```
import socket
import threading

# Lista de clientes conectados
lista_cliente = []

# Função para realizar o broadcast da mensagem para todos os clientes
def broadcast(mensagem, origin_client):
    """
    Envia uma mensagem para todos os clientes, exceto o cliente de origem.

    Parameters:
    - mensagem: Mensagem a ser enviada
    - origin_client: Cliente de origem que não deve receber a mensagem
    """
    mensagem_codificada = mensagem.encode()
    for cliente in lista_cliente:
        if cliente != origin_client:
            try:
                cliente.sendall(mensagem_codificada)
            except socket.error as e:
                # Lida com possíveis erros ao enviar a mensagem
                print(f"Erro ao enviar mensagem para um cliente: {e}")

# Função para remover um cliente da lista
def removerCliente(cliente):
    """
    Remove um cliente da lista.

    Parameters:
    - cliente: Cliente a ser removido
    """
    if cliente in lista_cliente:
        lista_cliente.remove(cliente)
        cliente.close()

# Função para lidar com a entrada do nome de usuário (RF-03)
def obterNomeUsuario(conn):
    try:
        conn.sendall("Digite seu nome de usuário: ".encode())
        nome = conn.recv(1024).decode()
        return nome
    except socket.error as e:
        print(f'Ocorreu um erro durante a entrada do nome de usuário: {e}')
        return None

# Função principal para receber dados de um cliente
def recebeDados(conn, ender):
    try:
        nome = obterNomeUsuario(conn)  # RF-03
        if nome is None:
            return  # Se ocorrer um erro ao obter o nome, encerrar a função

        print(f"Conectado com {nome}, IP: {ender[0]}, PORTA: {ender[1]}")

        # Informar outros usuários sobre a nova conexão (RF-06)
        mensagem_conexao = f"{nome} entrou no chat."
        broadcast(mensagem_conexao, conn)
```

```
        while True:
            mensagem = conn.recv(1024).decode()

            if mensagem.lower() == 'sair':
                # Informar outros usuários sobre a desconexão (RF-04)
                mensagem_desconexao = f"{nome} saiu do chat."
                broadcast(mensagem_desconexao, conn)

                # Remover cliente da lista (RF-02)
                removerCliente(conn)
                break

            mensagem_nome = f"{nome} >> {mensagem}"  # RF-07
            print(mensagem_nome)

            # Realizar broadcast da mensagem para todos os clientes (RF-02)
            broadcast(mensagem_nome, conn)

    except socket.error as e:
        print(f'Ocorreu um erro durante o recebimento de dados: {e}')
        removerCliente(conn)

# Configuração do servidor
HOST = '26.253.16.0'
PORT = 9999  # Use um número de porta válido, por exemplo, 9999

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((HOST, PORT))
sock.listen()
print(f"O Servidor {HOST}:{PORT} está aguardando conexões")

while True:
    try:
        conn, ender = sock.accept()
        lista_cliente.append(conn)

        threadCliente = threading.Thread(target=recebeDados, args=[conn, ender])
        threadCliente.start()

    except socket.error as e:
        print(f'Ocorreu um erro durante o ACCEPT() na conexão com um novo usuário:
{e}')
        continue
```