**Aplicação de Sockets em Python 3 para Estabelecimento de Conexão para Envio de Mensagens**

**Clara Bettencourt1, Gabriel Antunes1, Gabriela Amaral1, João Pedro Macabu1, Mariana Moraes1**

1Coordenação de Engenharia de Controle e Automação – Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro – RJ – Brasil

clarabette@poli.ufrj.br, g.queirozantunes@poli.ufrj.br, gabii.a.alcantara@poli.ufrj.br, jpcosta97@poli.ufrj.br, marianamoraesfs@poli.ufrj.br

***Abstract.*** *In this work two protocols were created for message exchange. In a first case, a client-server model was developed, where clients use the server as a bridge for communication and then store the conversation data. Next a peer-to-peer model was created without the need to use an intermediary and the conversation was also archived.*

***Resumo.*** *Neste trabalho foram criados dois protocolos para troca de mensagens. Em um primeiro caso foi elaborado um modelo cliente-servidor, onde os clientes utilizam o servidor como ponte para comunicação e esse último armazena os dados da conversa. Em seguida, um modelo peer-to-peer foi criado sem a necessidade de utilizar um intermediário, tendo a conversa também arquivada.*

1. Modelo Cliente-Servidor

Nesse tópico abordaremos os códigos cliente.py e servidor.py, os métodos usados para a construção de ambos e suas fontes. Acerca da estrutura, esse modelo é composto de dois códigos, o cliente.py (que executa a ação de conectar-se ao servidor.py e enviar mensagens para outros clientes ou receber as que estão lhe sendo enviadas) e o servidor.py (que se encarrega de estabelecer a conexão entre clientes criando um ambiente, um meio para que os mesmos possam se comunicar).

1.1. Cliente

Começaremos pelo código cliente.py, onde estabeleceremos uma conexão com os demais clientes.

1.1.1. Código cliente.py

Começamos nossa aplicação criando as variáveis globais tam\_cabe, host, port, end e nome.

* tam\_cabe é o tamanho definido do cabeçalho que usaremos
* host é o número do ip de onde o servidor está
* port é o número da porta que usaremos
* end consiste em criar o endereço usando o host e a port
* nome é uma variável que recebe o nome do usuário que rodar o código

Seguimos criando o mecanismo de socket para estabelecimento de conexão usando o protocolo IPv4 (AF\_NET) e o TCP (SOCK\_STREAM) na variável conn. Após, conectamos ao endereço do servidor com conn.connect(end).

1.1.2. Fontes

Depois da construção do esqueleto, usamos um código pronto da internet, encontrado em: *https://pythonprogramming.net/client-chatroom-sockets-tutorial-python-3/?completed=/server-chatroom-sockets-tutorial-python-3/*.Tomamos o mesmo como base para polir as mensagens e a apresentação do aplicativo, melhorando também nossa organização e entendimento de como proceder para executar o conjunto cliente-servidor com sucesso.

Nesse site, nos é apresentado e ensinado passo a passo do código chatroom\_cliente.py, em que seguimos as instruções usando das explicações para comentar o cliente.py e, eventualmente, compilar os comentários em um documento como este.

1.2. Servidor

Para o código do cliente funcionar, precisamos de um meio por onde as mensagens serão enviadas e recebidas, logo, surge a necessidade de um servidor.

1.2.1. Código servidor.py

1.2.2. Fontes