Сервер протокола SMTP

Задание

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции сервера протокола SMTP.

Приложение должно реализовывать следующие функции:

1) Поддержку хранения почтовых папок пользователей для какого-либо домена(например, test.ru)
2) Получение почты для поддерживаемого домена(test.ru)
3) Реализация функции ретрансляции почты для других доменов. Для этого с помощью DNS-запроса необходимо определить IP-адрес SMTP-сервера, отвечающего за искомый почтовый домен, (можно использовать компонент библиотекиInternetDirect TIdDNSResolver) и переслать по этому адресу необходимые сообщения
4) Одновременная работа с несколькими клиентами
5) Протоколирование соединения сервера с клиентом

Поддерживаемые команды

Разработанное приложение должно реализовывать следующие команды протокола SMTP:
• HELO – получение информации о домене пользователя
• MAIL FROM – получение адреса отправителя письма
• RCPT TO – получение адресата письма
• DATA – получение тела письма
• QUIT – завершение сеанса связи

Настройки приложения
Разработанное приложение должно предоставлять пользователю настройку следующих параметров:
1) задание номер порта сервера(по умолчанию- 25)
2) задание имени поддерживаемого почтового домена

Архитектура приложения

В приложении имеется класс SMTPServer, который взаимодействует с клиентом и реализует команды протокола SMTP. В этом классе реализованы следующие методы:
• fetch_command() - получает и обрабатывает команду от клиента
• send_response() - отвечает клиенту на команду
• store_msg() - хранит письма
• forward_msg() - отправляет письма
• route_mail() - управляет письмами (хранит и отправляет)
• def handle() - обработчик команд

Описание работы протокола

Основная задача протокола SMTP (Simple Mail Transfer Protocol) заключается в том, чтобы обеспечивать передачу электронных сообщений (почту). Для работы через протокол SMTP клиент создаёт TCP соединение с сервером через порт 25. Затем клиент и SMTP сервер обмениваются информацией пока соединение не будет закрыто или прервано. Основной процедурой в SMTP является передача почты (Mail Procedure). Далее идут процедуры форвардинга почты (Mail Forwarding), проверка имён почтового ящика и вывод списков почтовых групп. Самой первой процедурой является открытие канала передачи, а последней - его закрытие.

Команды SMTP указывают серверу, какую операцию хочет произвести клиент. Команды состоят из ключевых слов, за которыми следует один или более параметров. Ключевое слово состот из 4-х символов и разделено от аргумента одним или несколькими пробелами. Каждая командная строка заканчивается символами CRLF.

Дизайн протокола

При установлении соединения между клиентом и сервером, выводится сообщение:
220 Hello
После чего клиент может вводить команды согласно протоколу:
helo denis
250 DenisServer
и т.д.
При вводе команды quit соединение сбрасывается и выводится сообщение об удачной операции quit.

Описание команд протокола

Команда	Описание
HELO	Поле, обозначающее начало посылки письма. При нормальной работе в ответ сервер должен прислать сообщение, начинающееся с кода 250.
AUTH LOGIN	Поле, обозначающее начало передачи логина и пароля на SMTP - сервер. В случае успешной передачи сервер возвращает код 235 (авторизация пройдена).
MAIL	Поле, в котором указывается отправитель письма. Если отправитель

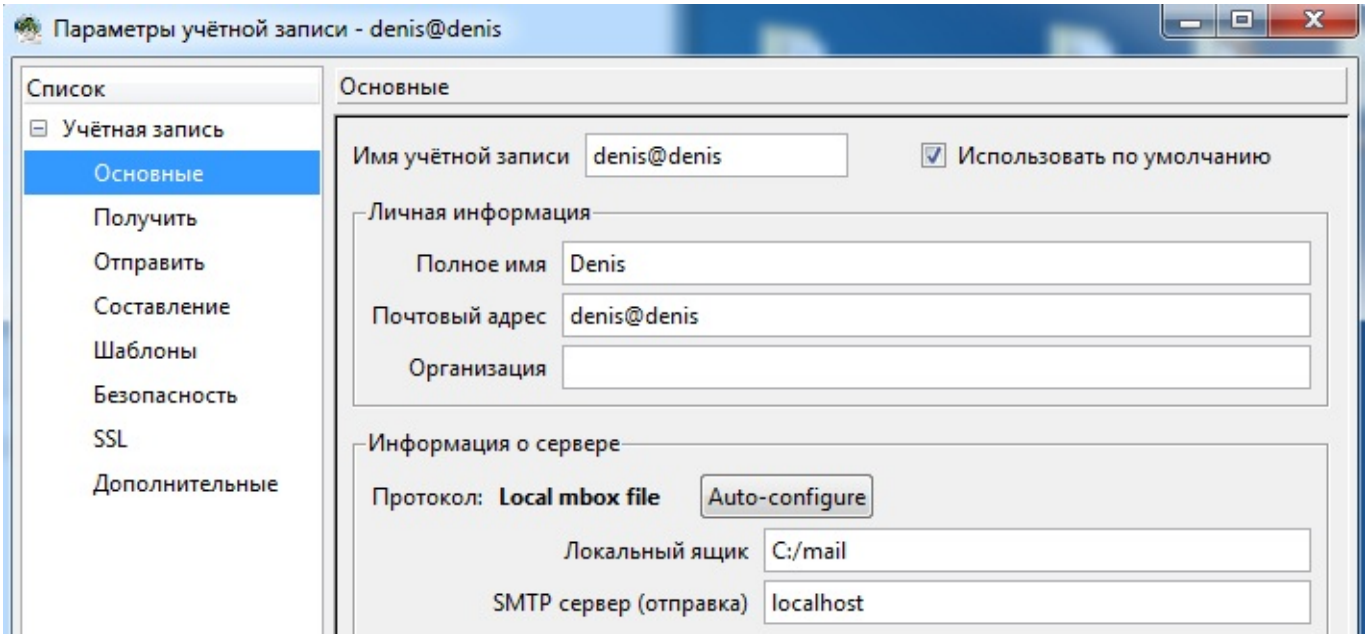
FROM:	корректен, то сервер должен прислать код 250.
RCPT TO:	поле, в котором указывается получатель письма (поле может дублироваться). Если получатель существует, то сервер возвращает код 250, в противном случае - 550.
DATA	Поле, сигнализирующее начало передачи текста сообщения. В ответ при нормальной работе сервер должен вернуть код 354 (знак того, что он готов принимать данные).
QUIT	Поле, которое указывает на то, то передача письма закончена. В случае успеха сервер должен вернуть код 221.

Описание среды разработки

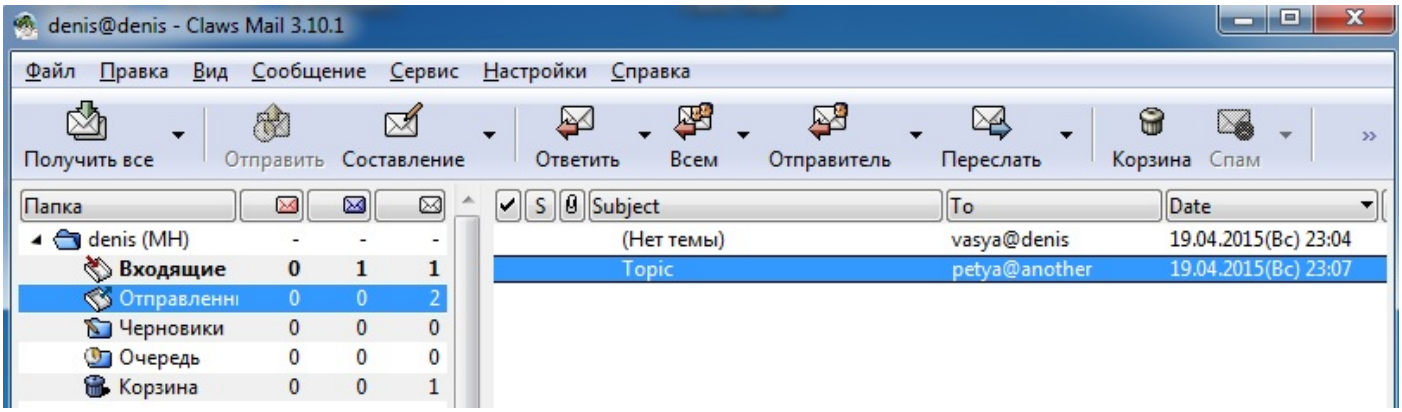
PyCharm 4.0.4
Python 2.7.9

Тестирование

Тестирование проводилось с помощью готового SMTP клиента Claws Mail 3.10.1. Создадим учетную запись и зададим номер порта (2525):



Зададим получателя, тему и текст сообщения:



Сообщение успешно отправилось, в хранилище писем C:\mail появилось новое письмо petya:
From MAILER-DAEMON
Date: Sun, 19 Apr 2015 23:07:29 +0300
From: Denis denis@denis
To: petya@another
Subject: Topic
Message-ID: 20150419230729.00002363@denis
X-Mailer: Claws Mail 3.10.1 (GTK+ 2.16.6; i586-pc-mingw32msvc)
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit

Topic!

Протоколирование между клиентом и сервером приведено ниже:

-> 220 Hello
<- helo [192.168.56.1]
-> 250 DenisServer
<- mail FROM:denis@denis
-> 250 OK
<- rcpt TO:petya@another
-> 250 OK
<- data
-> 354 Enter mail, end with “.” on a line by itself

Date: Sun, 19 Apr 2015 23:07:29 +0300 From: Denis denis@denis To: petya@another Subject: Topic Message-ID: 20150419230729.00002363@denis X-Mailer: Claws Mail 3.10.1 (GTK+ 2.16.6; i586-pc-mingw32msvc) MIME-Version: 1.0 Content-Type: text/plain; charset=US-ASCII Content-Transfer-Encoding: 7bit

Topic! .

-> 250 OK <- quit -> 221 Quit OK

Видно, SMTP сервер работает правильно.

Приложение. Код программы

```
import smtplib
import sys

__author__ = 'Denis'
import SocketServer
import logging


def start_server():
HOST, PORT = "127.0.0.1", int(sys.argv[2])

server = SocketServer.TCPServer((HOST, PORT), SMTPServer, False)
server.allow_reuse_address = True
server.server_bind()
server.server_activate()
return server

class SMTPServer(SocketServer.BaseRequestHandler):
server_name = sys.argv[1]
mail_dir = 'C:/mail'
server_addr = {'another': 'localhost'}
server_ports = {'another': 2526}

def fetch_command(self):
    buffer = ""
    while "\r\n" not in buffer:
        new_data = self.request.recv(1024)
        buffer += new_data
    print("<< " + buffer)
    return buffer.partition("\r\n")[0]

def send_response(self, code, message):
    response = str(code) + ' ' + message + '\r\n'
    print(">> " + response)
    self.request.send(response)

def store_msg(self, msg, user):
    f = open(self.mail_dir + '/' + user, 'a')
    f.write("From MAILER-DAEMON\r\n")
    f.write(msg + "\r\n")
    f.close()

def forward_msg(self, msg, addr, domain):
    if domain not in self.server_ports:
        print("Unknown server!")
        return
    server = smtplib.SMTP(self.server_addr[domain], self.server_ports[domain
    server.set_debuglevel(1)
    server.sendmail(self.from_addr, addr, msg)
    server.quit()

def route_mail(self, msg):
```

```
    for addr in self.to:
        user, sep, domain = addr.partition('@')
        if domain == self.server_name:
            self.store_msg(msg, user)
        else:
            self.forward_msg(msg, addr, domain)

def handle(self):
    self.send_response(220, 'Hello')

    self.from_addr = ""
    self.to = []

    while True:
        command = self.fetch_command()
        cmd_up = command.upper()
        if cmd_up.startswith('HELO'):
            self.send_response(250, 'DenisServer')
        elif cmd_up.startswith('AUTH LOGIN'):
            self.send_response(334, 'VXNlcm5hbWU6')
            self.fetch_command()
            self.send_response(334, 'UGFzc3dvcmQ6')
            self.fetch_command()
            self.send_response(235, '2.7.0 Authentication successful.')
        elif cmd_up.startswith('MAIL FROM'):
            self.from_addr = command.partition('<')[2].partition('>')[0]
            self.send_response(250, 'OK')
        elif cmd_up.startswith('RCPT TO:'):
            address = command.partition('<')[2].partition('>')[0]
            self.to.append(address)
            self.send_response(250, "OK")
        elif cmd_up == 'DATA':
            self.send_response(354, 'Enter mail, end with "." on a line by i
            data = ""
            while "\r\n.\r\n" not in data:
                data += self.request.recv(1024)
                print(data)
            msg = data.rpartition(".\r\n")[0]
            self.route_mail(msg)
            self.send_response(250, "OK")
        elif cmd_up == 'QUIT':
            self.send_response(221, "Quit OK")
            self.request.close()
            return
        else:
            self.send_response(502, "DenisServer Wrong Command!")

if __name__ == "__main__":
logging.basicConfig(level=logging.DEBUG)
server = start_server()
server.serve_forever()
```