

Práctica 2

Aplicación distribuida: Peer to Peer

Computación Distribuida 2021-1

Profesor: Salvador López Mendoza

Ayudante: Alexis Castro Hernández

Laboratorista: Jorge Erick Rivera López

Licenciatura en Ciencias de la Computación Facultad de Ciencias, UNAM

Alumno:

Jorge Iván Pérez Pérez 314211349

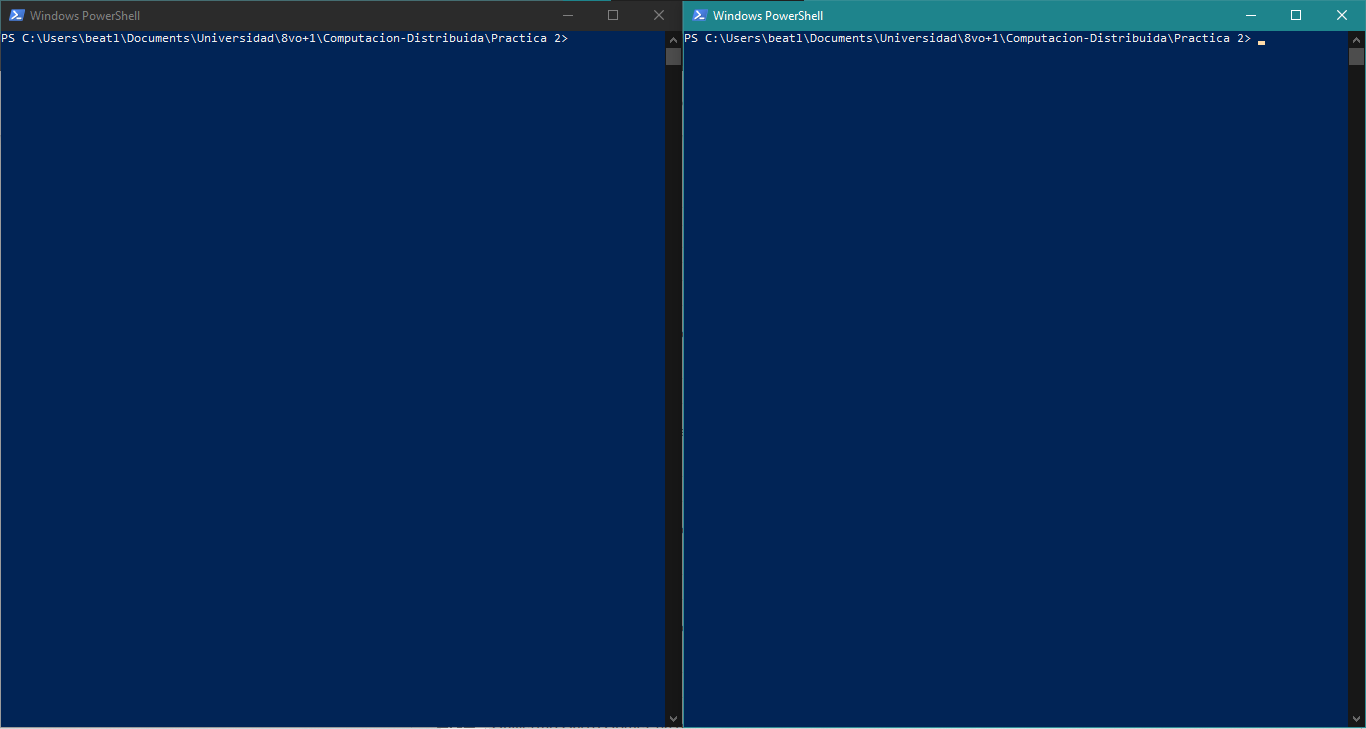
**Introducción**

En esta práctica veremos cómo funciona el modelo peer to peer creando un chat con Python, la idea del modelo peer to peer consiste en hacer una conexión entre computadoras sin intermediarios, es decir si recordamos el modelo cliente/servidor lo que hacíamos era que una computadora que tiene el rol de cliente manda una solicitud hacía otra computadora que funge como servidor y este al recibir la petición regresa una respuesta hacía el cliente, en el caso de peer to peer actúan simultáneamente como clientes y servidores lo que permite el intercambio directo de información, en cualquier formato, entre los ordenadores interconectados.

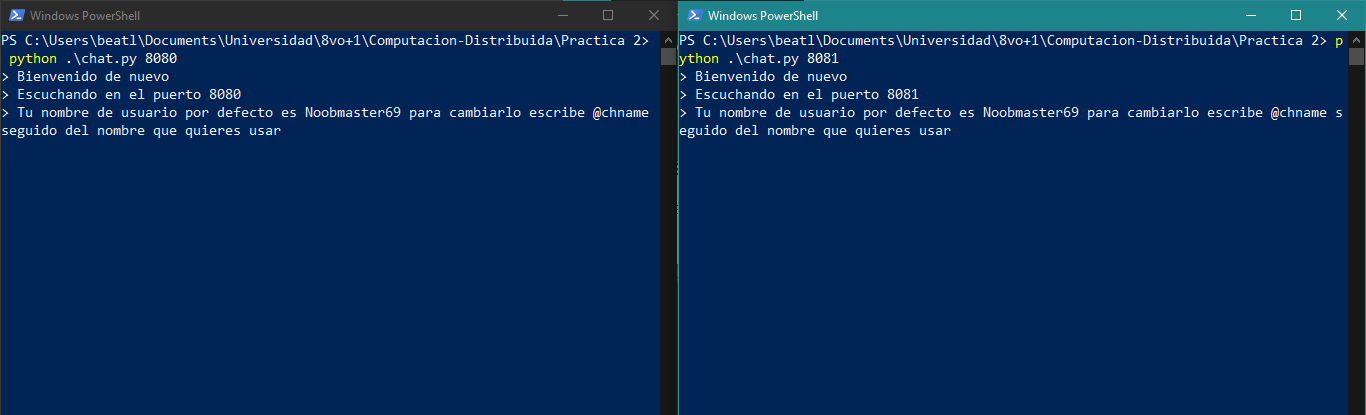
Para el chat lo que se hará es tener la parte del cliente para mandar los mensajes y conectarse a un chat nuevo por medio de puertos y la parte servidor recibirá esta información en el caso de ya haber entablado comunicación entre las computadoras la información recibida serán los mensajes que intercambien los usuarios y en caso de que no exista tal conexión la información recibida será la petición para crear la conexión entre usuarios.

**Funcionamiento**

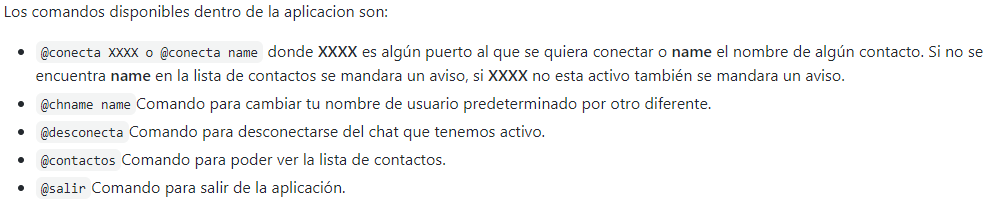
Para poder ver el funcionamiento abriremos 2 terminales para poder entablar la conexión.



Para la terminal izquierda nos conectaremos en el puerto 8080, mientras que en el de la derecha será en el puerto 8081, corremos el programa con el comando python ./chat.py 8080 si se corre desde Linux el comando es python3 chat.py 8080.



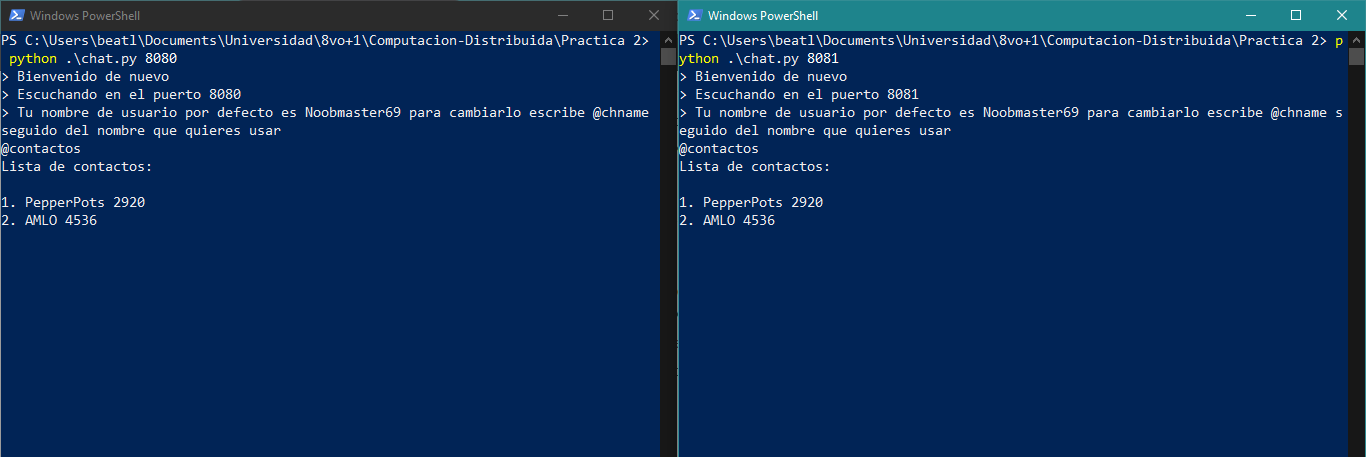
Al inicio del programa los 2 usuarios tendrán el mismo nombre de usuario que esta por defecto. Una vez dentro ya podemos utilizar los siguientes comandos





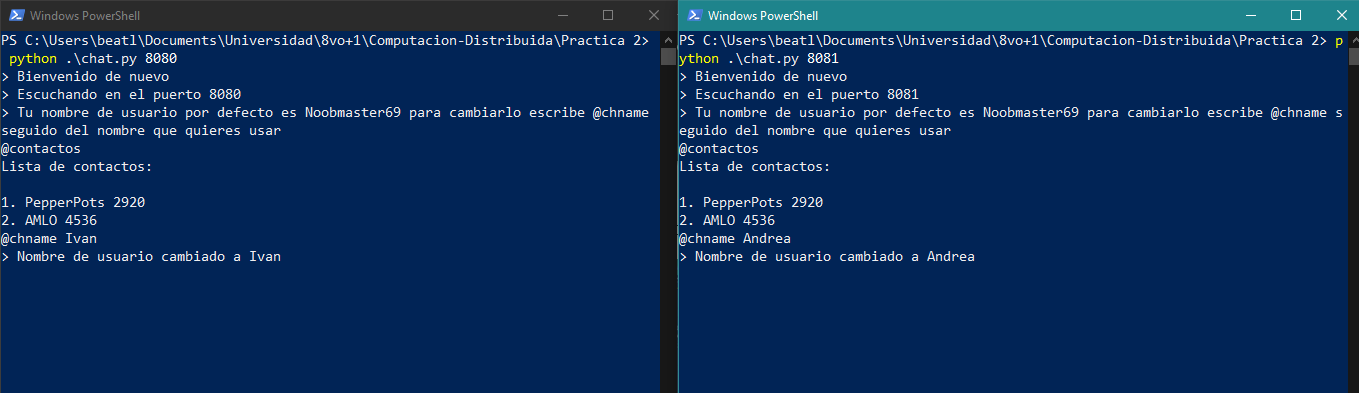
Para poder observar como es que se guardan los contactos una vez que se comunican por primera vez, primero veremos la lista de contactos por defecto, para posteriormente hacer una conexión entre los 2 usuarios que estamos utilizando y volver a ver los contactos ya actualizada, también nos va a servir para ver como podemos conectarnos con un contacto con su nombre de usuario y ya no por el puerto.

Primero veremos los contactos con @contactos:



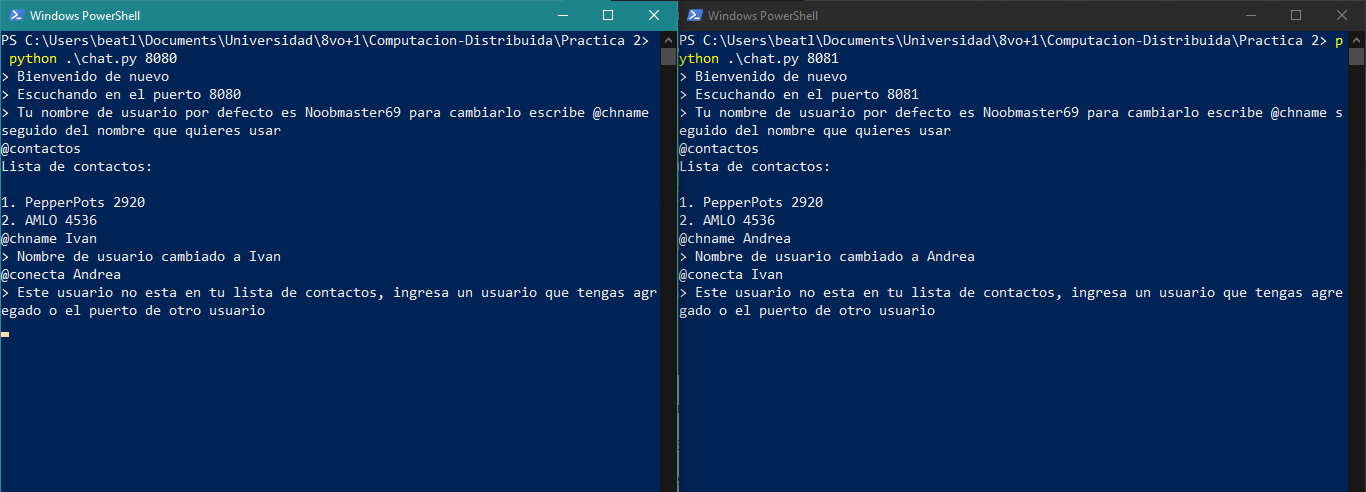
Como podemos ver los 2 usuarios tienen los mismos contactos ya que son los que están predeterminados.

Ahora probaremos el comando @chname:

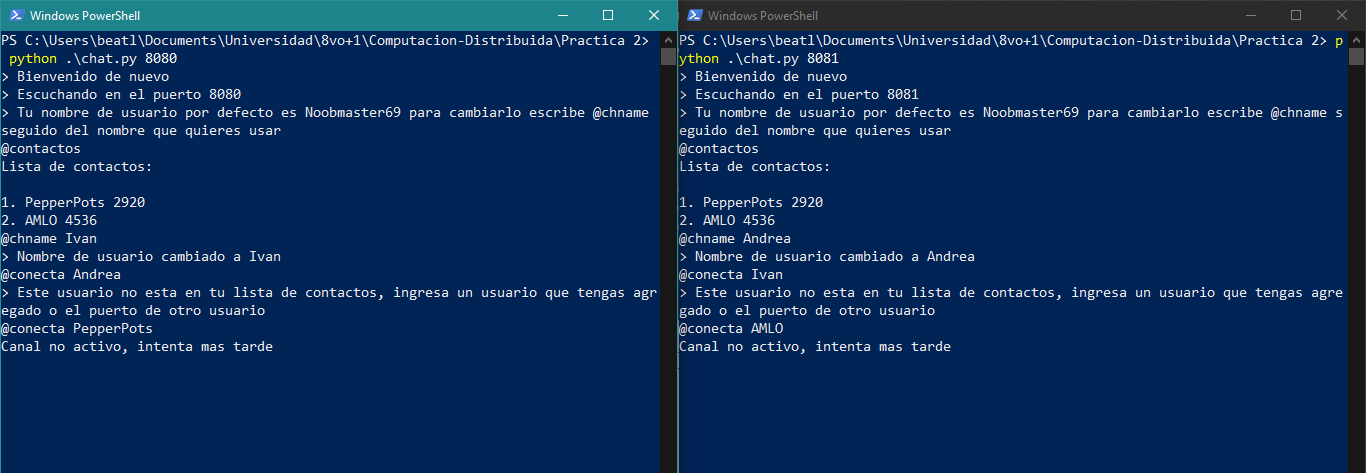


Para el usuario que está conectado por el puerto 8080 (terminal izquierda) le cambiamos el nombre por Ivan y al usuario que está conectado por el puerto 8081 (terminal derecha) le cambiamos el nombre por Andrea.

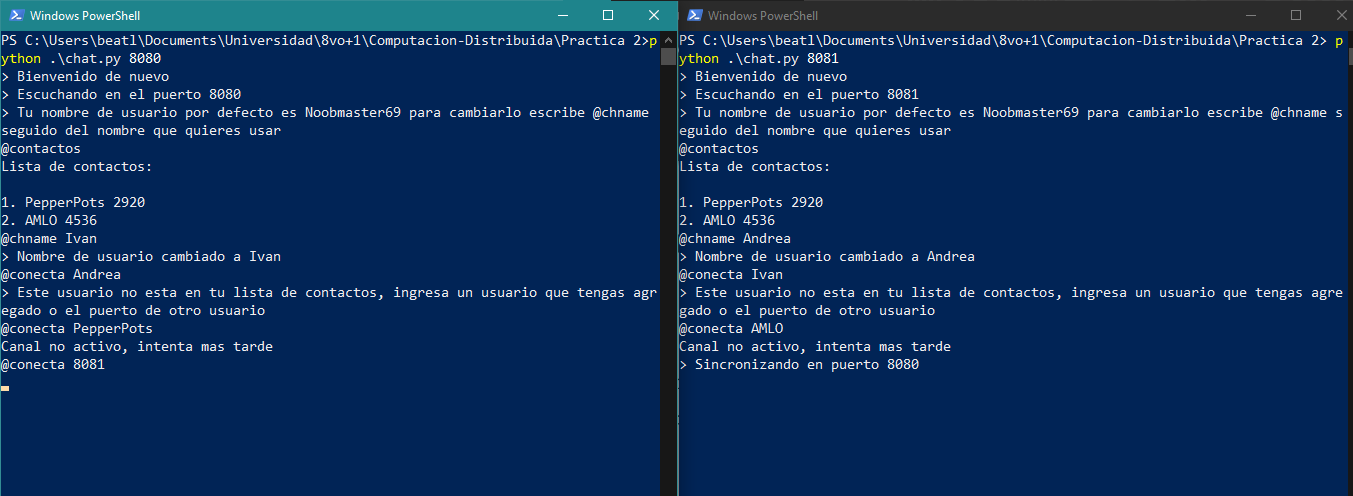
Ahora si nosotros intentáramos conectarnos mediante el nombre de alguno de los 2 usuarios no será posible ya que nunca se ha establecido una conexión entre ellos por lo tanto tampoco se encuentran en su lista de contactos.



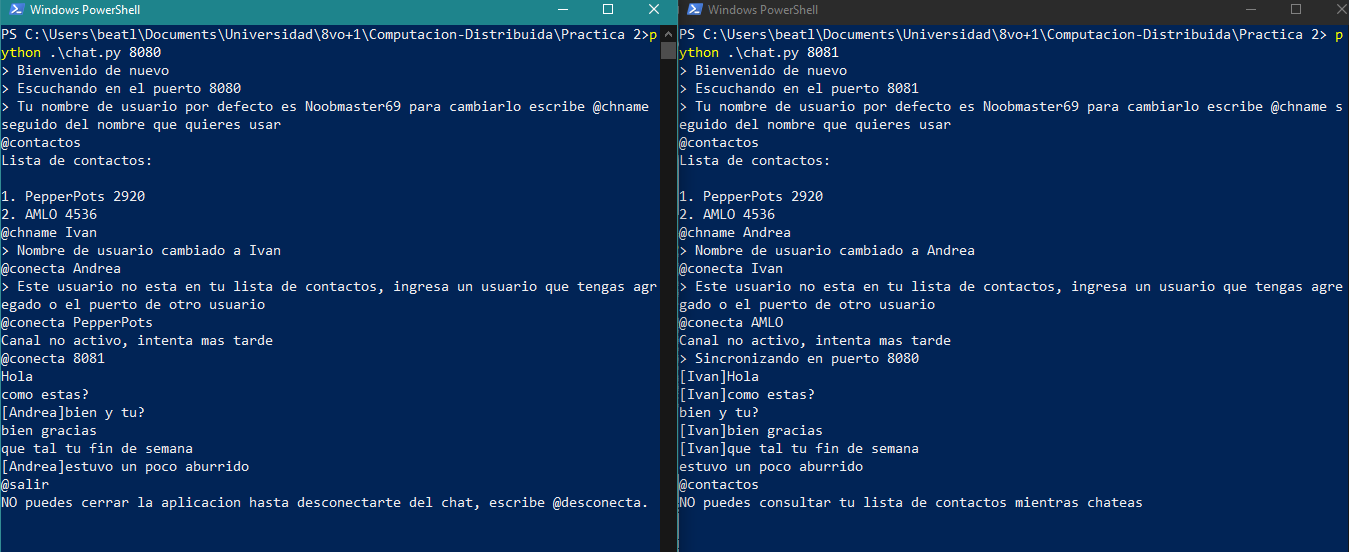
Por otro lado, si quisiéramos conectarnos con un usuario que, si está en la lista de contactos, pero el puerto no está activo también recibiremos un mensaje.



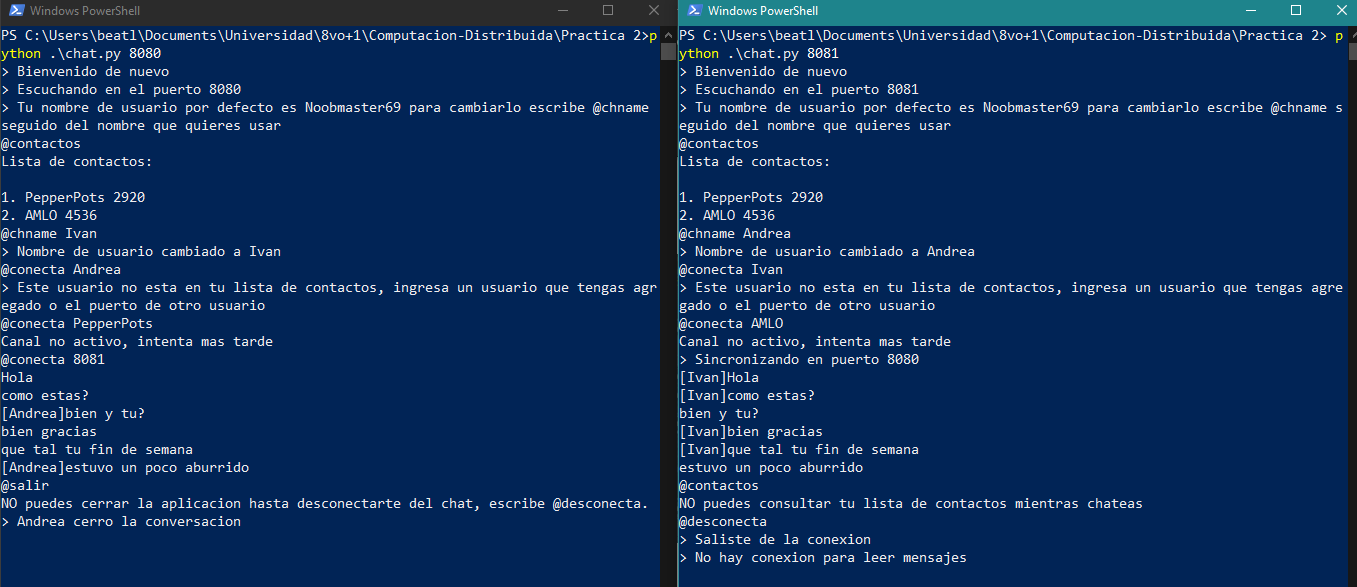
Ahora si probemos entablar una conexión con los 2 usuarios que tenemos. En este caso el usuario Ivan se tratará de conectar con el usuario Andrea mediante el puerto 8081 entonces se escribe el comando @conecta 8081



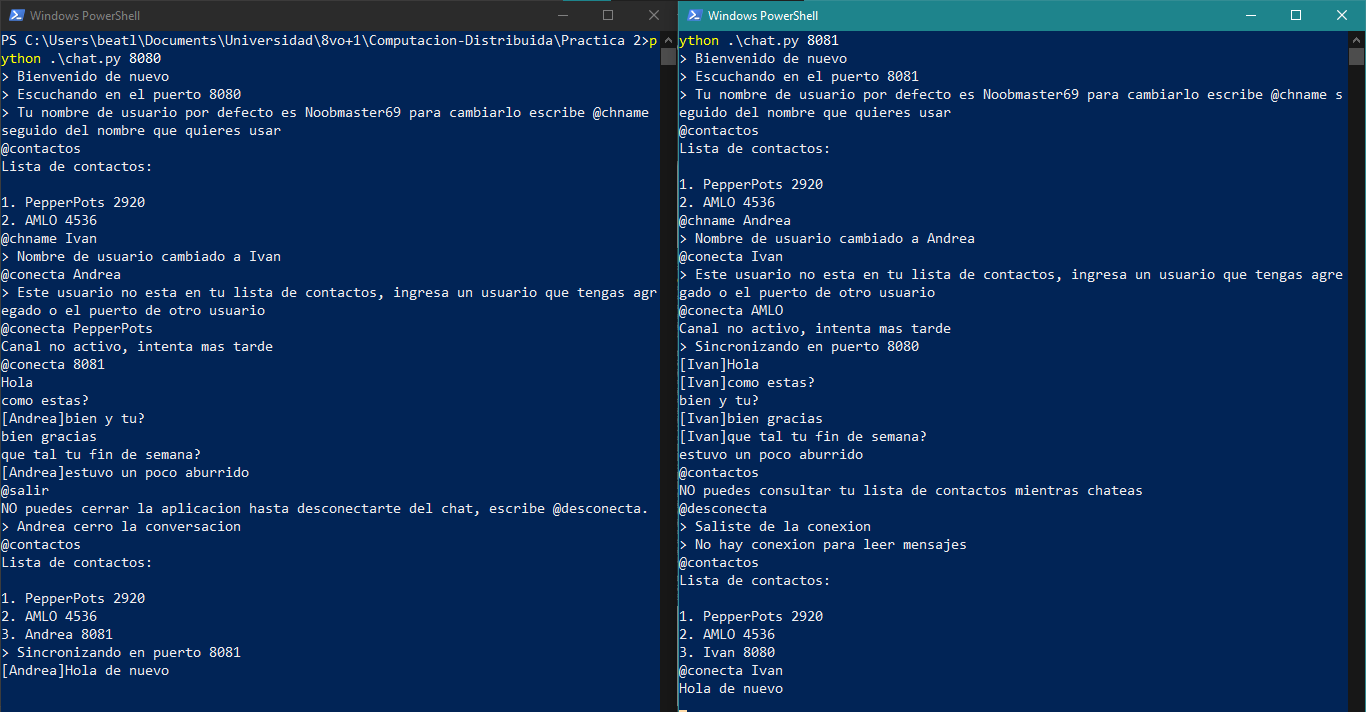
Como se puede observar del lado del usuario de la terminal derecha se le aparece un mensaje diciendo que se estableció una conexión con el puerto 8080 que es el puerto donde está conectado el usuario de la terminal izquierda. A partir de aquí se pueden mandar mensajes los 2 usuarios. Y además en esta parte no se pueden ocupar ningún comando excepto @desconecta.



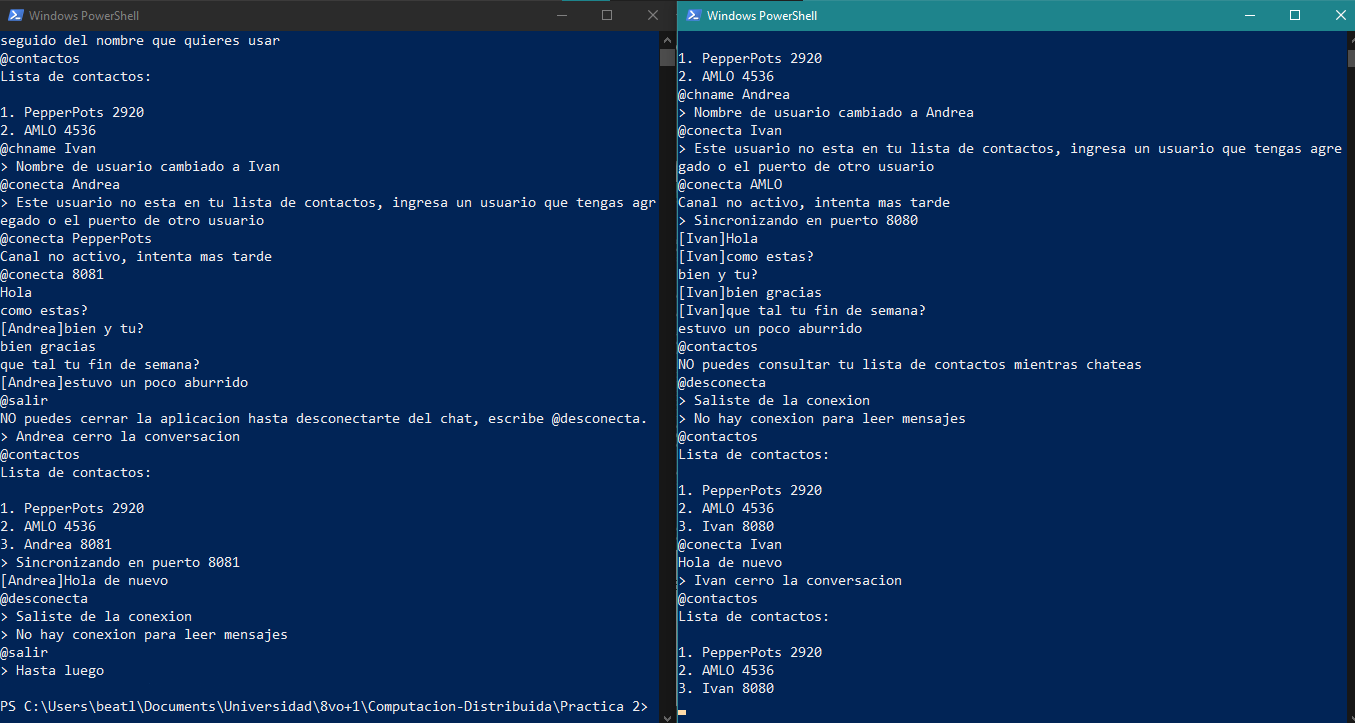
Para salir del chat basta con que alguno de los 2 ponga el comando @desconecta, y automáticamente la comunicación finaliza.



Ahora podemos comprobar que en ambos usuarios se registraron en la lista de contacto de cada uno, así al momento de querer conectarse de nuevo no va a ser necesario poner el puerto, simplemente se pondrá @conecta <nombre\_usuario>.



Por ultimo el comando @salir permite a cualquiera de los dos usuarios salir del programa sin afectar al otro usuario que sigue en el programa.



**Conclusiones**

Como se dijo al principio la parte Cliente de este programa atenderá todas aquellas tareas “locales”como las tareas hechas por los comandos @salir @contactos y @chname para cambiar el nombre de usuario, la parte servidor se encargará de todo lo que implique el intercambio de información entre los 2 usuarios como entablar una conexión con @conecta en donde se intercambia la información de los dos usuarios (puerto, nombre de usuario) para que posteriormente si es que aún no estaban agregados a sus listas de contactos estos se agreguen automáticamente después de entablar comunicación, desconectarse con @desconecta también ya que se recibe la información de que el usuario con el que se esta hablando ha cerrado la comunicación permitiendo que no se quede esperando en recibir mensajes y no pueda realizar otra tarea.

En lo personal al principio me costo entender como es que este modelo funcionaba, ya que no entendía muy bien como es que se podría tener las 2 partes juntas, pero quedó más claro una vez que empiezas analizar bien como es que se va a implementar la solución e incluso haciendo este reporte me di cuenta de cosas que pensé que ya tenía claro.

El enlace al repositorio es el siguiente: <https://github.com/Ivanjp/Computacion-Distribuida/tree/main/Practica%202>