

Pruebas

1. Uso esperado con RPC incluido, para el resto de pruebas no se mostrará la salida del servidor RPC, porque la implementación usada tarda mucho tiempo en ejecutarse, y se recomienda al profesor que para hacer las pruebas también comente las 8 líneas en el archivo server.c que son:

```
imprimir_operacion_usuario_1(&send_rpc, client);
```

En las líneas: 429, 463, 527, 574, 656, 721, 765, 817

Esto se debe a que puede tardar varios segundos la parte del servicio RPC.

Para esta prueba se realizaron las operaciones exitosas de

REGISTER user

CONNECT user

PUBLISHED lib1 descripcion de fichero

LIST_USERS

LIST_CONTENT user

DELETE lib1

LIST_CONTENT user

DISCONNECT user

```
s_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> publish lib1 descripcion de fichero
c> PUBLISH OK
c> list users
c> LIST USERS OK
c> user 127.0.0.1 1024
c> list content user
c> LIST CONTENT OK
c> lib1 "descripcion de fichero"
c> delete lib1
c> DELETE OK
c> list content user
c> LIST CONTENT OK
c> disconnect user
c> DISCONNECT OK
c> quit
c> QUIT OK
+++ FINISHED +++
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistema
s_distribuidos/Practica_final$
```

```
as_distribuidos/Practica_final$ ./server 3
s> REGISTER
s> CONNECT
s> PUBLISH
s> LIST USERS
s> LIST CONTENT
s> DELETE
s> LIST CONTENT
s> DISCONNECT
[]
```

terminal de servidor RPC:

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proy
user      REGISTER      12/05/2024 18:16:13
user      CONNECT      12/05/2024 18:16:49
user      PUBLISH      lib1      12/05/2024 18:17:32
user      LIST USERS      12/05/2024 18:18:32
user      LIST CONTENT      12/05/2024 18:19:52
user      DELETE lib1      12/05/2024 18:20:39
user      LIST CONTENT      12/05/2024 18:21:24
user      DISCONNECT      12/05/2024 18:21:55
[]
```

A Partir de esta prueba se muestran solo la terminal del server y del client.py por lo comentando anteriormente

2. Vamos a probar la operación registrar.
Las operaciones que se van a realizar son:
REGISTER user
REGISTER user

<pre> o yecto/sistemas_distribuidos/Practica_final\$ python3 client.py -s localhost -p 3000 c> register user c> REGISTER OK c> register user c> USERNAME IN USE c> █ </pre>	<pre> ~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final\$./server 3000 s> REGISTER s> REGISTER █ </pre>
--	--

3. Probar operación connect, aquí es interesante destacar la aproximación que hemos hecho donde solo puede haber un usuario conectado por terminal.

Las operaciones que se van a realizar:

Tras haber hecho REGISTER user, y siguiendo el orden de ejecución.

Terminal 1:

```

CONNECT user
CONNECT user2
CONNECT user

```

Terminal 2:

```

CONNECT user2
CONNECT user

```

<pre> o yecto/sistemas_distribuidos/Practica_final\$ python3 client.py -s localhost -p 3000 c> register user c> REGISTER OK c> connect user c> CONNECT OK c> connect user2 c> CONNECT FAIL, ONE CONNECTED USER PER TERMINAL c> connect user c> CONNECT FAIL, USER ALREADY CONNECTED c> █ </pre>	<pre> ~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final\$./server 3000 s> REGISTER s> CONNECT s> CONNECT s> CONNECT █ </pre>	<pre> o yecto/sistemas_distribuidos/Practica_final\$ python3 client.py -s localhost -p 3000 c> connect user2 c> CONNECT FAIL, USER DOES NOT EXIST c> connect user c> CONNECT FAIL, USER ALREADY CONNECTED c> █ </pre>
--	---	--

Destacar que las 2 operaciones finales de la terminal 1 no se enviarán al servidor, es client.py el que directamente está gestionando esos casos.

4. Probar operación PUBLISH.

Tras haber hecho REGISTER user y CONNECT user

Las operaciones que se van a realizar:

```

PUBLISH lib1 text desc
PUBLISH lib1 repito
LIST_CONTENT user
DISCONNECT user
PUBLISH user
UNREGISTER user
PUBLISH user

```

<pre> o yecto/sistemas_distribuidos/Practica_final\$ python3 client.py -s localhost -p 3000 c> register user c> REGISTER OK c> connect user c> CONNECT OK c> publish lib1 texto desc c> PUBLISH OK c> publish lib1 repito c> PUBLISH FAIL, CONTENT ALREADY PUBLISHED c> list content user c> LIST CONTENT OK c> lib1 "texto desc" c> disconnect user c> DISCONNECT OK c> publish lib1 desc c> PUBLISH FAIL, USER NOT CONNECTED c> unregister user c> UNREGISTER OK c> publish lib1 desc c> PUBLISH FAIL, USER DOES NOT EXIST c> █ </pre>	<pre> ~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final\$./server 3000 s> REGISTER s> CONNECT s> PUBLISH s> PUBLISH s> LIST CONTENT s> DISCONNECT s> UNREGISTER █ </pre>
--	--

5. Probar operación DELETE.

Tras haber hecho REGISTER user, CONNECT user, PUBLISH lib1 text desc y LIST_CONTENT user.

Las operaciones que se van a realizar:

DELETE lib1

LIST_CONTENT user

DELETE invalid

DISCONNECT user

DELETE lib1

UNREGISTER user

DELETE lib1

```
o yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> publish lib1 text desc
c> PUBLISH OK
c> list content user
c> LIST CONTENT OK
c> delete lib1
c> DELETE OK
c> list content user
c> LIST CONTENT OK
c> delete invalid
c> DELETE FAIL, CONTENT NOT PUBLISHED
c> disconnect user
c> DISCONNECT OK
c> delete lib1
c> DELETE FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> delete lib1
c> DELETE FAIL, USER DOES NOT EXIST
c> █

~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> PUBLISH
s> LIST CONTENT
s> DELETE
s> LIST CONTENT
s> DELETE
s> DISCONNECT
s> UNREGISTER
█
```

6. Probar operación DISCONNECT.

Tras haber hecho REGISTER user, CONNECT user.

Las operaciones que se van a realizar:

Terminal 2:

DISCONNECT user

Terminal 1:

DISCONNECT user

DISCONNECT user

DISCONNECT user2

```
yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> disconnect user
c> DISCONNECT OK
c> disconnect user
c> DISCONNECT FAIL / USER NOT CONNECTED
c> disconnect user2
c> DISCONNECT FAIL / USER DOES NOT EXIST
c> █

~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> DISCONNECT
s> DISCONNECT
s> DISCONNECT
s> DISCONNECT
█

cto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> disconnect user
c> DISCONNECT FAIL / CAN NOT DISCONNECT OTHER USERS
c> █
```

7. Probar operación LIST_USERS.

Tras haber hecho REGISTER user, CONNECT user.

Las operaciones que se van a realizar:

LIST_USERS

DISCONNECT user

LIST_USERS
UNREGISTER user
LIST_USERS

```
o yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> list users
c> LIST USERS OK
    user 127.0.0.1 1024
c> disconnect user
c> DISCONNECT OK
c> list users
c> LIST USERS FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> list users
c> LIST USERS FAIL, USER DOES NOT EXIST

~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> LIST USERS
s> DISCONNECT
s> UNREGISTER
[]
```

8. Probar operación LIST_CONTENT.

Tras haber hecho REGISTER user, CONNECT user, PUBLISH lib1 text desc.

Las operaciones que se van a realizar:

LIST_CONTENT user
LIST_CONTENT user2
DISCONNECT user
LIST_CONTENT user
UNREGISTER user
LIST_CONTENT user

```
o yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> publish lib1 text desc
c> PUBLISH OK
c> list content user
c> LIST CONTENT OK
    lib1 "text desc"
c> list content user2
c> LIST CONTENT FAIL, REMOTE USER DOES NOT EXIST
c> disconnect user
c> DISCONNECT OK
c> list content user
c> LIST CONTENT FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> list content user
c> LIST CONTENT FAIL, USER DOES NOT EXIST
c>

~/Documents/sistemas_distribuidos/proyecto/sistemas_distribuidos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> PUBLISH
s> LIST CONTENT
s> LIST CONTENT
s> DISCONNECT
s> UNREGISTER
[]
```

9. Probar operación UNREGISTER.

Tras haber hecho REGISTER user, CONNECT user.

Las operaciones que se van a realizar:

UNREGISTER user
DISCONNECT user
UNREGISTER user
UNREGISTER user

```
o yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> unregister user
c> CAN NOT UNREGISTER CONNECTED USERS
c> disconnect user
c> DISCONNECT OK
c> unregister user
c> UNREGISTER OK
c> unregister user
c> USER DOES NOT EXIST
c> █

~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> UNREGISTER
s> DISCONNECT
s> UNREGISTER
s> UNREGISTER
█
```

10. Probar operación GET_FILE.

Tras haber hecho en Terminal 1 REGISTER user, CONNECT user. y en Terminal 2

REGISTER user2 CONNECT user2

Las operaciones que se van a realizar:

Terminal 1:

GET_FILE user2 prueba.txt primero.txt

GET_FILE user2 no-existe.txt tercero.txt

Terminal 2:

GET_FILE user prueba.txt segundo.txt

DISCONNECT user2

Terminal 1:

GET_FILE user2 prueba.txt cuarto.txt

DISCONNECT user

GET_FILE user2 prueba.txt quinto.txt

UNREGISTER user2

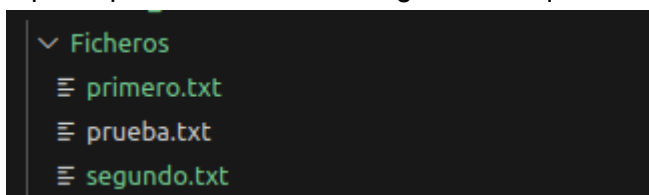
GET_FILE user2 prueba.txt sexto.txt

```
o yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> GET FILE user2 prueba.txt primero.txt
c> GET FILE OK
c> get file user2 no-existe.txt tercero.txt
GET FILE FAIL / FILE NOT EXIST
c> get file user2 prueba.txt cuarto.txt
c> GET FILE FAIL
c> disconnect user
c> DISCONNECT OK
c> get file user prueba.txt quinto.txt
c> GET FILE FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> get file user2 prueba.txt sexto.txt
c> GET FILE FAIL, USER DOES NOT EXIST
c> █

~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> REGISTER
s> CONNECT
s> GET FILE
s> GET FILE
s> GET FILE
s> DISCONNECT
s> GET FILE
s> DISCONNECT
s> UNREGISTER
█

o cto/sistemas_distribuidos/Practica_final$ python3 client.py -s localhost -p 3000
c> register user2
c> REGISTER OK
c> connect user2
c> CONNECT OK
c> GET FILE user prueba.txt segundo.txt
c> GET FILE OK
c> disconnect user2
c> DISCONNECT OK
c> █
```

Aquí se puede ver los archivo generado, que son nuevo:



En este caso tanto primero.txt como segundo.txt estan porque ambas terminales están ejecutando el mismo archivo