CK0249 - Redes de Computadores
Explicação da tarefa de Programação com Sockets
Lívia Belizário Rocha - 418304

O código é definido em duas funções, sendo elas a `main` e `request`:

```
def main():
    server_port = 3333
    server_socket = socket(AF_INET, SOCK_STREAM) # IPv4, TCP
    server_socket.bind(('', server_port))
    server_socket.listen(1)
    print("Servidor iniciado e pronto para receber uma requisição")

    while 1:
        connection_socket, addr = server_socket.accept()
        print(f"Requisição aceita vindo de {addr}")

        request(connection_socket)
```

- Em `server_port` é definida a porta (3333).
- Em `server_socket`, é criado o servidor socket passando dois parâmetros: `AF_INET` e `SOCK_STREAM`, `AF_INET` sendo utilizado para protocolos IPv4, e `SOCK_STREAM` sendo utilizado para TCP.
- Em `server_socket.bind` é feita a ligação entre o servidor socket e a porta 3333.
- `server_socket.listen` diz quantas conexões poderão ser feitas por vez, nesse caso, apenas uma.
- Entramos então em um loop que só acaba quando o servidor é fechado manualmente, então vemos a chamada do método `accept`, que irá criar uma nova conexão com o cliente.
- Depois vem a chamada da função `request`.

```python
def request(connection_socket):
    try:
        message = connection_socket.recv(1024).decode()

        filename = message.split()[1]
        file = open(f'./files/{filename[1:]}.txt', 'r')
        output = file.read()
        print("O arquivo foi encontrado")

        header = "HTTP/1.1 200 OK\n\n"
        connection_socket.send(header.encode())
        for i in range(0, len(output)):
            connection_socket.send(output[i].encode())

        print("Arquivo enviado com sucesso")

    except IOError:
        print("Arquivo não encontrado")
        header = "HTTP/1.1 404 Not Found"
        connection_socket.send(header.encode())

    print("Servidor pronto para receber outra requisição")
    connection_socket.close()
```

- Recebemos a mensagem do cliente, que precisa então ser quebrada para obtermos o nome do arquivo que ele quer recuperar. Se não fosse feito um `split`, essa seria a mensagem recebida.

```
GET /shrek HTTP/1.1
Host: localhost:3333
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: csrftoken=Gp6jFivwsxD36J4gePPaLbuwq4EBwocZGgcTfyZIcRJdIUK62sjYh4sN75FD86tI; XSRF-TOKEN=gaWzk3q2sqeInDjpxC7UN7BP6r
BKMmq5ynhP0dZTIG0JoGcerD797NgxifCEN5b4
Upgrade-Insecure-Requests: 1
```

- O arquivo então é buscado no sistema de arquivos (pasta `files`) e, se for encontrado, é mandado um cabeçalho dizendo que a requisição foi bem recebida e, logo depois, o próprio arquivo é enviado, linha por linha.

- Porém, caso o usuário passe um arquivo não existente, uma mensagem de erro irá aparecer.