МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ
ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчет по лабораторной №2

Почтовые протоколы

по дисциплине «Протоколы компьютерного взаимодействия»

Выполнил   студент гр. ФИБ-4301-51-00   __________________ Кочкин В.Р.

Проверил   преподаватель каф. ПМиИ   __________________ Белиц А.Б.

Киров 2022

Цель работы

Познакомиться с почтовыми протоколами и реализовать получение и отправку сообщений.

Задания

Задание 1

Установите на виртуальную машину почтовый сервер, например, hMailServer (https://www.hmailserver.com/download). Настройте его для работы в режиме сети виртуального адаптера хоста. Дайте название домену (ветка Domains). Создайте почтовый аккаунт (ветка Accounts). Аутентификацию почтового сервера настройте без использования сертификатов.

Задание 2

Проверьте работоспособность почтового сервера используя клиент telnet (telnet или PuTTY).

Задание 3

В ходе лабораторной работы необходимо разработать программу на языке python, которая реализует функции отправки текстовых почтовых сообщений по протоколу SMTP.

Задание 4

В ходе лабораторной работы необходимо разработать программу на языке python которая реализует функции получения почтовых текстовых сообщений по протоколу POP3 или протоколу IMAP (на выбор).

Задание 5

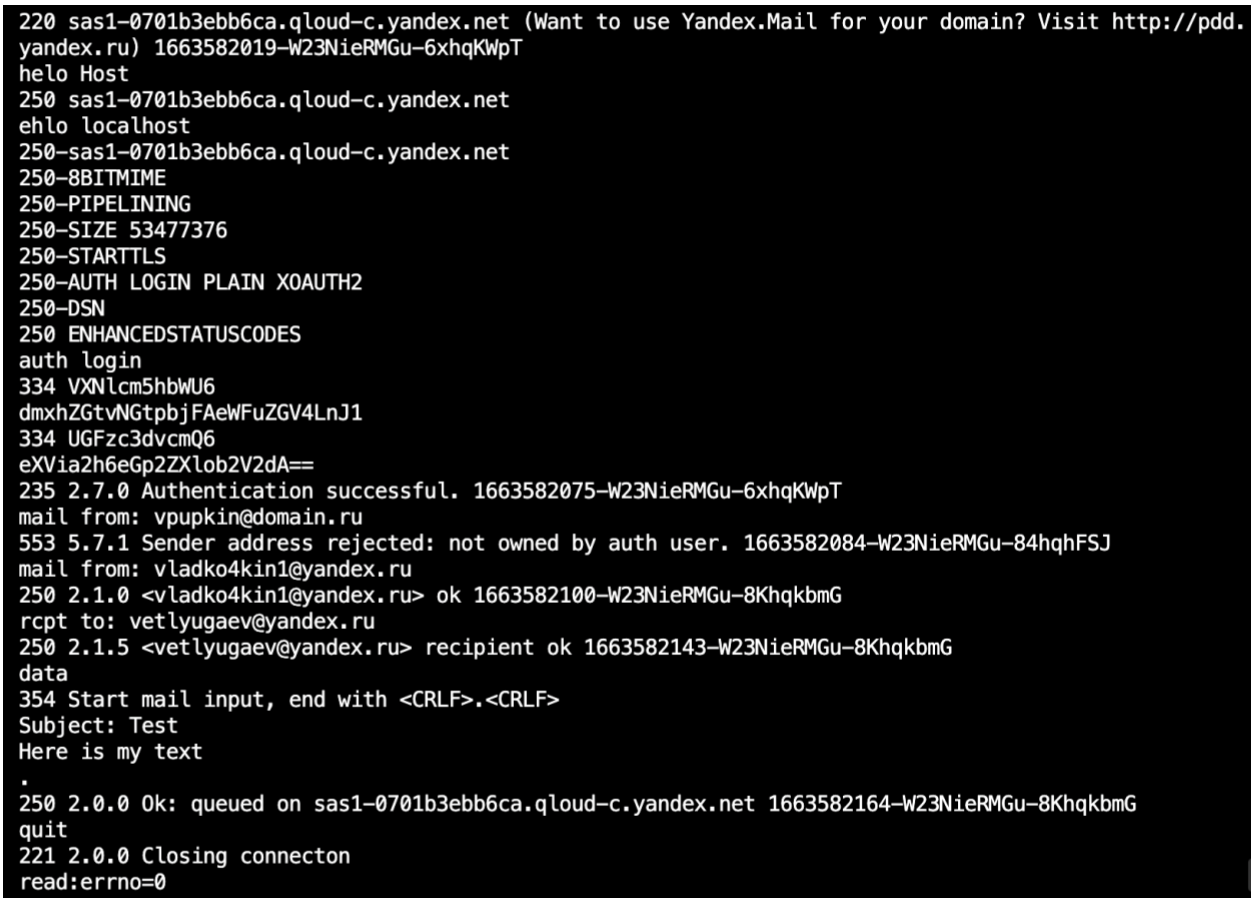
В ходе лабораторной работы необходимо разработать программу на языке python реализующую взаимодействие по протоколам POP3 и SMTP при помощи сокетов.

Полученные результаты

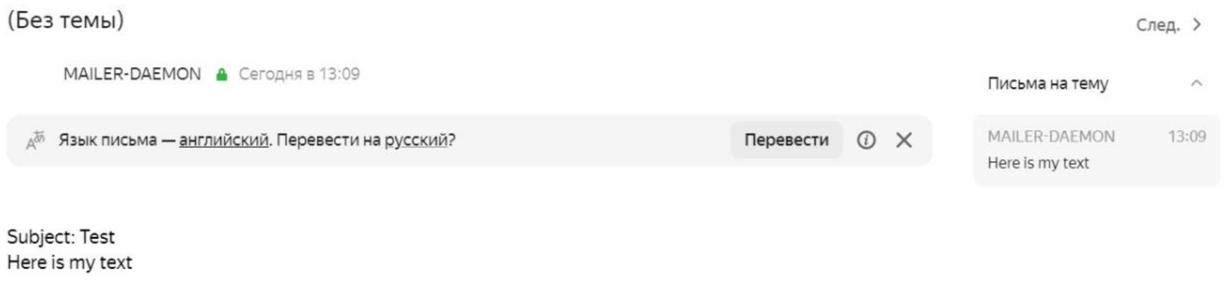
Задание 1

Сначала предполагалось использование почтового сервера Яндекса, но потом для упрощения работы был использован почтовый сервер одногруппника без сертификатов.

Задание 2



```
220 sas1-0701b3ebb6ca.qloud-c.yandex.net (Want to use Yandex.Mail for your domain? Visit http://pdd.
yandex.ru) 1663582019-W23NieRMGu-6xhqKWpT
helo Host
250 sas1-0701b3ebb6ca.qloud-c.yandex.net
ehlo localhost
250-sas1-0701b3ebb6ca.qloud-c.yandex.net
250-8BITMIME
250-PIPELINING
250-SIZE 53477376
250-STARTTLS
250-AUTH LOGIN PLAIN XOAUTH2
250-DSN
250 ENHANCEDSTATUSCODES
auth login
334 VXNlcm5hbWU6
dmxhZGtvNGtpbjFAeWFuZGV4LnJ1
334 UGFzc3dvcmQ6
eXVia2h6eGp2ZXlob2V2dA==
235 2.7.0 Authentication successful. 1663582075-W23NieRMGu-6xhqKWpT
mail from: vpupkin@domain.ru
553 5.7.1 Sender address rejected: not owned by auth user. 1663582084-W23NieRMGu-84hqhFSJ
mail from: vladko4kin1@yandex.ru
250 2.1.0 <vladko4kin1@yandex.ru> ok 1663582100-W23NieRMGu-8KhqkbmG
rcpt to: vetlyugaev@yandex.ru
250 2.1.5 <vetlyugaev@yandex.ru> recipient ok 1663582143-W23NieRMGu-8KhqkbmG
data
354 Start mail input, end with <CRLF>.<CRLF>
Subject: Test
Here is my text
.
250 2.0.0 Ok: queued on sas1-0701b3ebb6ca.qloud-c.yandex.net 1663582164-W23NieRMGu-8KhqkbmG
quit
221 2.0.0 Closing connecton
read:errno=0
```



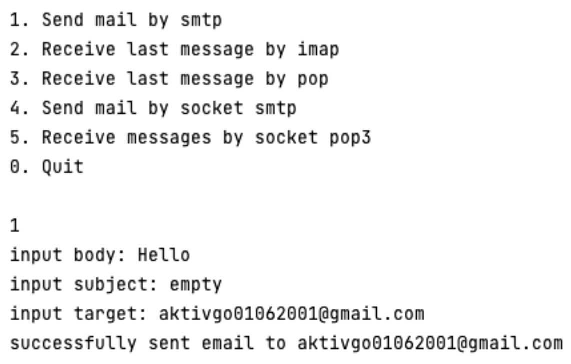
(Без темы)

MAILER-DAEMON Сегодня в 13:09

Язык письма — английский. Перевести на русский?

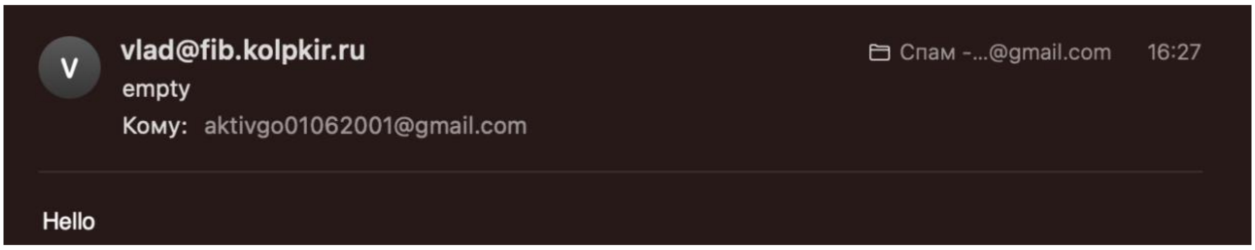
Subject: Test
Here is my text

Задание 3



```
1. Send mail by smtp
2. Receive last message by imap
3. Receive last message by pop
4. Send mail by socket smtp
5. Receive messages by socket pop3
0. Quit

1
input body: Hello
input subject: empty
input target: aktivgo01062001@gmail.com
successfully sent email to aktivgo01062001@gmail.com
```



vlad@fib.kolpkir.ru

empty

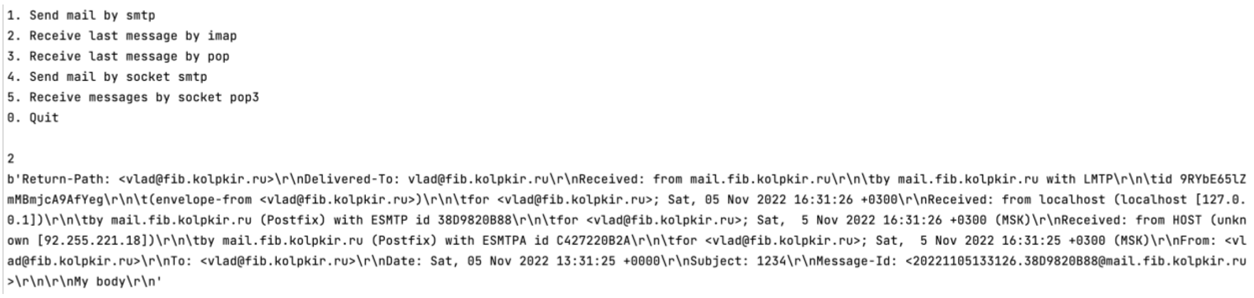
Кому: aktivgo01062001@gmail.com

Спам -...@gmail.com 16:27

Hello

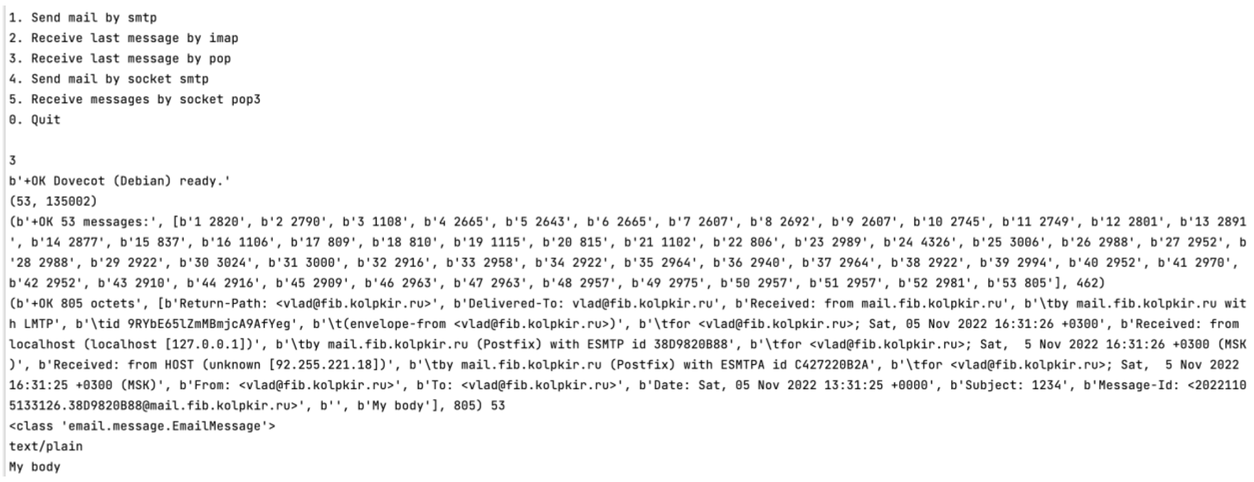
Листинг программы приведен в [приложении А Задание 3](#).

Задание 4



```
1. Send mail by smtp
2. Receive last message by imap
3. Receive last message by pop
4. Send mail by socket smtp
5. Receive messages by socket pop3
0. Quit

2
b'Return-Path: <vlad@fib.kolpkir.ru>\r\nDelivered-To: vlad@fib.kolpkir.ru\r\nReceived: from mail.fib.kolpkir.ru\r\n\tby mail.fib.kolpkir.ru with LMTP\r\n\tid 9RYbE65lZ
mMBmjcA9AfYeg\r\n\t(envelope-from <vlad@fib.kolpkir.ru>)\r\n\tfor <vlad@fib.kolpkir.ru>; Sat, 05 Nov 2022 16:31:26 +0300\r\nReceived: from localhost (localhost [127.0.
0.1])\r\n\tby mail.fib.kolpkir.ru (Postfix) with ESMTP id 38D9820B88\r\n\tfor <vlad@fib.kolpkir.ru>; Sat,  5 Nov 2022 16:31:26 +0300 (MSK)\r\nReceived: from HOST (unkn
own [92.255.221.18])\r\n\tby mail.fib.kolpkir.ru (Postfix) with ESMTPA id C427220B2A\r\n\tfor <vlad@fib.kolpkir.ru>; Sat,  5 Nov 2022 16:31:25 +0300 (MSK)\r\nFrom: <vl
ad@fib.kolpkir.ru>\r\nTo: <vlad@fib.kolpkir.ru>\r\nDate: Sat, 05 Nov 2022 13:31:25 +0000\r\nSubject: 1234\r\nMessage-Id: <20221105133126.38D9820B88@mail.fib.kolpkir.ru
>\r\n\r\nMy body\r\n'
```



```
1. Send mail by smtp
2. Receive last message by imap
3. Receive last message by pop
4. Send mail by socket smtp
5. Receive messages by socket pop3
0. Quit

3
b'+OK Dovecot (Debian) ready.'
(53, 135002)
(b'+OK 53 messages:', [b'1 2820', b'2 2790', b'3 1108', b'4 2665', b'5 2643', b'6 2665', b'7 2607', b'8 2692', b'9 2607', b'10 2745', b'11 2749', b'12 2801', b'13 2891
', b'14 2877', b'15 837', b'16 1106', b'17 809', b'18 810', b'19 1115', b'20 815', b'21 1102', b'22 806', b'23 2989', b'24 4326', b'25 3006', b'26 2988', b'27 2952', b
'28 2988', b'29 2922', b'30 3024', b'31 3000', b'32 2916', b'33 2958', b'34 2922', b'35 2964', b'36 2940', b'37 2964', b'38 2922', b'39 2994', b'40 2952', b'41 2970',
b'42 2952', b'43 2910', b'44 2916', b'45 2909', b'46 2963', b'47 2963', b'48 2957', b'49 2975', b'50 2957', b'51 2957', b'52 2981', b'53 805'], 462)
(b'+OK 805 octets', [b'Return-Path: <vlad@fib.kolpkir.ru>', b'Delivered-To: vlad@fib.kolpkir.ru', b'Received: from mail.fib.kolpkir.ru', b'\tby mail.fib.kolpkir.ru wit
h LMTP', b'\tid 9RYbE65lZmMBmjcA9AfYeg', b'\t(envelope-from <vlad@fib.kolpkir.ru>)', b'\tfor <vlad@fib.kolpkir.ru>; Sat, 05 Nov 2022 16:31:26 +0300', b'Received: from
localhost (localhost [127.0.0.1])', b'\tby mail.fib.kolpkir.ru (Postfix) with ESMTP id 38D9820B88', b'\tfor <vlad@fib.kolpkir.ru>; Sat,  5 Nov 2022 16:31:26 +0300 (MSK
)', b'Received: from HOST (unknown [92.255.221.18])', b'\tby mail.fib.kolpkir.ru (Postfix) with ESMTPA id C427220B2A', b'\tfor <vlad@fib.kolpkir.ru>; Sat,  5 Nov 2022
16:31:25 +0300 (MSK)', b'From: <vlad@fib.kolpkir.ru>', b'To: <vlad@fib.kolpkir.ru>', b'Date: Sat, 05 Nov 2022 13:31:25 +0000', b'Subject: 1234', b'Message-Id: <2022110
5133126.38D9820B88@mail.fib.kolpkir.ru>', b'', b'My body'], 805) 53
<class 'email.message.EmailMessage'>
text/plain
My body
```

Листинг программы приведен в [приложении А Задание 4](#).

Задание 5

```
1. Send mail by smtp
2. Receive last message by imap
3. Receive last message by pop
4. Send mail by socket smtp
5. Receive messages by socket pop3
0. Quit

4
input body: My body
input subject: 1234
input target: vlad@fib.kolpkir.ru
After MAIL FROM command: 250 2.1.0 Ok

After RCPT TO command: 250 2.1.5 Ok

After DATA command: 354 End data with <CR><LF>.<CR><LF>

Response after sending message body: 250 2.0.0 Ok: queued as C427220B2A

successfully sent email to vlad@fib.kolpkir.ru
```

```
1. Send mail by smtp
2. Receive last message by imap
3. Receive last message by pop
4. Send mail by socket smtp
5. Receive messages by socket pop3
0. Quit

5
+OK 52 messages:
1 2820
2 2790
3 1108
4 2665
```

```
52
+OK 805 octets
Return-Path: <vlad@fib.kolpkir.ru>
Delivered-To: vlad@fib.kolpkir.ru
Received: from mail.fib.kolpkir.ru
        by mail.fib.kolpkir.ru with LMTP
        id 9RYbE65lZmMBmjcA9AfYeg
        (envelope-from <vlad@fib.kolpkir.ru>)
        for <vlad@fib.kolpkir.ru>; Sat, 05 Nov 2022 16:31:26 +0300
Received: from localhost (localhost [127.0.0.1])
        by mail.fib.kolpkir.ru (Postfix) with ESMTP id 38D9820B88
        for <vlad@fib.kolpkir.ru>; Sat,  5 Nov 2022 16:31:26 +0300 (MSK)
Received: from HOST (unknown [92.255.221.18])
        by mail.fib.kolpkir.ru (Postfix) with ESMTPA id C427220B2A
        for <vlad@fib.kolpkir.ru>; Sat,  5 Nov 2022 16:31:25 +0300 (MSK)
From: <vlad@fib.kolpkir.ru>
To: <vlad@fib.kolpkir.ru>
Date: Sat, 05 Nov 2022 13:31:25 +0000
Subject: 1234
Message-Id: <20221105133126.38D9820B88@mail.fib.kolpkir.ru>

My body
.
```

Листинг программы приведен в приложении А Задание 5.

Вывод

В ходе лабораторной работы я познакомился с почтовыми протоколами и реализовал получение и отправку сообщений.

Приложения

Приложение А. Листинги программ

Точка входа.

main.py

```
import os
import time

from smtp.client import SMTPClient
from imap.client import IMAPClient
from pop3.client import POP3Client
from smtp.socket_client import SocketSmtpClient
from pop3.socket_client import SocketPop3Client

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

HOST = os.getenv('HOST')


def get_auth():
    return {
        'login': os.getenv('EMAIL_LOGIN'),
        'password': os.getenv('EMAIL_PASSWORD')
    }


def create_smtp_message():
    msg = MIMEMultipart()

    body = input("input body: ")

    msg['From'] = os.getenv("EMAIL_LOGIN")
    msg['Subject'] = input("input subject: ")

    msg["Date"] = time.strftime("%a, %d %b %Y %H:%M:%S %z")

    msg['To'] = input("input target: ")

    msg.attach(MIMEText(body, 'plain'))

    return msg


def create_socket_smtp_message():
    msg = {}

    body = input("input body: ")
    msg['body'] = body

    msg['from'] = os.getenv("EMAIL_LOGIN")
    msg['subject'] = input("input subject: ")
    msg['to'] = input("input target: ")
```

```python
    return msg


if __name__ == '__main__':
    auth = get_auth()
    smtp = SMTPClient(HOST, auth)
    imap = IMAPClient(HOST, auth)
    pop3 = POP3Client(HOST, auth)
    socket_smtp = SocketSmtpClient(HOST, 25, auth)
    socket_pop3 = SocketPop3Client(HOST, 110, auth)

    while True:
        print(
            '1. Send mail by smtp\n' +
            '2. Receive last message by imap\n' +
            '3. Receive last message by pop\n' +
            '4. Send mail by socket smtp\n' +
            '5. Receive messages by socket pop3\n' +
            '0. Quit\n'
        )

        choose = int(input())

        if choose == 1:
            message = create_smtp_message()
            smtp.send_mail(message)
            print("successfully sent email to", message['To'], '\n')
        elif choose == 2:
            print(imap.receive_last_mail(), '\n')
        elif choose == 3:
            print(pop3.receive_last_mail(), '\n')
        elif choose == 4:
            message = create_socket_smtp_message()
            socket_smtp.send_mail(message)
            print("successfully sent email to", message['to'], '\n')
        elif choose == 5:
            while True:
                print(socket_pop3.list())
                choose = int(input())
                if choose == 0:
                    break
                print(socket_pop3.receive_mail(choose))
        else:
            smtp.close()
            pop3.close()
            socket_smtp.close()
            socket_pop3.close()
            exit()
```

Задание 3.

```python
import smtplib


class SMTPClient:
    def __init__(self, host: str, auth: []):
        self.smtp = smtplib.SMTP(host)
        self.smtp.login(auth['login'], auth['password'])

    def send_mail(self, message):
        self.smtp.sendmail(message['From'], message['To'],
message.as_string())

    def close(self):
        self.smtp.close()
```

Задание 4.

IMAP:

```
import imaplib


class IMAPClient:
    def __init__(self, host: str, auth: []):
        self.imap = imaplib.IMAP4(host)
        self.imap.login(auth['login'], auth['password'])

    def receive_last_mail(self):
        self.imap.list()
        self.imap.select("inbox")

        result, data = self.imap.uid('search', "ALL")

        latest_email_uid = data[0].split()[-1]
        result, data = self.imap.uid('fetch', latest_email_uid, '(RFC822)')

        return data[0][1]
```

POP3:

```
import poplib

from email.parser import BytesParser
from email.policy import default


class POP3Client:
    def __init__(self, host: str, auth: []):
        self.pop3 = poplib.POP3(host)
        self.pop3.user(auth['login'])
        self.pop3.pass_(auth['password'])

    def receive_last_mail(self):
        print(self.pop3.getwelcome())

        stat = self.pop3.stat()
        print(stat)

        l = self.pop3.list()
        print(l)

        r = self.pop3.retr(len(l[1]))
        print(r, len(l[1]))

        bp = BytesParser(policy=default).parsebytes(b'\r\n'.join(r[1]))
        print(type(bp))

        for part in bp.walk():
            print(part.get_content_type())
            if part.get_content_maintype() == 'text':
                return part.get_content()

    def close(self):
        self.pop3.close()
```

Задание 5.

POP3:

```
from socket import *
import base64
import time


class SocketPop3Client:
    def __init__(self, host: str, port: int, auth: []):
        self.client_socket = socket(AF_INET, SOCK_STREAM)
        self.__initialize__(host, port)
        self.__authorize__(auth)

    def __initialize__(self, host: str, port: int):
        self.client_socket.connect((host, port))
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '+OK':
            raise '+OK reply not received from server.'

    def __authorize__(self, auth: []):
        user_msg = 'USER %s\r\n' % auth['login']
        self.client_socket.send(user_msg.encode())
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '+OK':
            raise '+OK reply not received from server.'

        pass_msg = 'PASS %s\r\n' % auth['password']
        self.client_socket.send(pass_msg.encode())
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '+OK':
            raise '+OK reply not received from server.'

    def list(self):
        list_msg = 'LIST\r\n'
        self.client_socket.send(list_msg.encode())
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '+OK':
            raise '+OK reply not received from server.'
        return recv

    def receive_mail(self, index: int):
        retr_msg = 'RETR %d\r\n' % index
        self.client_socket.send(retr_msg.encode())
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '+OK':
            raise '+OK reply not received from server.'
        return recv

    def close(self):
        q = "QUIT\r\n"
        self.client_socket.send(q.encode())
        self.client_socket.close()
```

SMTP:

```
from socket import *
import base64
import time


class SocketSmtpClient:
    def __init__(self, host: str, port: int, auth: []):
        self.client_socket = socket(AF_INET, SOCK_STREAM)
        self.__initialize__(host, port)
        self.__authorize__(auth)

    def __initialize__(self, host: str, port: int):
        self.client_socket.connect((host, port))
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '220':
            raise '220 reply not received from server.'

        helo_command = 'EHLO HOST\r\n'
        self.client_socket.send(helo_command.encode())
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '250':
            raise '250 reply not received from server.'

    def __authorize__(self, auth: []):
        base64_str = ('\x00' + auth['login'] + "\x00" +
auth['password']).encode()
        base64_str = base64.b64encode(base64_str)
        auth_msg = 'AUTH PLAIN '.encode() + base64_str + '\r\n'.encode()
        self.client_socket.send(auth_msg)
        recv = self.client_socket.recv(1024).decode()
        if recv[:3] != '235':
            raise '235 reply not received from server.'

    def send_mail(self, message):
        mail_from = 'MAIL FROM:<%s>\r\n' % message['from']
        self.client_socket.send(mail_from.encode())
        recv = self.client_socket.recv(1024).decode()
        print('After MAIL FROM command: ' + recv)

        rcpt_to = 'RCPT TO:<%s>\r\n' % message['to']
        self.client_socket.send(rcpt_to.encode())
        recv = self.client_socket.recv(1024).decode()
        print('After RCPT TO command: ' + recv)

        data = 'DATA\r\n'
        self.client_socket.send(data.encode())
        recv = self.client_socket.recv(1024).decode()
        print('After DATA command: ' + recv)

        _from = 'From: <%s>\r\n' % message['from']
        self.client_socket.send(_from.encode())

        to = 'To: <%s>\r\n' % message['to']
        self.client_socket.send(to.encode())

        date = time.strftime('Date: %a, %d %b %Y %H:%M:%S +0000\r\n',
time.gmtime())
        self.client_socket.send(date.encode())

        subject = "Subject: %s\r\n" % message['subject']
        self.client_socket.send(subject.encode())

        body = message['body'] + '\r\n.\r\n'
        self.client_socket.send(body.encode())

        recv = self.client_socket.recv(1024)
        print('Response after sending message body: ' + recv.decode())

    def close(self):
        q = "QUIT\r\n"
        self.client_socket.send(q.encode())
        self.client_socket.close()
```