Atividade Avaliativa #04: SOCKETS – Client-Server APP With ECHO

Conforme os exemplos disponibilizados no Moodle, podemos implementar um Chat utilizando SOCKETS baseado no protocolo TCP.

Sendo assim, nessa atividade avaliativa, você deverá fazer os ajustes nos códigos base dos exemplos disponibilizados no Moodle para que sejam atendidos os seguintes requisitos:

a) Deverão ser estabelecidos alguns “comandos” que o cliente irá enviar para o servidor;

b) Cada comando deverá ser interpretado pelo servidor e este retornará ao cliente a resposta correspondente;

c) Cada implementação dos comandos no servidor deverão ser realizados através da implementação de funções definidas pelo usuário em um arquivo separado chamado de **funcoes_socket.py**;

d) Esses comandos deverão executar as seguintes tarefas:

i) Deverá haver um comando que solicite ao servidor a sua data e hora;

ii) Deverá haver um comando que repasse ao servidor uma URL e o servidor retorne ao cliente a rota do servidor até a URL informada.

DICA: Se você executar o comando `tracert -d4 www.uol.com` em um terminal do Windows, terá o caminho que sua máquina faz até atingir o portal UOL. Um exemplo de resposta do comando está a seguir:

```
Rastreando a rota para amazonas.uol.com.br [200.147.35.224] com no máximo 30 saltos:
1     9 ms     9 ms     9 ms  192.168.0.1
2    28 ms    21 ms    20 ms  10.17.0.1
3    22 ms    21 ms    24 ms  177.195.26.21
4     *        *       25 ms  201.57.195.101
5     *        *        *     Esgotado o tempo limite do pedido.
6     *        *        *     Esgotado o tempo limite do pedido.
7    83 ms    83 ms     *     200.230.251.2
8    86 ms    79 ms    81 ms  200.244.216.122
9    85 ms    88 ms    88 ms  200.211.219.210
10    80 ms    82 ms    82 ms  186.234.26.65
11    89 ms    80 ms    87 ms  200.147.26.30
12    80 ms    82 ms    79 ms  200.147.35.224
Rastreamento concluído.
```

É possível executar esse comando dentro de um programa em Python e obter um resultado como uma *string*, com toda a resposta.

Para tanto usando os seguintes comandos (ao final toda o resultado da rota está na varável **strCaminho** (as linhas estão separadas por **\r\n**):

```
import subprocess
strCMD = 'tracert -d4 www.uol.com'
strCaminho = subprocess.run (strCMD, capture_output=True).stdout.decode('latin1')
```

iii) Deverá haver um comando que repasse ao servidor uma *string* e o servidor retorne ao cliente essa *string* criptografada utilizando o algoritmo de Vigenère. A chave de criptografia deverá estar no final da mensagem enviada. Cabe ao aluno definir como separar a mensagem da chave para que o servidor identifique quem é quem;

iv) Deverá haver um comando que repasse ao servidor um período (data inicial e data final) e o servidor retorne ao cliente um arquivo com extensão **.json** cujo conteúdo deverá estar em formato de um dicionário. O dicionário de retorno deverá conter as datas e suas respectivas cotações de venda e de compra:

DICA: Para essa requisição deve-se utilizar como referência a URL de requisição ao Banco Central conforme exemplo a seguir (o período está em vermelho):

https://olinda.bcb.gov.br/olinda/servico/PTAX/versao/v1/odata/CotacaoDolarPeriodo(dataInicial=@dataInicial,dataFinalCotacao=@dataFinalCotacao)?@dataInicial=%2701-01-2023%27&@dataFinalCotacao=%2712-31-2023%27&$top=100&$format=json

Atenção para quando der erro, retornar ao cliente uma mensagem informando qual foi o erro dado (vide códigos de retorno de erro do protocolo HTTP).

v) Qualquer outra mensagem enviada ao servidor que ele não reconhecer como mcomando ele deverá informar ao cliente que ele não reconheceu o comando enviado:.