 2:
        get_last_n_messages()
    elif menu_count == 3:
        send_simple_mail()
    elif menu_count == 4:
        open_an_email()
```

6. Outputs:

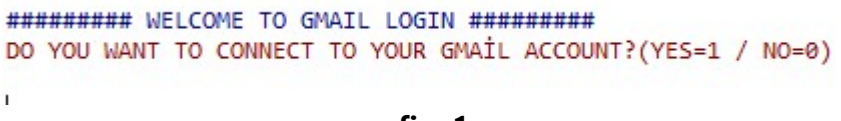

fig. 1

On figure 1 as we can see when we run the program it is asking us do you want to connect to you gmail account, so if we press the 1 then will appear next on figure 2

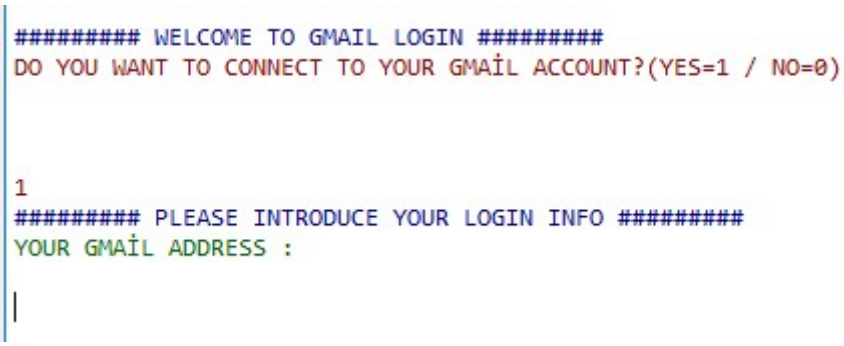

fig.2

After accepting login its asks gmail address and password on figure 2 and 3

```
######### PLEASE INTRODUCE YOUR LOGIN INFO #########
YOUR GMAİL ADDRESS :

networkprogramming.utm.sa@gmail.com
YOUR GMAİL PASSWORD :

Warning: QtConsole does not support password mode, the text you type will be visible.
network2018
```

fig. 3

```
Hello, networkprogramming.utm.sa@gmail.com
You have  1  unread messages.

2. TO GET LAST EMAILS
3. TO SEND AN EMAIL
4. TO READ EMAIL
```

fig. 4

After successfully login we can see our menu and we can see how many emails we have unread.

```
HOW MANY EMAILS DO YOU WANT TO FETCH? :

1

SUBJECT: Re: hii
SENDER: Sezer Aksoy <aksoysezer95@gmail.com>
DATE: Thu, 03 May 2018 08:06:57 +0000

First sentence: Response


2. TO GET LAST EMAILS
3. TO SEND AN EMAIL
4. TO READ EMAIL
```

fig. 5

```
3
YOUR EMAIL MESSAGE:

aksoysezer95@gmail.com
YOUR EMAIL SUBJECT:

Test on 09.05.2018
RECEIVER ADDRESS :

aksoysezer95@gmail.com
CC (IF YOU DO NOT WANT TO SEND PLEASE LEAVE IT ALONE) :
```

fig. 6

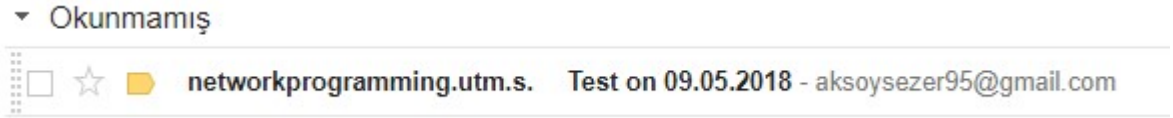

fig. 7

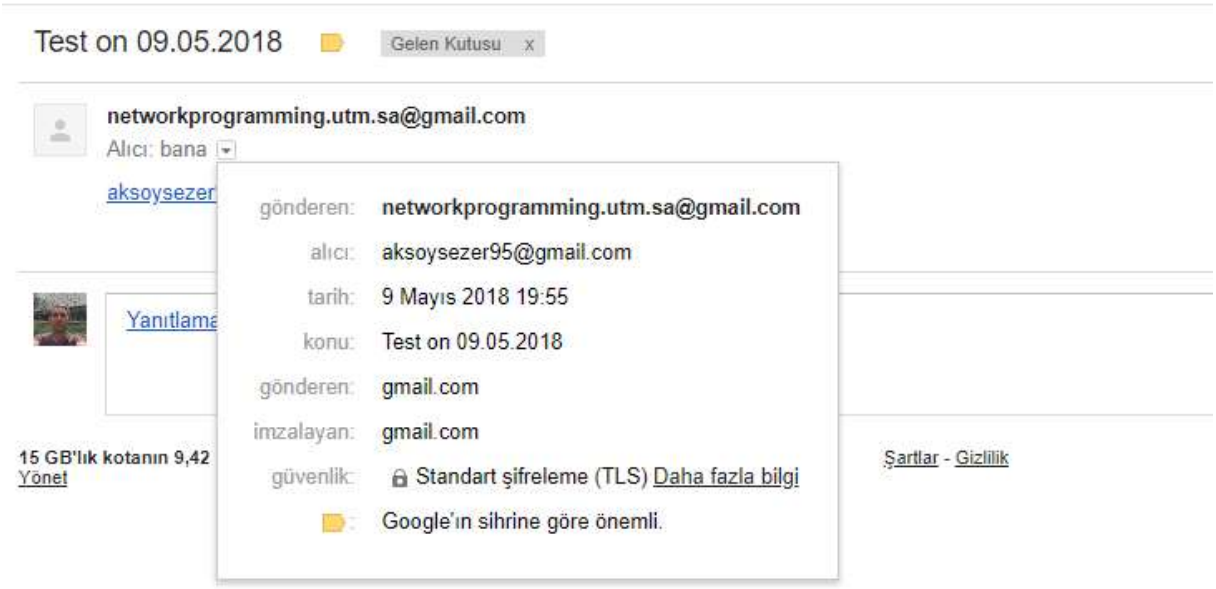

fig. 8

Conclusion

This laboratory work is useful because we learnt about the basics of network programming.

Specially, email client- are great for fetching data.

I made my email client software so I can easily manage my gmail account with my program.

Bibliography

https://pythonprogramminglanguage.com/read-gmail-using-python/
https://docs.python.org/2/library/email-examples.html
http://www.pythonforbeginners.com/code-snippets-source-code/using-python-to-send-email