

Projeto de pesquisa - Criando ambientes virtuais de conversação com uso de sockets

Curso: Engenharia de Software – UnB – FGA

Disciplina: Fundamentos de Redes de Computadores

Professor: Fernando William Cruz

Aluno: Carlos Eduardo de Sousa Fiuza – 190056843

Aluno: Paulo Vitor Silva Abi Acl – 190047968

Brasília - DF

2022

1 INTRODUÇÃO

Esse projeto tem como objetivo a criação de uma aplicação que disponibilize salas de bate-papo virtuais através da arquitetura TCP/IP. Para o perfeito funcionamento desse sistema é necessário que o cliente possa criar salas de bate-papo com nome e limite de participantes ou somente ingressar em um chat já existente, além disso deve estar disponível o diálogo entre os clientes das salas, que ele possa sair do chat assim que desejar e também funções extras como listar participantes de uma determinada sala.

2 METODOLOGIA UTILIZADA

Para a realização do trabalho, inicialmente houve uma conversa entre a dupla onde ficou decidido que cada um faria estudos individuais para melhor compreender o escopo do projeto. Posteriormente a dupla se reuniu para realizar o desenvolvimento do chat através da programação por pares, onde levou aproximadamente 7 horas para concluir. Para a confecção do relatório final a dupla se reuniu em chamada de voz através do *discord*, onde foi debatido sobre cada tópico e foi feita a divisão do trabalho.

3 DESCRIÇÃO DA SOLUÇÃO

Para a construção da solução foi usada a linguagem python junto das bibliotecas:

- *types*
- *threading*
- *socket*
- *selectors*
- *sys*

Respeitando os requisitos propostos e a fim de manter a organização do código foram criadas 3 classes, que são:

1. Client: guarda dados referente a se está ativo ou não, e um módulo *selector* que além de controlar os eventos de *I/O* armazena as mensagens em buffer a serem enviadas. Com essa classe é possível conectar, enviar e receber mensagens para uma sala. Para isso acontecer temos:
 - a. Criação do *socket*;
 - b. Conexão do *socket* com a sala;
 - c. Registro do *socket* e eventos de *I/O* no módulo *selector*;
 - d. Criação de uma *thread* com loop para evento de leitura;
 - e. Loop para evento de escrita na *main thread*;
2. Room: guarda lista de clientes conectados, módulo *selector* para controle de evento de *Input*, porta e *host* do *socket* criado e id da sala. Com essa classe é possível criar uma sala de bate-papo entre clientes, gerenciar novos clientes que queiram se conectar (caso ainda possua vagas), desconectar cliente inativo e listar todos os clientes conectados. Para isso acontecer temos:
 - a. Criação do *socket*;
 - b. *Bind* no *host* e porta;
 - c. Execução do método *listen* do *socket*;
 - d. Registro do *socket* e dos eventos de leitura e escrita no módulo *selector*;
 - e. Loop para capturar de novo cliente e para manejar o bate-papo entre os clientes.
3. Server: guarda informação quanto a se o servidor está ativo ou não, além de uma lista de salas. Com essa classe é possível criar salas de bate-papo, listar salas ativas, visualizar logs de execução das salas em operação e visualizar clientes que conectaram e desconectaram. Para isso acontecer temos:
 - a. Loop para captura de *Input* do usuário;
 - b. Caso parâmetros sejam concedidos nova *thread* criada com instância da classe Room.

Para iniciar a solução é necessário primeiro executar o arquivo `server.py` que representa o servidor, após executado será possível visualizar a seguinte tela no terminal:

```
-- Escreva 'listar' para receber todas as salas ativas
-- Digite o IP, porta, num_max pessoas e identificador a qualquer momento para criar uma sala
-> 
```

Figura 1 – Captura de tela da execução do arquivo server.py

Com isso é possível executar os seguintes comandos:

- Criação de sala a partir da inserção do IP, porta, número máximo de pessoas na sala e identificador.

```
-> 127.0.0.1 5000 10 sala_teste
Escutando em ('127.0.0.1', 5000)
-> 
```

Figura 2 – Captura de tela da execução do comando de criar sala

- Escrita da palavra “listar” para listagem de todas as salas ativas no servidor:

```
-> listar
Salas:
sala_teste 127.0.0.1 5000
-> 
```

Figura 3 – Captura de tela da execução do comando "listar"

Após a criação bem-sucedida da sala é possível realizar a conexão do cliente na mesma através da execução do arquivo client.py passando como argumento o IP e porta da sala, e um identificador para o cliente. Com isso será possível observar no terminal a seguinte tela:

```
Conexão bem-sucedida
Digite o que desejar, sendo que "listar" retorna todos os clientes na sala
>> Bem vindo a sala sala_teste

```

Figura 4 – Captura de tela da execução do arquivo client.py

Com isso será possível:


1. Listar outros clientes conectados na mesma sala:

```
listar
>> Clientes conectados:
Voce
127.0.0.1 36534

```

Figura 5 – Captura de tela da execução do comando listar


2. Enviar mensagens para os outros clientes da sala:



```
ola, tudo bem?  
□
```

Figura 6 – Captura de tela ao enviar mensagem

3. Receber mensagens de outro clientes:



```
>> Bem vindo a sala sala_teste  
>> 36532: ola, tudo bem?  
□
```

Figura 7 – Captura de tela ao receber mensagem

4 CONCLUSÃO

4.1 RESULTADOS

Dessa forma, conseguimos chegar a um resultado satisfatório através da solução desenvolvida, onde os usuários conseguem criar salas virtuais de bate-papo, podendo se conectar a esse chat e realizar conversas com outros clientes. No entanto, são encontradas algumas imperfeições, como no caso da criação de threads excessivas, sendo que para cada cliente é necessário uma *thread* para leitura e a outra para escrita, além disso, no identificador do cliente consta o número da *thread* e não o que foi informado como identificação.

4.2 CONSIDERAÇÕES DOS MEMBROS

Com isso é possível listar as considerações de cada membro:

- Carlos Eduardo: O projeto foi muito importante para a fixação do conteúdo teórico, sendo muito satisfatório a confecção do mesmo. A participação do Paulo foi bem importante pois sempre buscou focar nos requisitos, dando a devida importância de cada funcionalidade dentro do código. Acredito que este relatório junto ao código produzido merecem uma nota 9.
- Paulo Vitor: O projeto foi importante para uma melhor compreensão do protocolo de comunicação TCP/IP. Acabei tendo uma participação em todas etapas, sendo que na parte do código tive uma contribuição menor do que o Carlos. Acredito que o projeto teve bons resultados e conseguimos cumprir todos os requisitos básicos, daria uma nota 9.

REFERÊNCIAS

TANENBAUM, A. S.; WETHERALL, D. **Computer Networks**, Fifth Edition. [s.l.] Prentice Hall, 2010

Nathan Jennings. **Socket Programming in Python (Guide)**. Realpython.com. Disponível em: <<https://realpython.com/python-sockets/>>. 2022

Deepak Srivatsav. **SALA DE BATE-PAPO SIMPLES USANDO PYTHON. ACERVO LIMA**. Disponível em: <<https://acervolima.com/sala-de-bate-papo-simples-usando-python/>>. 2022