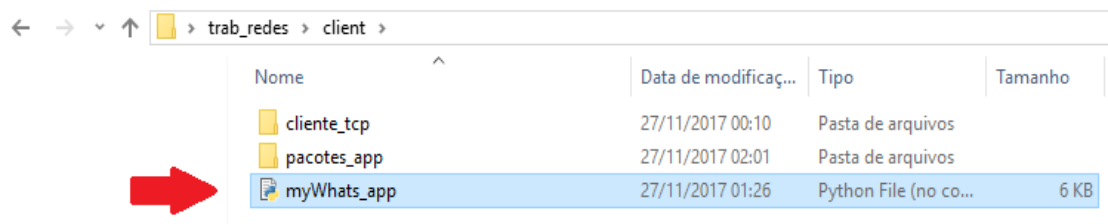


Trabalho de redes – myWhatsApp

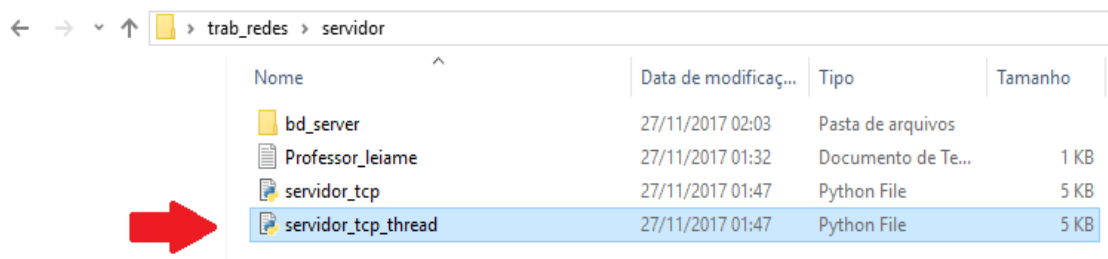
Aluno: Hugo Ramalho – ramalho.hg@gmail.com

1. COMO EXECUTAR:

O trabalho está dividido em dois diretórios principais, um guarda o servidor, e outro o cliente. Primeiro, é necessário iniciar a aplicação servidora, que se encontra no caminho \trab_redes\servidor\servidor_tcp_threads:

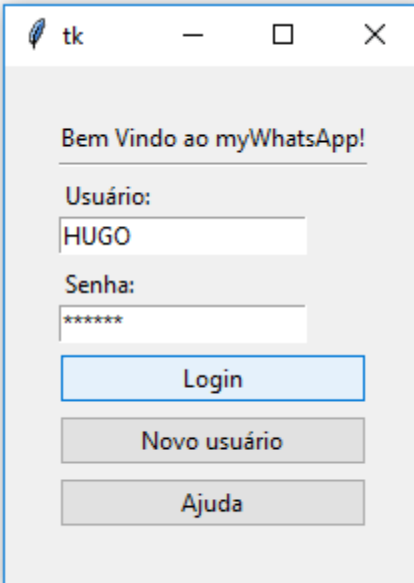


Após iniciar o servidor, deve-se iniciar a aplicação-cliente *myWhats_app.py*, que se encontra no caminho: \trab_redes\client\myWhats_app.py. Caso o programa cliente não encontre uma conexão com o servidor, uma mensagem de erro é invocada.



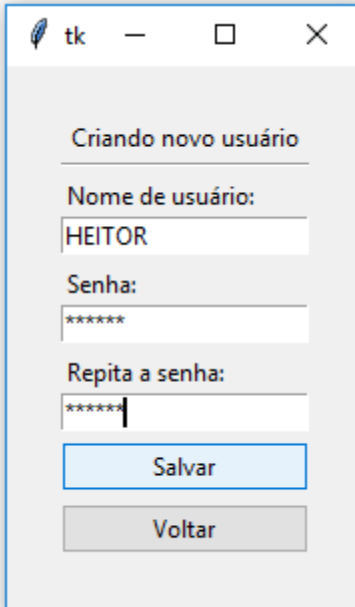
2. MANUAL

Para iniciar uma sessão, é necessário entrar com um usuário e uma senha:



A screenshot of a Tkinter window titled 'tk' with standard window controls. The window displays a login form titled 'Bem Vindo ao myWhatsApp!'. It contains two input fields: 'Usuário:' with the text 'HUGO' and 'Senha:' with masked characters '*****'. Below the fields are three buttons: 'Login' (highlighted in blue), 'Novo usuário', and 'Ajuda'. A red arrow points from the number '123456' to the password field.

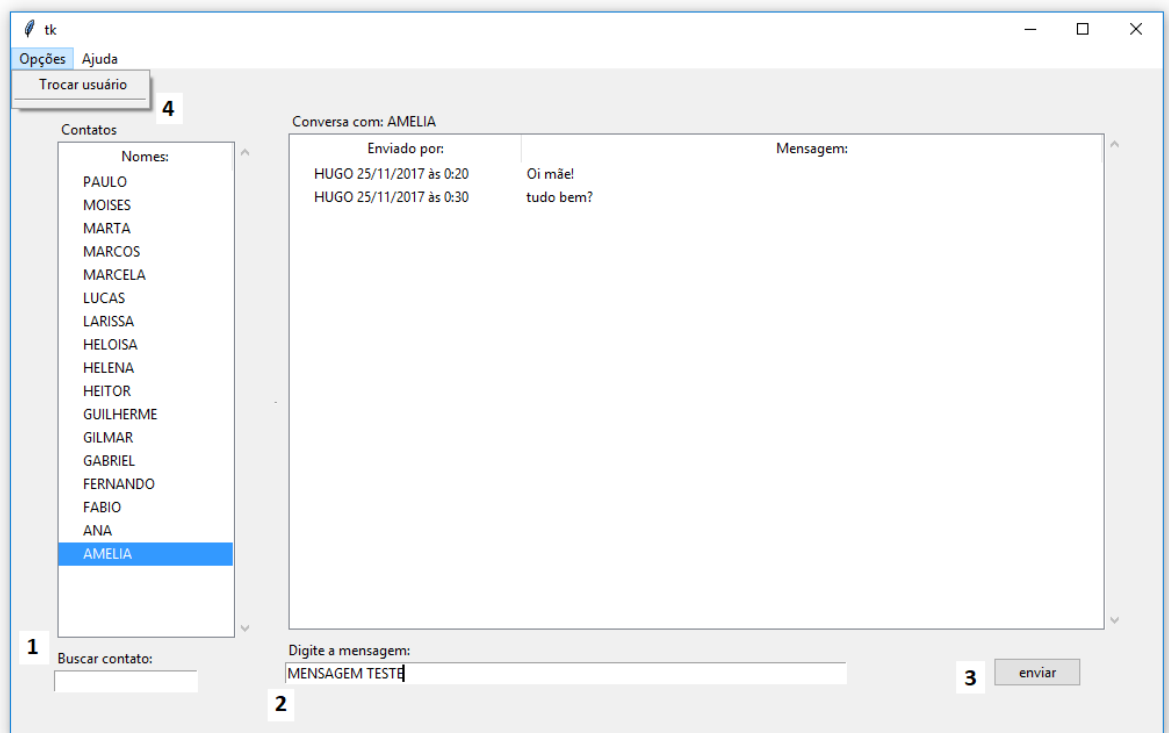
Alguns usuários já estão cadastrados na aplicação, **a senha de todos eles é 123456**. O nome de alguns dos usuários: **HUGO, LUCAS, GILMAR**, etc... É possível também cadastrar um novo usuário:



A screenshot of a Tkinter window titled 'tk' with standard window controls. The window displays a registration form titled 'Criando novo usuário'. It contains three input fields: 'Nome de usuário:' with the text 'HEITOR', 'Senha:' with masked characters '*****', and 'Repita a senha:' with masked characters '*****'. Below the fields are two buttons: 'Salvar' (highlighted in blue) and 'Voltar'.

Basta entrar com algum nome de usuário que ainda não esteja cadastrado, e definir uma senha. As senhas nos dois campos precisam ser iguais. Um aviso é invocado caso as senhas não sejam iguais, ou caso o nome de usuário a ser cadastrado já esteja sendo usado.

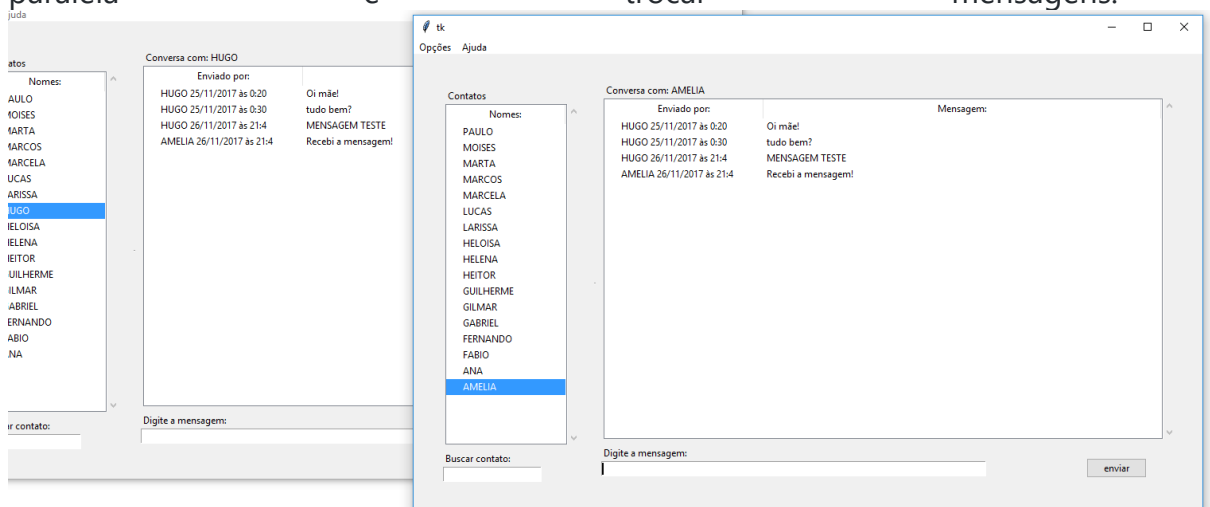
Após iniciada uma sessão, o programa inicia o frame de chat. Para iniciar uma conversa, é necessário escolher um contato, que pode ser feito selecionando seu nome pela lista, ou realizando uma busca pelo nome:



Por tópicos:

- 1) É possível buscar um contato pelo nome. Caso o usuário não exista, um aviso é exibido notificando a inexistência do contato. Pode-se selecionar um contato listado. Todos os contatos são visíveis, ou seja, a relação contato **tem** contato não existe, todos os contatos conhecem uns aos outros.
- 2) A mensagem a ser enviada deve ser digitada na entrada de texto.
- 3) Para enviar a mensagem, basta apertar *Enter* ou clicar no botão *Enviar*.
- 4) Para trocar de usuário, basta selecionar a opção "*Trocar usuário*", em *Opções*.

O servidor suporta múltiplos clientes, ou seja, é possível iniciar uma nova sessão paralela e trocar mensagens:



Porém, para receber a mensagem em tempo real é necessário atualizar a página: clicar em um contato diferente do atual e em seguida voltar para o contato. Isso ocorre porque o cliente não está "ouvindo" o servidor o tempo

todo, apenas ouve após o envio de uma requisição. Para que o cliente ouça permanentemente o servidor, seria necessária uma conexão assíncrona, subdividir o cliente em duas threads, uma de envio outra de recepção. Por limitações de tempo, a aplicação não contará com esse recurso.

3. DESENVOLVIMENTO

A aplicação foi desenvolvida inteiramente em **Python, versão 3.6.1**, fazendo uso dos módulos: socket, socketServer, json, threading, tkinter e sqlite3. Todos eles nativos do Python.

O módulo socket foi usado para implementar o aplicação-cliente. O módulo socketServer, para implementar a aplicação-servidora. O uso do socketServer se deu pela facilidade em implementar um "threaded request handler". A princípio o desenvolvimento da aplicação-servidora fez uso do módulo socket, porém sua implementação se mostrou muito dispendiosa.

O módulo json foi usado para serializar os dicionários-requisição, possibilitando o envio de estruturas de dados mais concisas entre cliente-servidor. Caso contrário toda comunicação se daria por meio de strings.

O módulo threading, usado para implementar o "threaded request handler" do servidor.

O módulo tkinter para desenvolver toda a interface gráfica do programa.

E o módulo sqlite3, foi usado para persistir as mensagens (dentro da aplicação servidora) em um arquivo no formato .db.

A aplicação foi dividida em pacotes, de forma a modularizar o programa, possibilitando o uso de outro script de servidor, desde que o handler() faça as mesmas chamadas de funções, a parte estrutural pode ser completamente diferente.

O mesmo se aplica ao cliente, neste caso, com um grau maior de modularização, já que a classe cliente não precisa fazer chamada de função alguma, apenas manter o padrão de envio de requisições cliente-servidor, que no caso do programa, se dá por meio de dicionários, e a requisição é especificada na chave 'req'. O servidor responde a requisição com outro dicionário, contendo o 'feedback'. 'feedback' = 0 indica um fluxo de eventos normal.