

Chat P2P

Trabalho prático
Redes de Computadores

Introdução

O motivo do trabalho foi desenvolver um chat híbrido, onde os clientes se comunicam diretamente entre si em uma rede local e o servidor guarda somente as informações dos clientes.

O aplicativo desenvolvido usa o protocolo UDP da camada de transporte, e um protocolo desenvolvido pelos alunos na camada de aplicação.



Configuração

Para que a aplicação funcione é preciso que o servidor(ChatTrucoServer.py) esteja rodando em alguma máquina da rede.

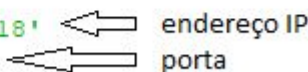
Para configurar o cliente é preciso informar no arquivo Info.py o endereço IP e a porta padrão do servidor.

```
#coding: utf-8

from socket import *

class Info:
    verifica_on = 0
    tempo_servidor = 1

    meuEmail = ''
    serverIp = '10.3.1.18'
    serverPort = 54319
```

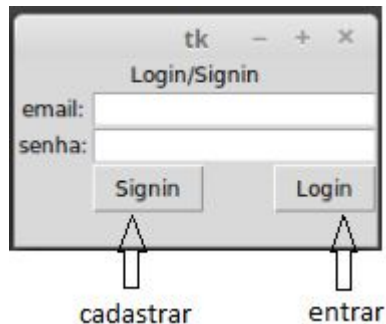


endereço IP
porta

Execução

Para executar a aplicação cliente execute o arquivo login.py

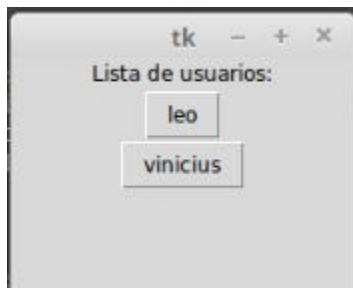
A tela que será exibida tem a opção de (Signin)cadastrar um novo usuário, ou entrar(Login) com um usuário existente.



Execução

Após efetuar o Login(entrar), o usuário irá visualizar todos os clientes que estão online no servidor configurado no arquivo Info.py

Clicando no botão com o nome do usuário será aberta uma nova janela com o chat direto com o usuário selecionado.



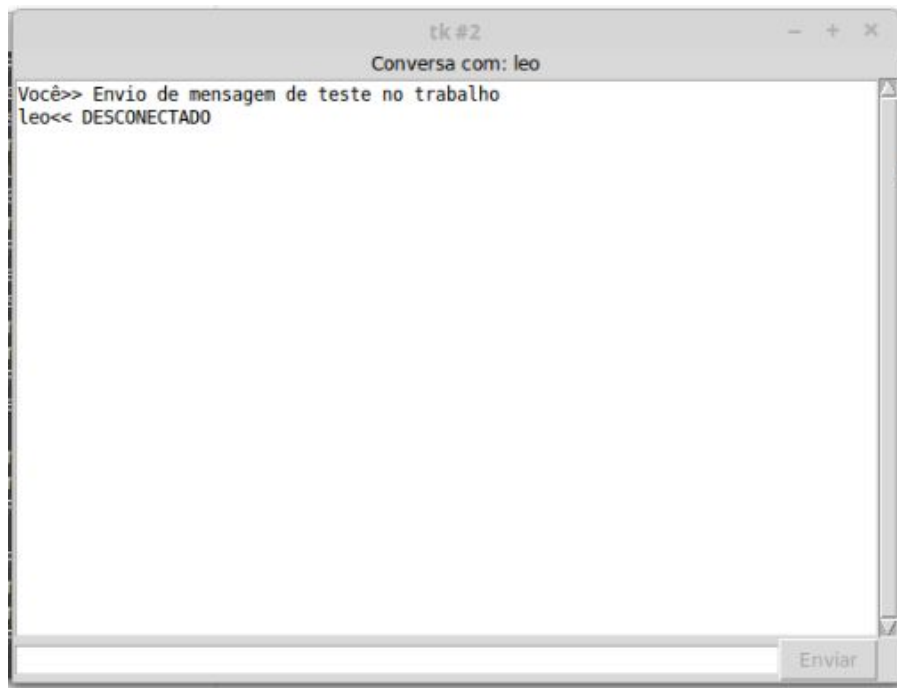
Execução

Na janela do chat é possível enviar uma nova mensagem e visualizar as mensagens recebidas daquele usuário.



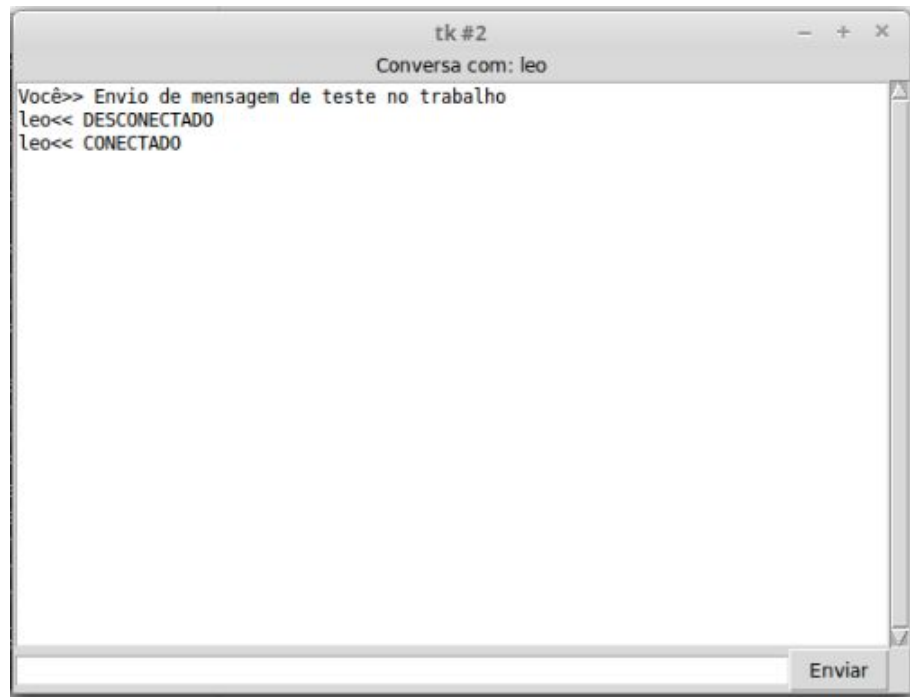
Execução

Se o usuário com quem estiver no chat se desconectar uma mensagem é exibida na área de conversa informando “Desconectado” e o botão de enviar fica desabilitado, impossibilitando que a mensagem seja enviada se o destino não estiver apto a receber, removendo também o usuário da lista de online.



Execução

Caso a janela esteja aberta e o usuário volte a se conectar, uma mensagem “Conectado” aparecerá na área de conversa e o botão enviar será ativado. O usuário também aparecerá novamente na lista de usuários online.



Servidor

O trabalho do servidor é informar a cada cliente a lista de todos os clientes com os IP's atuais online.

Para atualizar os IP's dos clientes assim como verificar se estão online, o servidor periodicamente envia uma mensagem("Truco") a cada cliente. O cliente confirmará seu IP respondendo ao servidor com uma mensagem("Seis"). Caso o cliente não responda a uma determinada quantidade de mensagens enviadas pelo servidor, esse cliente será considerado offline.



Servidor

```
""" Funcionalidade do comando truco """
def func_cmd_truco():
    print("cmd truco")

    global reg_truco
    global qtd_truco

    eliminarDesconectados()

    lista = obterListaOn()

    if(len(lista) <= 0):
        return

    for usuario in lista:
        ip = obterIpOnline(usuario)
        email = obterEmailOnline(usuario)
        responder(Info.comando_truco + " " + onsToString(lista),[ip, Info.serverPort]) #envia um truco para o cliente
        posicao = encontrarTrucado(email) #guarda a posição
        if( posicao == -1): # se não encontrar na lista de trucado adiciona o registro com 1 ocorrencia de trucado
            reg_truco.append(email)
            qtd_truco.append(1)
        else:
            qtd_truco[posicao] += 1 # senão soma a qntdade de vezes que foi trucado e não respondeu

    print("Trucado:",ip,email,qtd_truco[posicao])

""" Funcionalidade do comando seis """
def func_cmd_seis(mensagem,address):
    global qtd_truco
    if( len(mensagem.split()) == 2):
        email = getEmail(mensagem)
        posicao = encontrarTrucado(email)
        if( posicao != -1):
            qtd_truco[posicao] = 0
        print("seis recebido",email)
```

Observações

Se o usuário for considerado offline pelo servidor sua aplicação será encerrada. Para se conectar novamente basta abrir a aplicação e utilizar o email e senha cadastrados anteriormente.



Trabalho desenvolvido pelos alunos:

Leonardo Carvalho

Tarlles Roman

Vinicius Tristão

