

Trabalho Prático: Implementação de um Chat P2P

[Link do repositório no github](#)

Índice

- [P2P Chat](#)
- [1. Tracker](#)
- [2. Peer](#)

P2P Chat

1. Tracker

O tracker é um servidor que mantém uma lista de peers conectados e suas respectivas portas. Ele permite que os peers se registrem e obtenham informações sobre outros peers disponíveis na rede.

Para iniciar uma instância do tracker, são necessários os parâmetros de host, porta e o caminho para o banco de dados de usuários. O `users_db` é um arquivo JSON que armazena o nome e a senha criptografada dos usuários registrados.

```
class TrackerServer:
    def __init__(self, host='0.0.0.0', port=6060, users_db='users_db.json'):
        self.host = host
        self.port = port
        self.users_db = users_db
        self.users = self.load_users()
        self.peers = {}
        self.login_history = {}
        self.last_ping = {}
        self.rooms = {}
```

Os atributos principais do tracker incluem:

- `host` : Endereço IP do tracker.
- `port` : Porta na qual o tracker escuta conexões.
- `users_db` : Caminho para o arquivo JSON que armazena os usuários registrados.
- `users` : Dicionário que carrega os usuários do banco de dados.
- `peers` : Dicionário que mapeia os peers conectados e seus respectivos endereços IP e portas.
- `login_history` : Dicionário que registra o histórico de login dos usuários.
- `last_ping` : Dicionário que armazena o último ping recebido de cada peer para saber se estão ativos.
- `rooms` : Dicionário que mapeia salas de chat e seus respectivos participantes.

Em relação aos principais métodos:

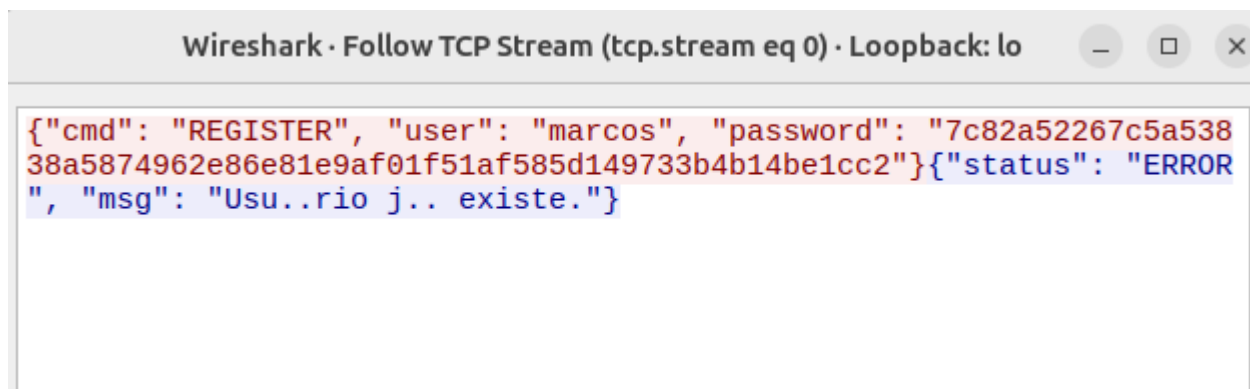
- `load_users()` : Carrega os usuários do banco de dados salvo no arquivo `users_db.json`.

- `save_users()` : Salva os usuários no banco de dados no arquivo `users_db.json`.
- `handle_login()` : Autentica um usuário recebendo nome e senha, e registra seu peer. Depois da autenticação, o peer é adicionado à lista de peers conectados.
- `handle_register()` : Registra um novo usuário com nome e senha, verifica se o nome de usuário já existe.
- `handle_list_peers()` : Retorna a lista de peers conectados, mostrando o nome, o seu último login e seus status online ou offline de acordo com o seu último ping (se foi recebido nos últimos 30 segundos).
- `handle_ping()` : Atualiza o timestamp do último ping de um peer.
- `handle_create_room()` : Cria uma nova sala de chat, verifica se o nome da sala já existe e adiciona o peer como participante.
- `handle_list_rooms()` : Retorna a lista de salas de chat disponíveis, mostrando o nome da sala e os participantes.
- `handle_join_room()` : Adiciona um peer a uma sala de chat existente.
- `handle_client()` : Método principal que lida com as requisições dos peers: LOGIN, REGISTER, LIST_PEERS, PING, CREATE_ROOM, LIST_ROOMS, JOIN_ROOM. Ele recebe as requisições dos peers e chama o método apropriado para tratá-las.
- `start()` : Inicia o servidor tracker, escutando conexões na porta especificada e aguardando requisições dos peers.

Captura de tela

Registro de usuário

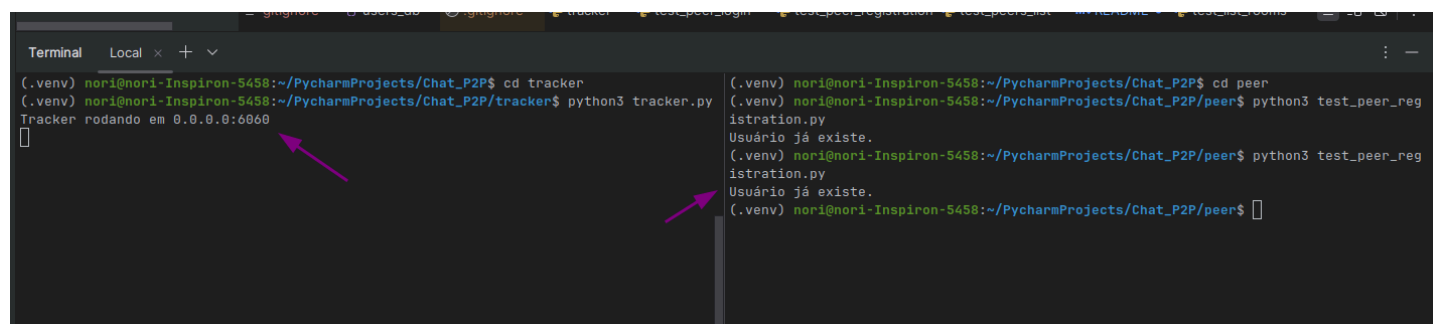
Caso em que usuário já existe:



Comunicação entre o tracker e o peer:

*Loopback: lo						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
tcp.stream eq 0						
No.	Time	Source	Destination	Protocol	Length	Info
5	1.973692493	127.0.0.1	127.0.0.1	TCP	74	54122 → 6060 [SYN] Seq=0 Win=65495 I
6	1.973703517	127.0.0.1	127.0.0.1	TCP	74	6060 → 54122 [SYN, ACK] Seq=0 Ack=1
7	1.973714785	127.0.0.1	127.0.0.1	TCP	66	54122 → 6060 [ACK] Seq=1 Ack=1 Win=6
8	1.973778931	127.0.0.1	127.0.0.1	TCP	183	54122 → 6060 [PSH, ACK] Seq=1 Ack=1
9	1.973783900	127.0.0.1	127.0.0.1	TCP	66	6060 → 54122 [ACK] Seq=1 Ack=118 Wir
10	1.974070448	127.0.0.1	127.0.0.1	X11	116	Event: <Unknown eventcode 123>
11	1.974077275	127.0.0.1	127.0.0.1	TCP	66	54122 → 6060 [ACK] Seq=118 Ack=51 Wj
12	1.974102872	127.0.0.1	127.0.0.1	TCP	66	6060 → 54122 [FIN, ACK] Seq=51 Ack=1
13	1.974182519	127.0.0.1	127.0.0.1	TCP	66	54122 → 6060 [FIN, ACK] Seq=118 Ack=
14	1.974201393	127.0.0.1	127.0.0.1	TCP	66	6060 → 54122 [ACK] Seq=52 Ack=119 Wj

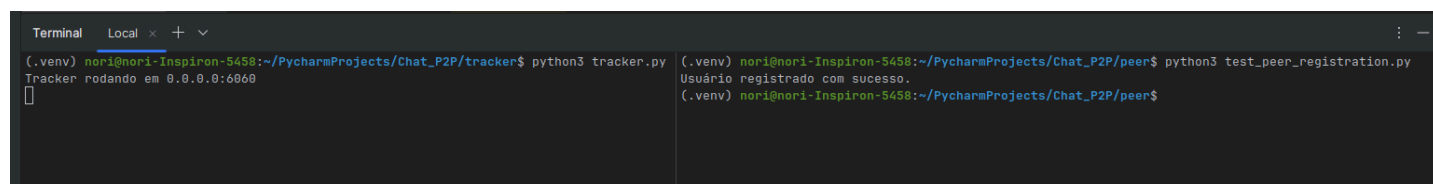
Logs do tracker e do peer:



```
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P$ cd tracker
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/tracker$ python3 tracker.py
Tracker rodando em 0.0.0.0:6060
[]

(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P$ cd peer
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$ python3 test_peer_registration.py
Usuário já existe.
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$ python3 test_peer_registration.py
Usuário já existe.
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$ []
```

Caso de sucesso:

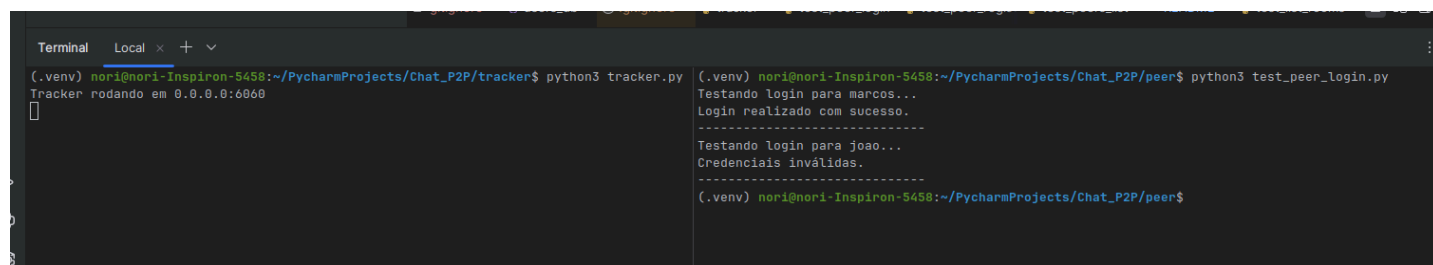


```
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/tracker$ python3 tracker.py
Tracker rodando em 0.0.0.0:6060
[]

(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$ python3 test_peer_registration.py
Usuário registrado com sucesso.
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$
```

Autenticação de usuário

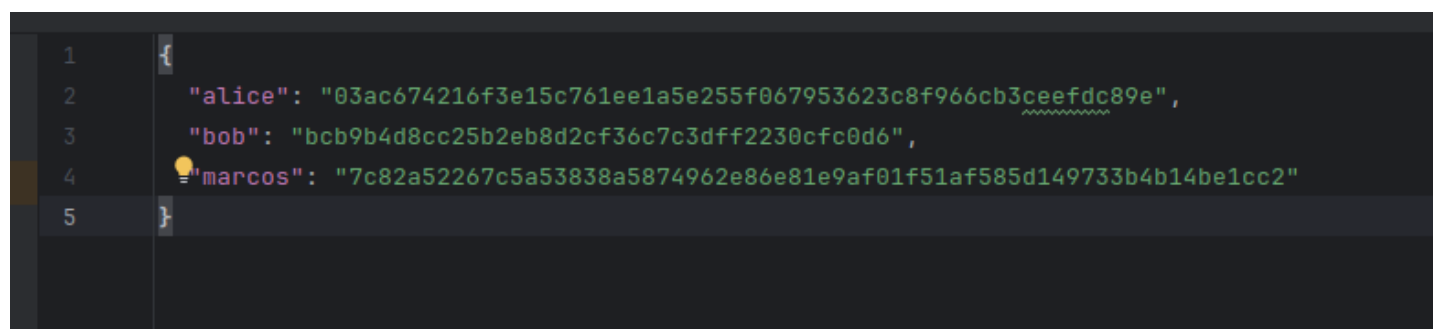
Teste de autenticação com válido e inválido:



```
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/tracker$ python3 tracker.py
Tracker rodando em 0.0.0.0:6060
[]

(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$ python3 test_peer_login.py
Testando login para marcos...
Login realizado com sucesso.
-----
Testando login para joao...
Credenciais inválidas.
-----
(.venv) nori@nori-Inspiron-5458:~/PycharmProjects/Chat_P2P/peer$
```

Registro de informação no arquivo JSON



```
1 {
2   "alice": "03ac674216f3e15c761ee1a5e255f067953623c8f966cb3ceefdc89e",
3   "bob": "bcb9b4d8cc25b2eb8d2cf36c7c3dfff2230cfc0d6",
4   "marcos": "7c82a52267c5a53838a5874962e86e81e9af01f51af585d149733b4b14be1cc2"
5 }
```

2. Peer