CENTRO UNIVERSITÁRIO FEI

CIÊNCIA DA COMPUTAÇÃO

PROJETO: SERVIDOR SMTP

ANDY SILVA BARBOSA – RA: 22.218.025-9

RAFAEL ZACARIAS PALIERINI - RA: 22.218.030-9

RUBENS DE ARAÚJO RODRIGUES MENDES - RA: 22.218.009-3

VITOR ACOSTA DA ROSA – RA: 22.218.006-9

SÃO BERNARDO – SP

2020

1. INSTRUÇÕES PARA USO

O projeto contém um total de sete arquivos, dois deles são referentes aos clientes (**clientHardcoded.py** e **clientInput.py**), outros dois relativos aos servidores que receberão e tratarão as mensagens enviadas (ambos com os nomes **server.py**, mas em diretórios diferentes) e os três restantes são arquivos de extensões .jpeg, .txt e .gif para o envio como anexo.

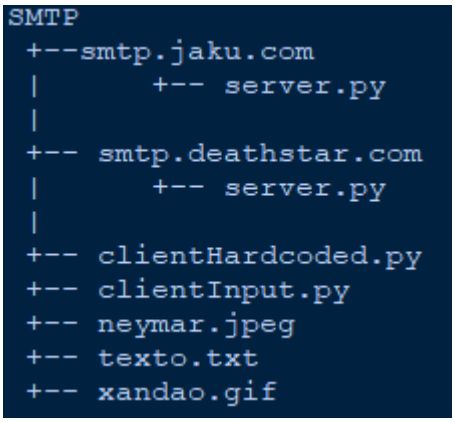

Figura 1 – Estruturação dos diretórios e arquivos do projeto

1.1. Preparando o ambiente

Antes de tudo, é de suma importância que o Tkinter esteja instalado em sua máquina. O Tkinter é um módulo padrão do Python para desenvolvimento de interfaces gráficas (ou *GUIs*). Esse módulo é utilizado no arquivo **clientInput.py** para a geração de um *filedialog*, uma janela que permite a abertura de arquivos.

Caso você não tenha certeza se esse módulo já está instalado em sua máquina, é possível testar utilizando as seguintes linhas de código.

```
import tkinter

tkinter._test()
```

Se o Tkinter estiver instalado em sua máquina, a janela representada na figura 2 irá aparecer.

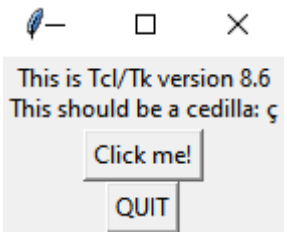

Figura 2 – Tkinter instalado na máquina com sucesso.

Caso apareça alguma mensagem de erro, você deve instalar o Tkinter em sua máquina. Para o sistema operacional Windows esse módulo já vem por padrão na instalação do Python, porém, caso encontre problemas ao executar o arquivo, tente utilizar no CMD o comando **pip install tk** através do pip (Packager Installer for Python), que também é disponível ao instalar o Python no computador.

Entretanto, se o sistema operacional for Linux, teremos uma sintaxe diferente, como por exemplo:

sudo apt install python3-tk - para distros derivadas de Debian (como o Ubuntu) ou **sudo pacman -S tk** - para distros como Manjaro, ArchLinux.

1.2. Ligando os servidores

Para a correta utilização e visualização do funcionamento, é necessário primeiramente que seja acessado os diretórios dos servidores (smtp.jaku.com e smtp.deathstar.com) e executado os arquivos **server.py** simultaneamente, e que esses permaneçam abertos durante a interação com o cliente. Dessa forma, os servidores poderão receber, tratar e responder as mensagens enviadas.

Perceba que, os diretórios smtp.jaku.com e smtp.deathstar.com conterão todos os anexos enviados e salvará todos os dados das mensagens (isso é, o remetente, destinatário(s), título, assunto, informações sobre os anexos e codificações) em um arquivo chamado **ServerMSG.txt**. Logo, para verificar o funcionamento correto, é possível acessar o diretório e avaliar se os anexos enviados pelos clientes estão lá e se as mensagens enviadas foram salvas corretamente.

Outra informação importante é:

o domínio jaku.com esperará mensagens no localhost:1025. E, o domínio deathstar.com esperará mensagens no localhost:1030.

1.3. Interação utilizando clientes

Perceba que, no diretório raiz, existem dois arquivos de extensão **.py**, um chamado **clientHarcoded.py** outro **clientInput.py**.

O primeiro, como o nome diz, trata-se de um arquivo com todos os dados do e-mail embutidos no código fonte, esse cliente se comunicará com o servidor smtp.jaku.com. Por padrão, o e-mail enviado por esse cliente será destinado à três usuários de domínios diferentes, a saber: jaku.com, deathstar.com e genshinimpact.com. Esse último, não possui um servidor próprio e existe somente para testar as respostas dos domínios caso tente ser contatado.

Além disso, três anexos padrão serão enviados: uma imagem jpeg, um arquivo de texto e um arquivo gif. Entretanto, é possível inserir mais anexos via código, precisamente, a partir da linha 83 do código há a instrução de como fazê-lo.

Já o segundo cliente, dado pelo arquivo **clientInput.py** permite que o usuário insira os dados do e-mail, o único dado fixo em código é o remetente (vader@deathstar.com). Ao executar esse arquivo, instruções serão mostradas via console esperando os dados. Esse cliente não possibilita a inserção de diversos destinatários de uma vez, caso o objetivo seja enviar para mais de um receptor, é necessário executar novamente o código.

O ponto interessante desse cliente é a possibilidade de escolher arquivos para enviar como anexo diretamente de sua máquina, sem a necessidade de estar no mesmo diretório do cliente, através de um *filedialog*.

O layout recomendado para acompanhar toda a transação de dados entre os servidores é manter os consoles no esquema da figura 3.

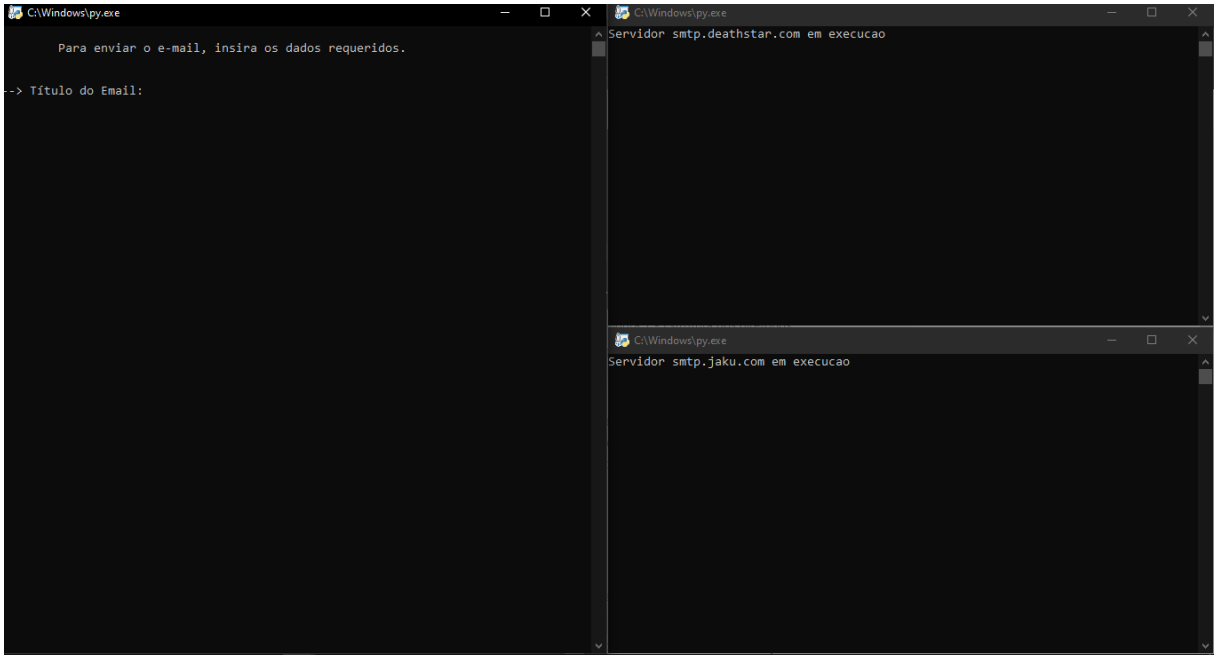

Figura 3 – Disposição dos consoles para visualização e inserção de dados

1.4. Algumas ideias para a execução

Após executar ambos os servidores, você pode testar o funcionamento de forma rápida executando o **clientHardcoded.py**.

Caso queira explorar todas funcionalidades do servidor, você pode utilizar o **clientInput.py** e testar o envio para o mesmo servidor (smtp.deathstar.com), para servidor com domínios diferentes e que são alcançáveis (smtp.jaku.com) e para servidores com domínios que estão fora de alcance (como o genshinimpact.com utilizado no **clientHardcoded.py**).

A figura 4 mostra um exemplo de comunicação entre os servidores smtp.deathstar.com e smtp.jaku.com com o envio de anexo.

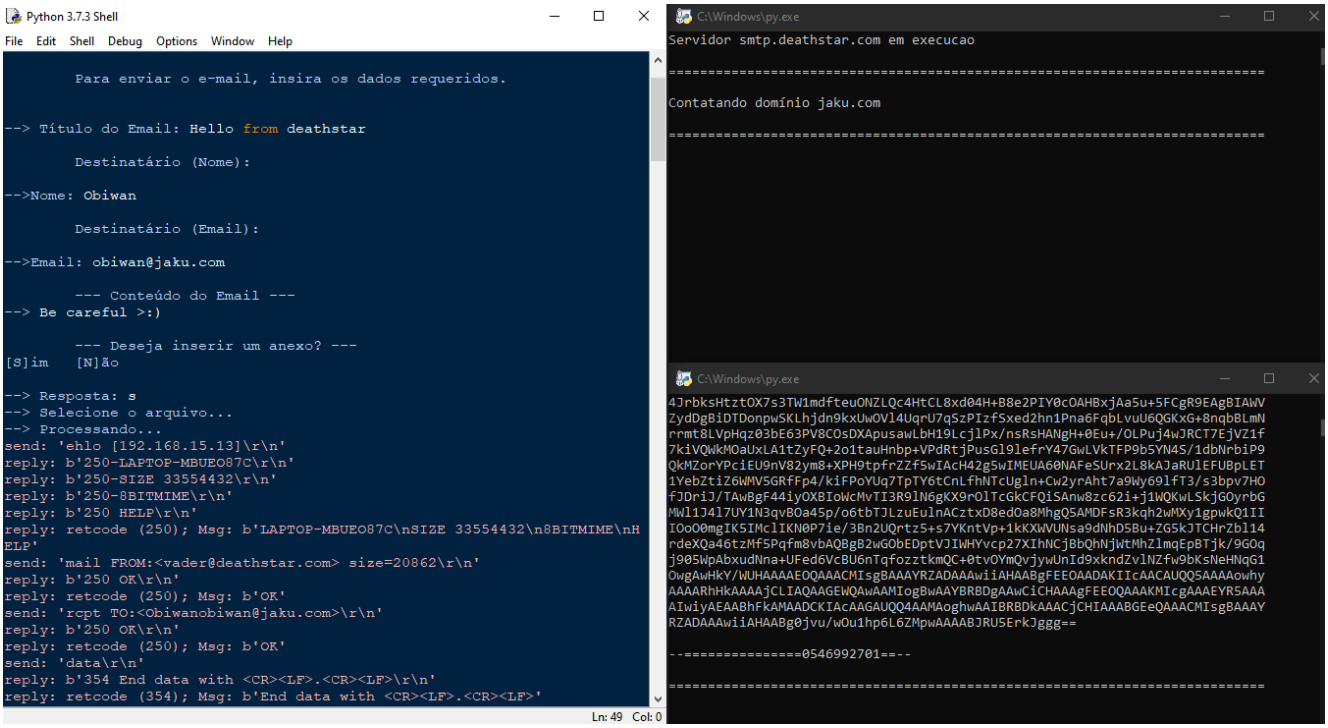

Figura 4 – Exemplo de funcionalidade entre cliente e servidor

Todos os dados enviados estarão presentes no diretório do servidor alvo, no exemplo, os dados estarão no diretório smtp.jaku.com.

2. CONSTRUÇÃO DO PROJETO

2.1. Clientes

O primeiro cliente **clientHardcoded.py** enviará a mensagem para o servidor smtp.jaku.com sem a necessidade de entrada de dados pelo usuário. O e-mail desse arquivo possui três destinatários: um do mesmo domínio (jaku.com) e dois de domínio diferente (deathstar.com e genshinimpact.com).

Os anexos desse cliente, por padrão são: uma imagem de extensão jpeg, um arquivo de texto e um gif.

O segundo cliente (clientInput.py) enviará a mensagem para o servidor smtp.deathstar.com, diferentemente do cliente hardcoded, esse contará com o usuário para preencher os dados da mensagem como o título, destinatário, assunto, conteúdo e até mesmo o anexo.

Note que, a mensagem enviada por esse cliente contará somente com um destinatário e um tipo de anexo por vez. Caso o usuário deseje enviar mais de um anexo é necessário executar novamente o clientInput.py.

2.2. Servidores

O servidor smtp.jaku.com ficará aguardando contato na porta 1025 do *localhost* as mensagens encaminhadas para esse servidor terão três respostas:

1. O destinatário pertence ao mesmo domínio, logo não há a necessidade de reenvio das informações e elas serão printadas no terminal que está executando esse servidor, além de salvar a mensagem no diretório do arquivo .py (bem como os anexos, se existirem).

2. O destinatário pertence ao domínio deathstar.com, nesse cenário, é contatado reencaminhado a mensagem para o servidor correto através do tratamento do domínio correto e as mensagens são reenviadas.

3. No último caso, o domínio de destino não é tratado pelo servidor e a mensagem não terá tratamento.

O servidor smtp.deathstar.com ficará aguardando contato na porta 1030 do *localhost*. Assim como o servidor smtp.jaku.com, as mensagens encaminhadas para esse servidor terão três respostas:

1. O destinatário pertence ao mesmo domínio, logo não há a necessidade de reenvio das informações.

2. O destinatário pertence ao domínio jaku.com, nesse cenário, é contatado o servidor correto e as mensagens são reenviadas.

3. No último caso, o domínio de destino não é tratado pelo servidor e a mensagem não terá tratamento.

2.3. Processamento de dados no servidor

O método *process_message* será o responsável por adquirir as informações da mensagem enviadas pelo cliente e gerenciá-las visando o correto destino dos dados. Dentro desse método, existem estruturas condicionais que tratam o encaminhamento das mensagens, se o domínio do destinatário é o mesmo, no do servidor será salvo os anexos (se existirem) e o corpo da mensagem, caso contrário, a mensagem será reenviada para o servidor com o domínio correspondente.

Essas estruturas condicionais resolvem os nomes de domínio recebido pelo parâmetro *rcpttos*, que são os destinatários do e-mail enviado pelo cliente, garantindo o correto encaminhamento dos dados. Esse comportamento simula o **protocolo DNS** (note que somente a ideia (resolução dos nomes de domínio) desse protocolo é implementada).

Além disso, toda mensagem que possui anexo recebida pelo servidor passa pelo processo de decodificação antes de armazenar no diretório do servidor, já que esses anexos, quando enviados pelo cliente, estarão codificados em *base64*.