

INTRODUCCIÓN A SISTEMAS DISTRIBUIDOS
PONTIFICIA UNIVERSIDAD JAVERIANA

MEMORIA HISTÓRICA

Michael Andrés Vargas
María Camila Jaramillo Benavides

Presentado a:
Enrique Gonzalez Guerrero

Bogotá

TABLA DE CONTENIDO

1. DESCRIPCIÓN DEL PROBLEMA	4
2. RESTRICCIONES	5
3. SOLUCIÓN.....	6
3.1. DIAGRAMAS UML	7
4. ESCENARIOS	11
5. REFERENCIAS	12

TABLA DE FIGURAS

FIGURA 1. ARQUITECTURA	5
FIGURA 2. DIAGRAMA DE PAQUETES.....	5
FIGURA 3. FILES/SRC	7
FIGURA 4. PEER/SRC.....	8
FIGURA 5. SIDECAR/SRC.....	9
FIGURA 6. TRACKER/SRC.....	10
FIGURA 7. SERVIDOR	11
FIGURA 8. CLIENTE.....	12

MEMORIA HISTÓRICA

1. DESCRIPCIÓN DEL PROBLEMA

Se propuso la realización de un sistema de memoria histórica haciendo uso de middlewares en ambientes de cómputo distribuidos, con el fin de proponer una solución basada en el modelo arquitectónico P2P. Esta entrega tuvo como fin el desarrollo de una memoria histórica de las víctimas del conflicto armado inspirada en el protocolo de BitTorrent.

Para el entendimiento del desarrollo de este proyecto, es necesario que el lector tenga conocimiento de los siguientes términos:

- **Archivo:** Fichero que contiene la información necesaria sobre el contenido (no tiene el contenido).
- **Índex:** Listado de ficheros manejados por el servidor de nombres.
- **Tracker:** Servidor especial que contiene la información necesaria para que los peers se conecten unos con otros.
- **Pieces:** Contenido dividido en trozos.
- **Peers:** Usuarios en la conexión.
- **Leechers:** Usuarios que están descargando el archivo, pero no lo tienen todavía completo.
- **Seeds:** Usuarios con el archivo completo.
- **Swarn:** Usuarios en general que el tracker busca. [1]

Para el desarrollo del proyecto, se hizo uso de los siguientes términos, con el fin de permitir a un usuario, generar un archivo que recapitule la información necesaria relacionada con el conflicto armado a partir de un conjunto de archivos fuente.

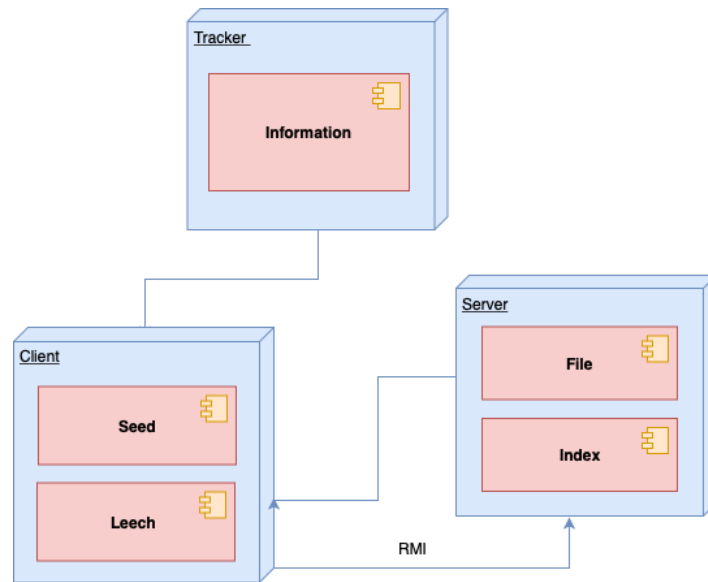


FIGURA 1. ARQUITECTURA

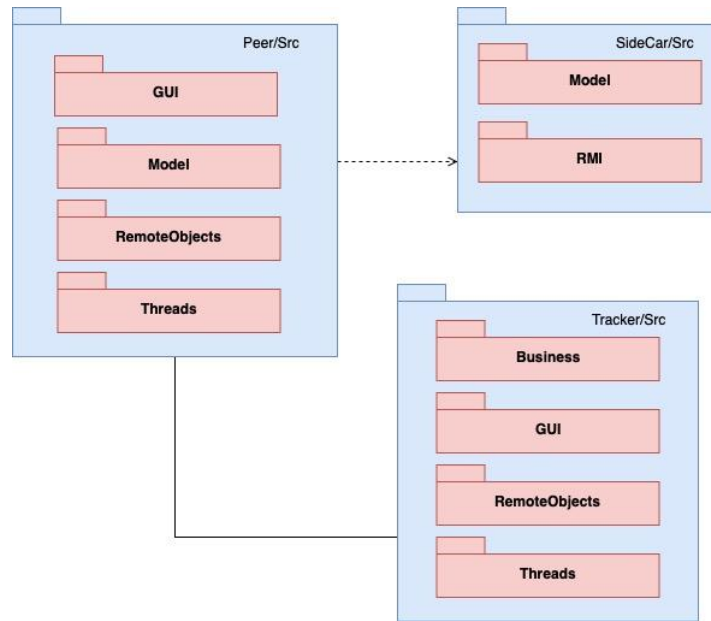


FIGURA 2. DIAGRAMA DE PAQUETES

2. RESTRICCIONES

El desarrollo debe inspirarse en el protocolo BitTorrent. Las descargas se deben manejar de manera paralela, teniendo en cuenta que todas las máquinas pueden ser fuentes y receptoras de archivos al mismo tiempo. La descarga de un archivo debe continuar cuando una máquina se desconecta. Sin embargo, deben evitarse mecanismos centralizados, teniendo en cuenta que se pueden generar cuellos de botella. Se cuenta con un proceso servidor único de nombres que puede ser el único componente centralizado del proyecto.

3. SOLUCIÓN

Para la solución se realizaron cuatro proyectos en los cuales, se distribuyó la carga de archivos, la distribución de estos y el cumplimiento del problema planteado siguiendo las restricciones especificadas. Para este proyecto en específico se realizó el envío de manera centralizada, teniendo en cuenta que por el tamaño de los archivos fue complejo evidenciar el paralelismo como en la descarga. A continuación, se podrán observar los diagramas de paquetes, con sus respectivas clases, teniendo en cuenta que, debido a la magnitud del proyecto, un diagrama de clases tendría un gran tamaño y no se podría observar de manera tan detallada, aún así en cada clase especificada, se pueden observar las listas y los objetos primarios en cada clase.

3.1. Diagrama de paquetes Files/Src

Este paquete tuvo como fin la lectura y la generación de archivos del proyecto. En este proyecto se realizó la lectura del archivo que contenía la información del cliente, es decir los respectivos archivos que deseaba, y por otro lado se realizó la lectura para la asignación de nuevos archivos con la respectiva información en la respectiva posición.

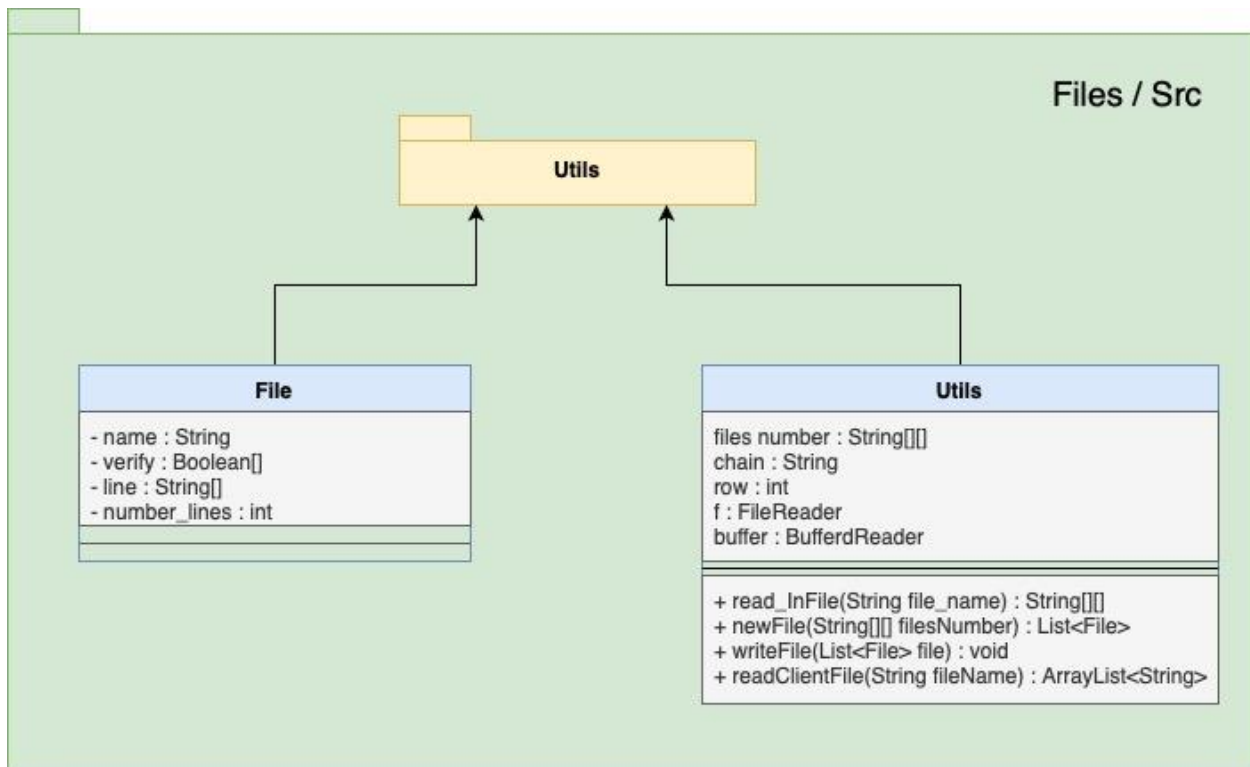


FIGURA 3. FILES/SRC

- **File:** Esta clase fue generada con el fin de estructurar los archivos que se iban a manejar.
- **ClientPeer:** Consta de cuatro funciones. `Read_inFile`, la cual recibe un `String` como parámetro que indica el nombre del archivo. Esta función tiene como fin leer el archivo que contiene el nombre y la línea, retornando una matriz con el nombre y la línea que se deseaba encontrar. `newFile`, recibe como parámetro una matriz de tipo `String`, la cual contiene el retorno de `Read_inFile`, esta función llena la clase del archivo, con la información obtenida en la matriz. `Write_file`, por otro lado, es el método que se encarga de escribir los archivos con la respectiva información guardada en la clase `File`. Finalmente, `readClientFile`, es un archivo con una lista de nombres los cuales son agregados a una lista.

3.2. Diagramas paquetes Peer/Src

Este paquete es el encargado de buscar y armar el archivo histórico.

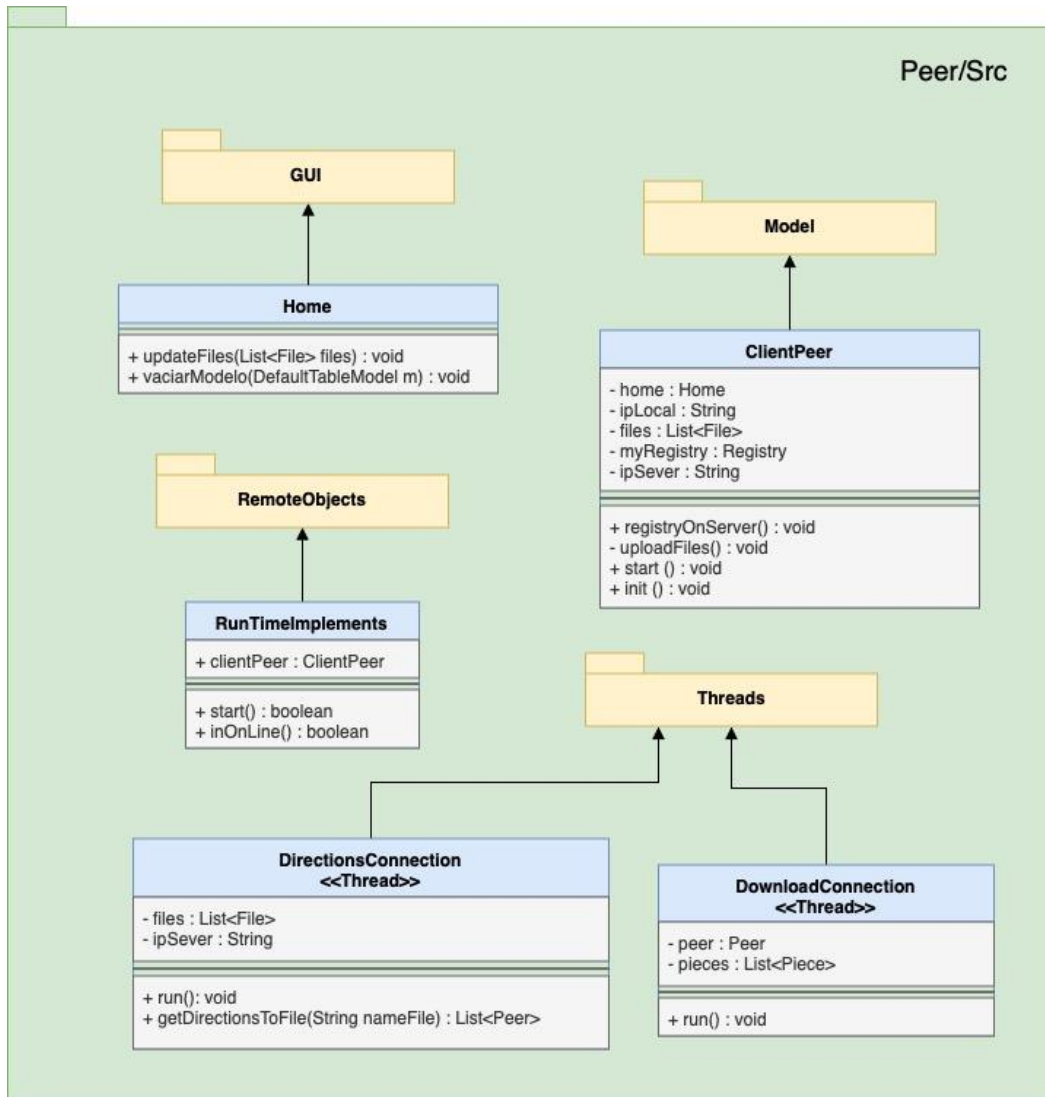


FIGURA 4. PEER/SRC

- **Home:** Clase definida para la visualización del usuario.
- **ClientPeer:** Clase fuente, se encarga de la lógica del consumidor y proveedor de los archivos.
- **RunTimeImplements:** Implementación de la interfaz ofrecida por el ClientPeer para eventos de inicio y para verificar que esté en línea.
- **DirectionsConnection:** Clase para preguntar al servidor de nombres las direcciones de los clientes o productores que tiene el archivo.
- **DownloadConnection:** Administrador de la descarga del histórico. Dispara los hilos para traer cada parte del documento y cuando tiene el archivo completo lo escribe en el nuevo archivo.

3.3. Diagrama paquetes SideCar/Src

Es la capa transversal que contiene las entidades de negocio de todo el sistema.

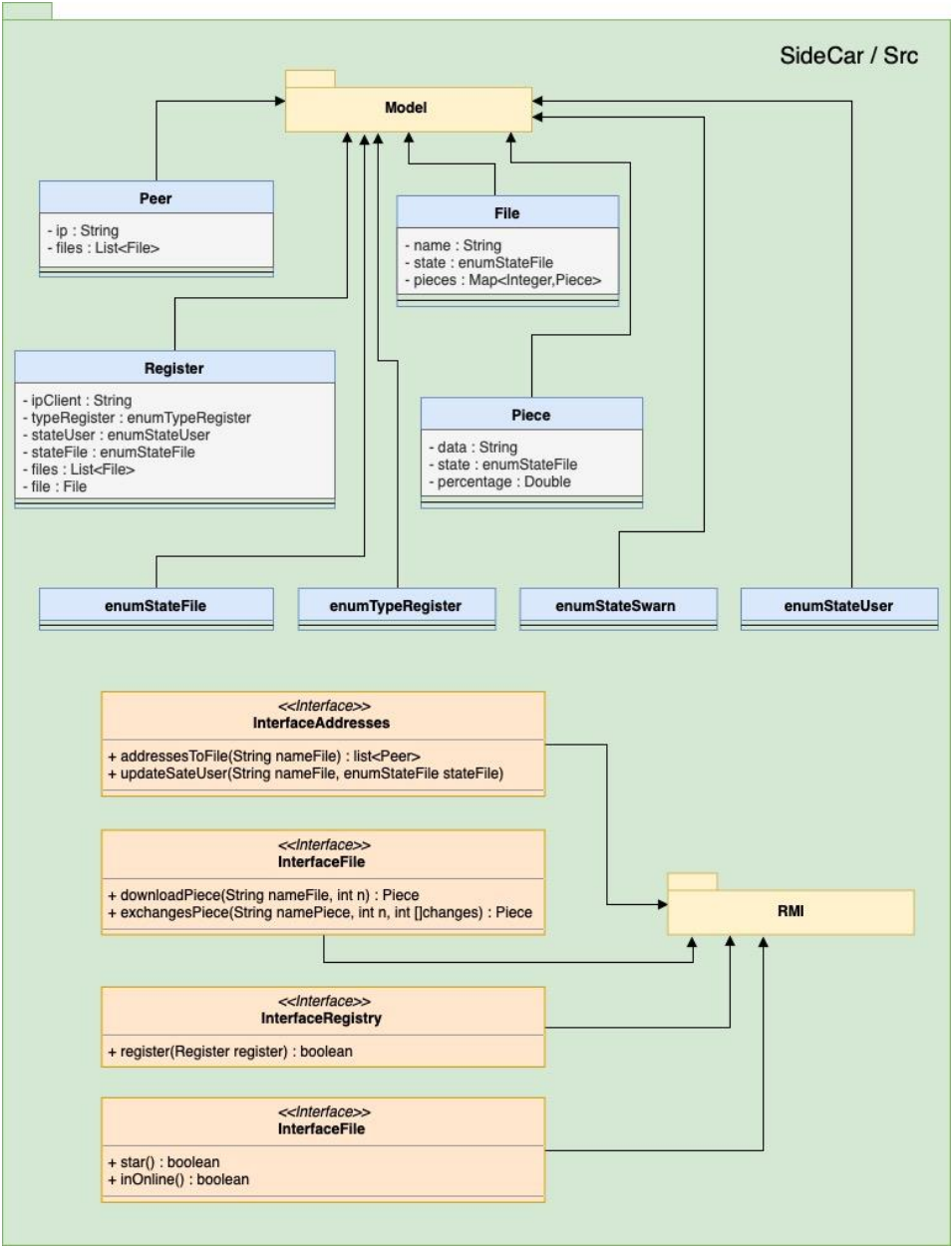


FIGURA 5. SideCAR/Src

3.4. Diagrama paquetes Tracker/Src

Este paquete contiene las clases del servidor de negocio y presentación. Este servidor es el encargado de contener toda la información respectiva a la dirección de los archivos.

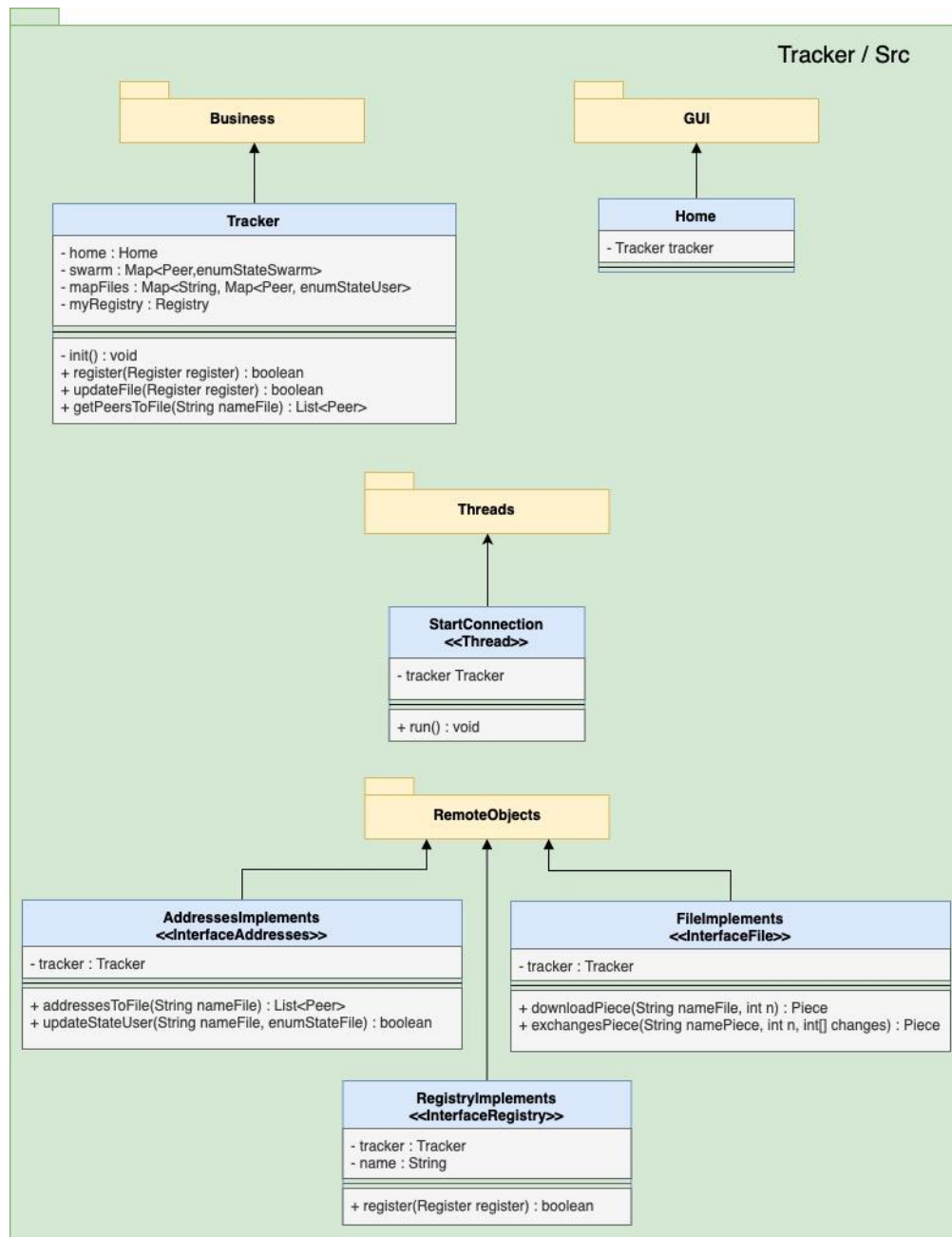


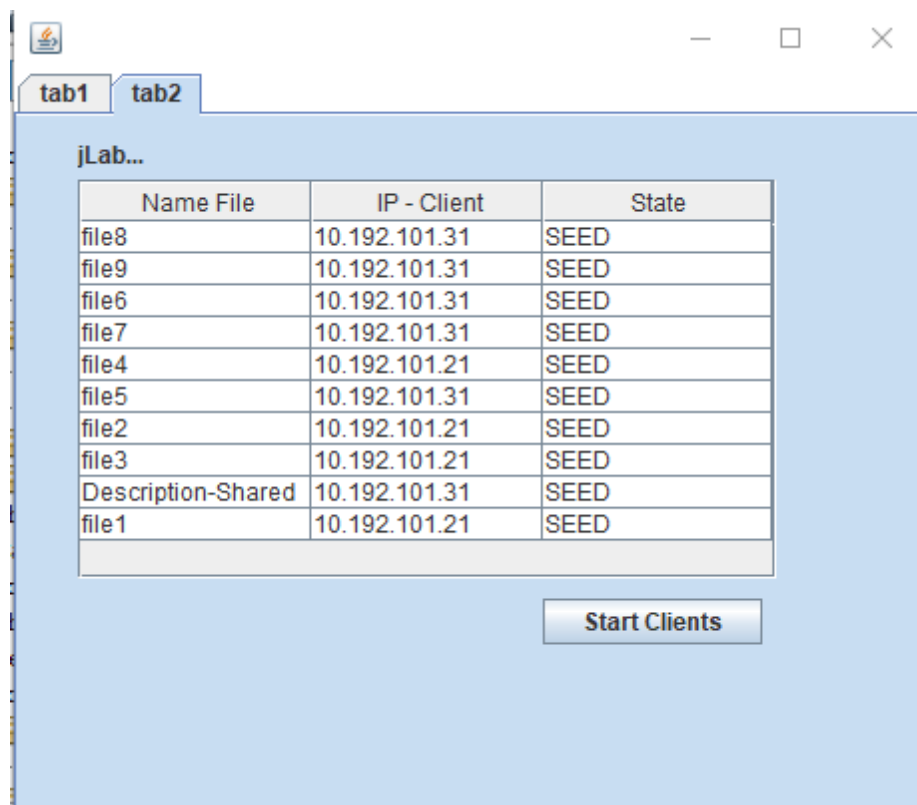
FIGURA 6. TRACKER/SRC

- **Tracker:** Es el administrador o gestor de la lógica de negocio del servidor de nombres.
- **Home:** Clase definida para la visualización de los Peers registrados y sus respectivos archivos.
- **StartConnection:** Es un hilo que se lanza para indicarle a los procesos clientes que inicien su proceso.

- **AdressesImplements:** Es la implementación de la interfaz prestada por el servidor para que los clientes consulten las direcciones de los servidores que contienen los archivos solicitados.
- **FileImplements:** Es la implementación de la interfaz prestada por peer para la descarga y carga de archivos.
- **RegistryImplements:** Es la implementación de la interfaz para que los peer se registren y informen qué archivos tienen.

4. ESCENARIOS

A continuación, se puede observar la interfaz del servidor en donde se puede observar la IP del cliente con su respectivo estado y el nombre del archivo que se compartió.



Name File	IP - Client	State
file8	10.192.101.31	SEED
file9	10.192.101.31	SEED
file6	10.192.101.31	SEED
file7	10.192.101.31	SEED
file4	10.192.101.21	SEED
file5	10.192.101.31	SEED
file2	10.192.101.21	SEED
file3	10.192.101.21	SEED
Description-Shared	10.192.101.31	SEED
file1	10.192.101.21	SEED

Start Clients

FIGURA 7. SERVIDOR

En este caso se puede observar la interfaz del cliente con sus respectivos archivos y su estado.

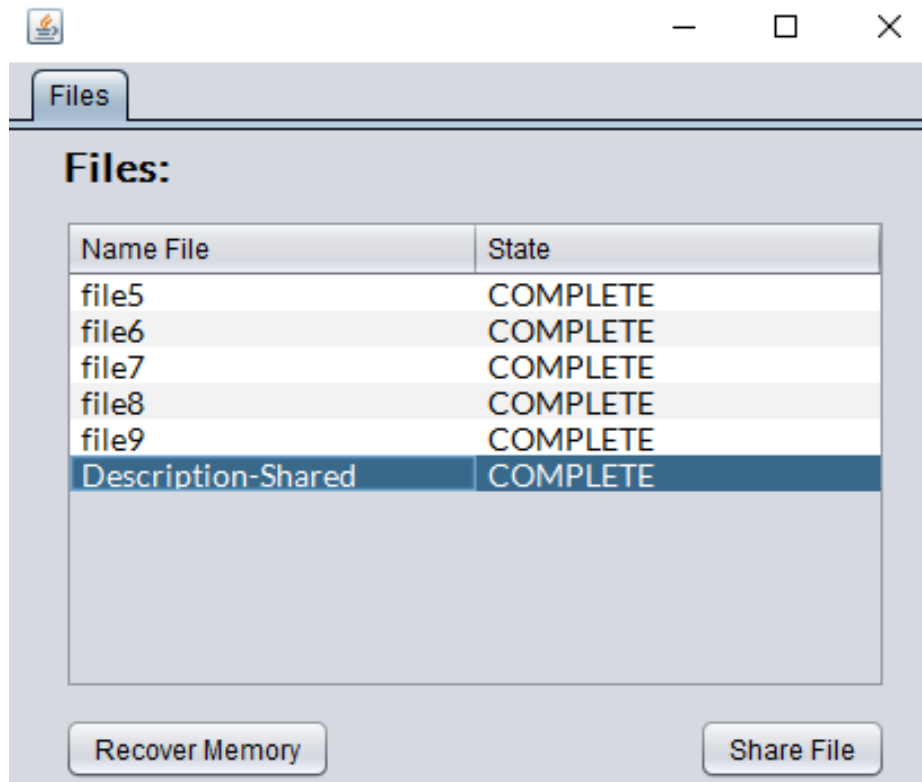


FIGURA 8. CLIENTE

5. REFERENCIAS

- [1] «Cómo funciona Bittorrent,» 2010. [En línea]. Available: <https://www.youtube.com/watch?v=IYbIW9R6HOW&t=589s>. [Último acceso: 17 Noviembre 2018].