

INDICE.

Memoria descriptiva.	3
El servidor.	3
Servicio de autenticación.	3
Servicio gestor.	3
Servicio de datos.	3
El repositorio.	3
Servicio servidor - operador.	3
Servicio cliente - operador.	3
El cliente.	3
Servicio disco - cliente.	3
Operativa general.	3
Operativa del servidor.	4
Operativa del repositorio.	4
Operativa del cliente.	4
Subir un fichero.	5
Bajar un fichero.	5
Borrar un fichero.	5
Listar ficheros.	6
Listar clientes.	6
Proyectos del sistema.	7
Common	7
Gui.	7
Utilis.	8
Fichero.	8
ServicioAutenticacionInterfase.	8
ServicioDatosInterfase.	8
ServicioGestorInterfase.	9
ServicioCloperadorInterfase.	9
ServicioSroperadorInterfase.	9
ServicioDiscoclienteInterfase.	9
Servidor.	9
Servidor.	9
ServicioAutenticacionImpl.	9
ServicioDatosImpl.	10
ServicioGestorImpl.	11
Repositorio.	12
Repositorio.	12
ServicioSroperadorImpl.	12
ServicioCloperadorImpl.	13
Cliente.	14
Cliente.	14
ServicioDiscoclienteImpl.	14
Diagrama de clases.	15
Ejemplos de funcionamiento.	16
Conclusiones y mejoras.	28
Código fuente.	30

MEMORIA DESCRIPTIVA.

El sistema básico de almacenamiento en la nube, se ha desarrollado mediante tres proyectos independientes, proyecto Servidor, proyecto Repositorio, proyecto Cliente, y un proyecto común a los tres proyectos anteriores, proyecto Common.

En el sistema de almacenamiento en la nube, hay tres tipos de actores:

- * El servidor: Proporciona tres tipos de servicios, autenticación, Gestor y Datos.
 - * Servicio de Autenticación: Este servicio, autentica y registra a los clientes y repositorios que se conecten al sistema.
 - * Servicio Gestor: Es el encargado de gestionar las subidas y bajadas de ficheros entre los clientes y los repositorios, pero sin hacer la subida y descarga propiamente dicha, la cual la hacen los propios clientes y repositorios. El servicio Gestor solo se limita en conectar al cliente con el repositorio correspondiente para que puedan hacer la subida o descarga de archivos.
 - * Servicio de Datos: Mantiene una serie de listas para almacenar información relativa a los clientes conectados al sistema, repositorios conectados al sistema y los repositorios asignados a cada cliente para almacenar sus archivos. El servicio de datos es privado para el servidor, ni los clientes ni los repositorios pueden acceder a él. Cualquier tipo de acceso que se realice a los datos manejados por el servicio de Datos, se hacen a través del servicio Gestor del servidor.
- * El repositorio: Es el encargado de almacenar en la nube los ficheros de los clientes que tiene asignados. Proporciona dos servicios:
 - * Servicio Servidor - Operador: Proporciona métodos para que el servidor pueda gestionar donde almacena cada cliente sus archivos. Además, realiza la bajada de ficheros desde el repositorio al cliente.
 - * Servicio Cliente - Operador: Se encarga de subir ficheros al repositorio que le corresponda al cliente, y borrar o borrar ficheros almacenados en el repositorio.
- * El cliente: Es el propietario de los ficheros y quien indica que ficheros subir, bajar y borrar de la nube. El cliente también proporciona un servicio:
 - * Servicio Disco - Cliente: Es el encargado de descargar un fichero desde el repositorio correspondiente, para ello, el servicio Servidor - Operador, utiliza el servicio Disco - Cliente para realizar la descarga.

Operativa general.

Al iniciarse el servidor, lanza sus tres servicios, autenticación, gestor y datos, a partir de este momento está listo para recibir peticiones de los clientes y repositorios.

Cuando se inicia un repositorio, este se autentica en el servidor mediante el servicio de autenticación, el cual le devuelve al repositorio un identificador único. Seguidamente el repositorio inicia sus servicios "servidor - operador" y "cliente - operador", quedando a partir de este momento listo para almacenar ficheros de clientes.

Al iniciarse un cliente, este introduce su nombre y se autentica en el servidor, devolviéndole este un identificador único. Seguidamente, el cliente inicia su servicio "disco - cliente", quedando a partir de este momento listo para subir, bajar y borrar ficheros.

Operativa del servidor.

Al iniciarse el servidor, comprueba si esta en ejecución rmi registry y en caso de no estarlo, lo inicia en el puerto 1099. Seguidamente inicia sus tres servicios "servicio de autenticación", "servicio gestor" y "servicio de datos".

El servicio de datos implementa tres tipos de listas, una lista de clientes que relaciona el identificador único de cada cliente que el nombre del cliente, una lista de repositorios que relaciona el identificador único de cada repositorio con el nombre de cada repositorio y una lista de clientes - repositorios que relaciona el identificador de cada cliente con el identificador del repositorio que le ha sido asignado para almacenar sus ficheros.

A partir de este momento, el servidor queda listo para recibir peticiones de los clientes y repositorios.

Por ultimo, el servidor muestra en pantalla un menú mediante el cual se puede mostrar en pantalla una lista de los clientes conectados, una lista de los repositorios conectados y una lista de los repositorios asignados a cada cliente.

Operativa del repositorio.

Al iniciarse el repositorio, hace una llamada al servicio de autenticación del servidor. El servicio de autenticación genera un identificador único para el repositorio y un nombre y llama al servicio datos para registrar al repositorio en la lista de repositorios conectados. Por ultimo, el servicio de autenticación devuelve la repositorio su identificador único en el sistema. Seguidamente, el repositorio inicia sus servicios "servidor - operador" y "cliente - operador", quedando listo para almacenar ficheros de clientes. El servicio "servidor - operador", mantiene de forma privada y local dos lista, una lista de clientes que almacena el identificador del los clientes que están asignados al repositorio y una lista de ficheros que relaciona los identificadores de los clientes con sus ficheros almacenados en el repositorio.

El repositorio muestra en pantalla un menú mediante el cual se puede mostrar en pantalla una lista de los clientes asociados al repositorio y una lista de los ficheros que cada cliente tiene almacenados en el repositorio.

Operativa del cliente.

Al iniciarse un cliente, hace una llamada al servicio de autenticación del servidor.

El servicio de autenticación, hace una llamada al servicio de datos, el cual busca el identificador del repositorio que este menos cargado en ese momento, es decir, el repositorio que tenga asignados menos clientes. El servicio de datos devuelve al servicio de autenticación dicho identificador de repositorio. Seguidamente, el servicio de autenticación da de alta al cliente en la lista de clientes conectados al sistema y en la lista que relaciona al cliente con el repositorio que le ha sido asignado.

Por ultimo, el servicio de autenticación llama al servicio servidor - operador del repositorio asignado al cliente para que cree una carpeta en el repositorio con el identificador único del cliente por nombre y devuelve al cliente su identificador único.

El cliente muestra en pantalla un menú mediante el cual puede subir un fichero al repositorio, bajar un fichero, borrar un fichero, listar los ficheros que tiene almacenados en el repositorio y mostrar los clientes que hay conectados.

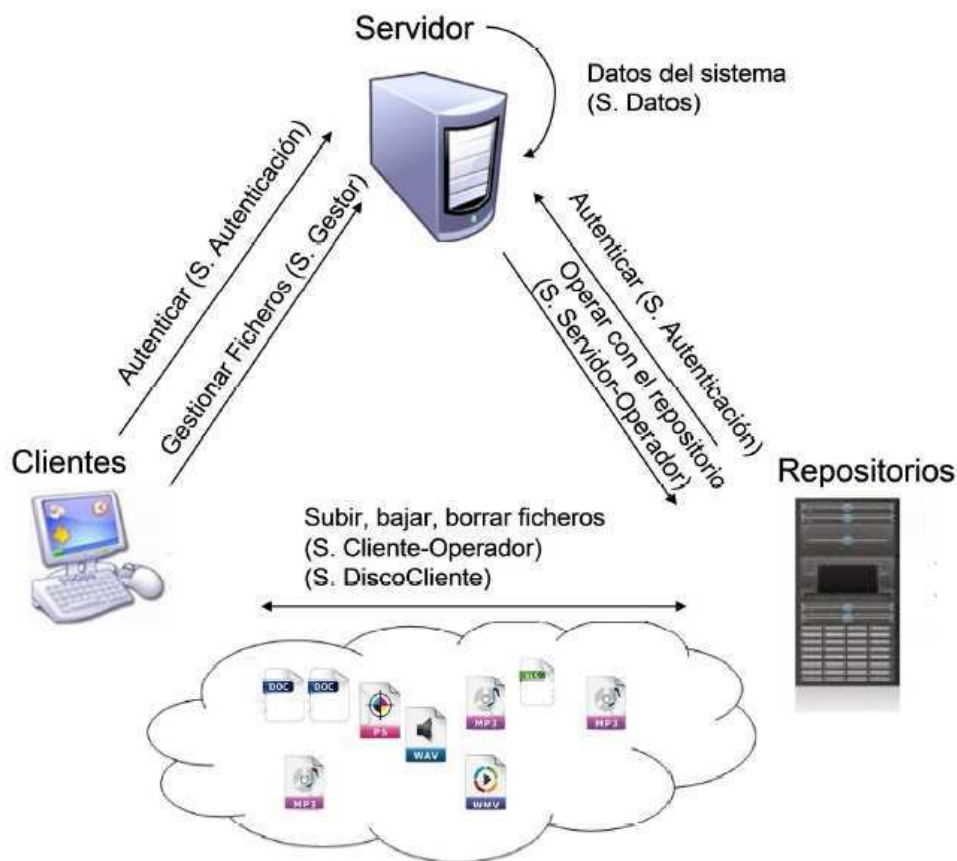
* Subir un fichero: Al subir un fichero, lo primero que hace el cliente es comprobar si el fichero existe en disco. Una vez realizada la comprobación, el cliente llama al servicio gestor del servidor indicándole su intención de subir un fichero y le pasa su identificador único y el nombre del fichero que quiere subir. El servicio gestor de servidor, se encarga, a partir de este momento, de localizar el repositorio que tiene asignado el cliente, en el cual esta almacenado el fichero, para ello el servicio gestor hace una llamada al servicio de datos preguntando por el identificador del repositorio asignado al cliente. El servicio de datos, busca en la lista de repositorios asignados a los clientes el identificador del cliente y obtiene el identificador del repositorio asignado, el cual le devuelve al servicio gestor. Seguidamente, el servicio gestor, una vez tiene el identificador del repositorio asignado al cliente, obtiene el nombre del servicio "cliente - operador" del repositorio y se lo devuelve al cliente, conectando de esta manera al cliente con su repositorio. Una vez el cliente tiene el nombre del servicio cliente - operador del repositorio que le ha sido asignado, hace una llamada al servicio cliente - operador indicándole su identificador de cliente y el fichero que quiere subir. El servicio cliente - operador del repositorio, almacena mediante stream, en la carpeta asignada al cliente, el archivo que el cliente quiere subir y devuelve al cliente true, si la subida del archivo se ha completado con éxito y false si ha ocurrido algún tipo de error.

* Bajar un fichero: Cuando un cliente quiere descargar un fichero, llama al servicio gestor del servidor y le pasa su identificador único de cliente, el nombre del fichero que quiere descargar y el nombre de su servicio disco - cliente, que utilizara el repositorio para descargar el fichero en cuestión al cliente. El servicio gestor, hace una llamada al servicio de datos para localizar el identificador único del repositorio asignado al cliente. El servicio de datos, buscara en la lista de repositorios asignados a clientes el identificador del repositorio asignado al cliente en cuestión y se lo devuelve al servicio gestor. Seguidamente, una vez el servicio gestor tiene el identificador del repositorio asignado al cliente, llama al servicio servidor - operador del repositorio y le hace una llamada para descargar el archivo, pasándole el identificador del cliente, el nombre del archivo a descargar y el nombre del servicio disco - cliente del cliente. El servicio servidor - operador del repositorio comprueba que existe el fichero a descargar en la carpeta del cliente, en cuyo caso, llama al servicio disco - cliente que es quien realiza la descarga al cliente por stream.

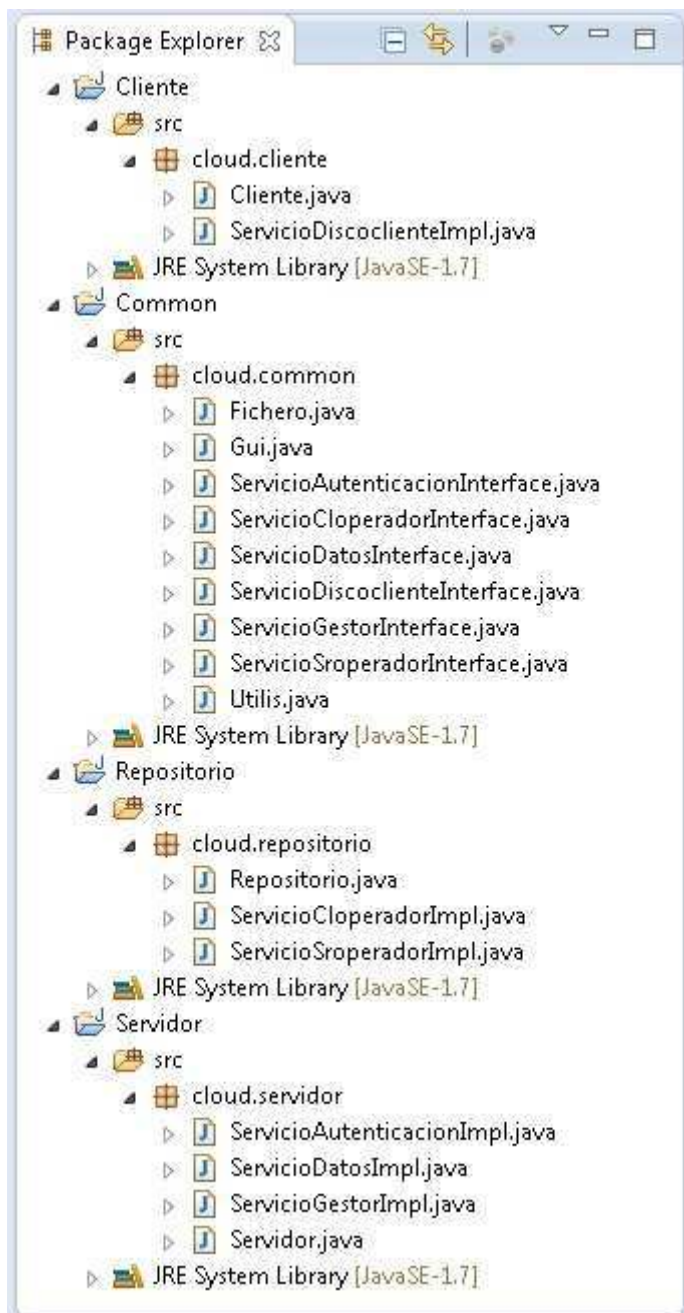
* Borrar un fichero: Cuando un cliente quiere borrar un fichero, llama al servicio gestor del servidor para obtener el nombre del servicio cliente - operador del repositorio que tiene asignado. El cliente, para ello, le pasa al servicio gestor su identificador de cliente. El servicio gestor del servidor, llama al servicio de datos para localizar el repositorio asignado al cliente, el servicio de datos busca en su lista de repositorios asignados a clientes el identificador del cliente y obtiene el identificador del repositorio asignado a dicho cliente, devolviendo el identificador del repositorio al servicio gestor. El servicio gestor, obtiene el nombre del servicio cliente - operador del repositorio y se lo devuelve al cliente. Una vez tiene el cliente el nombre del servicio cliente - operador del repositorio que tiene asignado, llama a dicho servicio y le pasa su identificador de cliente y el nombre del fichero que desea borrar. El servicio cliente - operador del repositorio, comprueba si existe en la carpeta del cliente el fichero que este quiere borrar y lo elimina.

*Listar ficheros: Para listar los ficheros que el cliente tiene almacenados en el repositorio, hace una llamada al servicio gestor del servidor pasándole su identificador de cliente. El servicio gestor, llama al servicio de datos para obtener el identificador del repositorio asignado al cliente. El servicio de datos busca en su lista de repositorios asignados a clientes el identificador del cliente y obtiene el identificador del repositorio asignado, devolviendo el identificador del repositorio al servicio gestor. Seguidamente, el servicio gestor, llama al servicio servidor - operador del repositorio asignado al cliente. El servicio servidor - operador, genera una lista con los meta datos de los ficheros que el cliente tiene almacenados en el repositorio y le devuelve la lista al servicio gestor. Una vez que el servicio gestor tiene la lista de ficheros del cliente, se la devuelve al cliente, el cual la mostrara en pantalla.

* Listar clientes: Para listar los clientes conectados al sistema, el cliente llama al servicio gestor. El servicio gestor llama al servicio de datos, pidiéndole la lista de clientes. El servicio de datos devuelve al servicio gestor dicha lista, que a su vez devuelve el servicio gestor al cliente. Una vez tiene el cliente la lista de clientes conectados, la muestra en pantalla.



PROYECTOS DEL SISTEMA.



Common.

Es un proyecto con código común a los otros tres proyectos.

* **Gui:** Clase utilizada para implementar la interfaz en modo texto de menús del servidor, repositorio y cliente.

*Métodos de la clase Gui:

```
public static int MostrarMenu(String name, String[] entradas) {  
    Muestra en pantalla un menú de nombre "name" con las opciones  
    indicadas en el String "entradas" y devuelve la opción seleccionada del  
    menú.
```

public static void LimpiarPantalla() throws IOException {
Borra la pantalla y queda a la espera de pulsar una tecla.

public static String IntroducirString(String texto) throws IOException {
Captura las pulsaciones de teclado y las devuelve en forma de String.

* **Utilis:** Contiene el CODEBASE que nos permitirá establecer el PATH de la ruta del class. Al pasarle una clase, se obtiene una ruta donde esta cargada. También permite iniciar rmi registry.

*Métodos de la clase Utilis:

public static void SetCodeBase(Class<?> c){
Establece el PATH de la ruta de Class.

public static void startRegistry(int RMIPortNum) throws RemoteException{
Inicia rmi registry en el puerto RMIPortNum.

* **Fichero:** Clase serializable para realizar el envío de ficheros por stream. Proporcionado por el equipo docente.

*Métodos de la clase Fichero:

public Fichero (String nombre, String propietario) {
Constructor de la clase Fichero. Crea un nuevo fichero con nombre "nombre" y cuyo propietario es "propietario".

public Fichero (String ruta, String nombre, String propietario){
Constructor de la clase Fichero. Crea un nuevo fichero en la ruta "ruta" con nombre "nombre" y cuyo propietario es "propietario".

public boolean escribirEn (java.io.OutputStream os){
Envía un fichero mediante stream.

public String obtenerPropietario(){
Devuelve el propietario del fichero.

public String obtenerNombre(){
Devuelve el nombre del fichero.

public long obtenerPeso(){
Devuelve en bytes el tamaño (peso) del fichero.

public long obtenerChecksum(){
Devuelve el checksum del fichero.

* **ServicioAutenticacionInterface:** Interface del servicio de autenticación. Se implementa en el servidor en la clase ServicioAutenticacionImpl.

* **ServicioDatosInterface:** Interface del servicio de datos. Se implementa en el servidor en la clase ServicioDatosImpl.

* **ServicioGestorInterface:** Interface del servicio gestor. Se implementa en el servidor en la clase ServicioGestorImpl.

* **ServicioCloperadorInterface:** Interface del servicio cliente - operador. Se implementa en el repositorio en la clase ServicioCloperadorImpl.

* **ServicioSroperadorInterface:** Interface del servicio servidor - operador. Se implementa en el repositorio en la clase ServicioSroperadorImpl.

* **ServicioDiscoclienteInterface:** Interface del servicio disco - cliente. Se implementa en el cliente en la clase ServicioDiscoclienteImpl.

Servidor.

Proyecto que implementa el servidor del sistema de almacenamiento en la nube.

* **Servidor:** Clase que implementa el main del servidor. Inicia rmi registry, inicia los servicios de autenticación, datos y gestor del servidor y muestra el menú de opciones del servidor.

* Métodos de la clase servidor:

**private static void listar_clientes() throws RemoteException,
NotBoundException {**

Implementa la opción listar clientes del menú del servidor. Muestra en pantalla una lista de los clientes conectados al sistema.

**private static void listar_repositorios() throws RemoteException,
NotBoundException {**

Implementa la opción listar repositorios del menú del servidor. Muestra en pantalla una lista de los repositorios conectados al sistema.

**private static void listar_clirep() throws RemoteException,
NotBoundException {**

Implementa la opción clientes repositorios del menú del servidor. Muestra en pantalla una lista de los clientes conectados al sistema junto con su repositorio asignado.

* **ServicioAutenticacionImpl:** Clase que implementa la interface ServicioAutenticacionInterface.

* Métodos de la clase ServicioAutenticacionImpl:

**public int Autenticar_cliente(String nombre) throws RemoteException,
NotBoundException;**

Autentica un cliente en el sistema devolviéndole un identificador único.

**public void Desconectar_cliente(int id) throws RemoteException,
NotBoundException;**

Desconecta del sistema al cliente con identificador id, borrándolo de las listas del servicio de datos.

**public int Autenticar_repositorio() throws RemoteException,
NotBoundException;**

Autentica un repositorio en el sistema devolviéndole un identificador único.

**public void Desconectar_repositorio(int id) throws RemoteException,
NotBoundException;**

Desconecta del sistema al repositorio con identificador id, borrándolo de las listas del servicio de datos.

public int getPuerto() throws RemoteException;

Devuelve un numero de puerto.

* **ServicioDatosImpl:** Clase que implementa la interface ServicioDatosInterface.

* Métodos de la clase ServicioDatosImpl:

**public void lista_clientes_set_cliente(int id, String nombre) throws
RemoteException;**

Introduce un cliente con identificador "id" y nombre "nombre" en la lista de clientes.

**public int lista_clientes_get_idcliente(String nombre) throws
RemoteException;**

Devuelve el "id" del cliente con nombre "nombre" de la lista de clientes.

**public String lista_clientes_get_nombrecliente(int id) throws
RemoteException;**

Devuelve el nombre del cliente con identificador "id".

public void lista_clientes_borrar_cliente(int id) throws RemoteException;

Borra al cliente con identificador "id" de la lista de clientes.

public int lista_clientes_get_numclientes() throws RemoteException;

Devuelve el numero de clientes registrados en la lista de clientes.

**public Map<Integer, String> lista_clientes_obtener_lista() throws
RemoteException;**

Devuelve la lista de clientes.

**public void lista_repositorios_set_repositorio(int id, String nombre) throws
RemoteException;**

Introduce un repositorio con identificador "id" y nombre "nombre" en la lista de repositorios.

**public int lista_repositorios_get_idrepositorio(String nombre) throws
RemoteException;**

Devuelve el identificador "id" del repositorio con nombre "nombre".

public String lista_repositorios_get_nombrerepositorio(int id) throws RemoteException;

Devuelve el nombre del repositorio con identificador "id".

public void lista_repositorios_borrar_repositorio(int id) throws RemoteException;

Borra al repositorio con identificador "id" de la lista de repositorios.

public int lista_repositorios_get_numrepositorios() throws RemoteException;

Devuelve el numero de repositorios registrados en la lista de repositorios.

public Map<Integer, String> lista_repositorios_obtener_lista() throws RemoteException;

Devuelve la lista de repositorios.

public void lista_clirep_set_clirep(int idcli, int idrep) throws RemoteException;

Introduce el identificador de cliente "idcli" y el identificador de repositorio "idrep" que están asociados lista de clientes - repositorios asociados.

public int lista_clirep_get_idrepositorio(int idcli) throws RemoteException;

Devuelve el identificador del repositorio asignado al cliente con identificador "idcli".

public int lista_clirep_get_idcliente(int idrep) throws RemoteException;

Devuelve un identificador de un cliente que este asociado al repositorio con identificador "idrep".

public void lista_clirep_borrar_clirep(int idcli) throws RemoteException;

Borra de la lista de asociaciones de clientes repositorio las asociación del cliente "idcli" con su repositorio.

public int lista_clirep_get_numclirep() throws RemoteException;

Devuelve el numero de asociaciones entre clientes y repositorios.

public Map<Integer, Integer> lista_clirep_obtener_lista() throws RemoteException;

Devuelve la lista de asociaciones entre clientes y repositorios.

public int repositorio_menos_cargado() throws RemoteException;

Devuelve el identificador del repositorio menos cargado, es decir, el identificador del repositorio que menos clientes tenga asignados.

* **ServicioGestorImpl:** Clase que implementa la interface ServicioGestorInterface.

* Métodos de la clase ServicioGestorImpl:

public String Localizar_repositorio(int id_cli) throws RemoteException, NotBoundException;

Devuelve el nombre del repositorio asignado al cliente "id_cli".

public void Bajar_fichero(int id_cli, String nombre_fichero, String cliente) throws RemoteException, NotBoundException;

Gestiona la acción de bajar un fichero desde un repositorio al cliente, sin hacer la descarga propiamente dicha.

public List<String> listar_ficheros(int id_cli) throws RemoteException, NotBoundException;

Devuelve una lista de los ficheros del cliente con identificador "id_cli" almacenados en su repositorio asociado.

public Map<Integer, String> listar_clientes() throws RemoteException, NotBoundException;

Devuelve una lista con los clientes conectados al sistema.

public List<Fichero> listar_metadatos(int id_cli) throws RemoteException, NotBoundException;

Devuelve una lista con los meta datos de los ficheros del cliente con identificador "id_clip" almacenados en el repositorio que tiene asignado.

Repositorio.

Proyecto que implementa un repositorio del sistema de almacenamiento en la nube.

* **Repositorio:** Clase que implementa el main del repositorio. Autentica el repositorio en el sistema, inicia los servicios "servidor - operador" y "cliente - operador" y muestra el menú de opciones del repositorio.

* Métodos de la clase repositorio:

private static void listar_clientes() throws RemoteException, NotBoundException;

Muestra en pantalla los clientes asociados al repositorio.

private static void listar_ficheros_clientes() throws RemoteException, NotBoundException;

Muestra en pantalla la lista de clientes y sus ficheros que hay en el repositorio.

* **ServicioSroperadorImpl:** Clase que implementa la interface ServicioSroperadorInterface.

* Métodos de la clase ServicioSroperadorImpl:

public void descargar_fichero(int id_cli, String nombre_fichero, String nombre_servicioDiscocliente) throws RemoteException, NotBoundException;

Descarga un fichero con nombre "nombre_fichero" desde el repositorio al cliente con identificador "id_cli" mediante el servicio disco - cliente del cliente "id_cli".

public void lista_clientes_set_cliente(int id_cli, String nombre) throws RemoteException;

Introduce un cliente con identificador "id_cli" y nombre "nombre" en la lista local de clientes del repositorio.

public Map<Integer, String> lista_clientes_obtener_lista() throws RemoteException;

Devuelve la lista local de clientes asociados al repositorio.

public void lista_ficheros_set_fichero(int id_cli, String nombre_fichero) throws RemoteException;

Introduce el nombre de un fichero "nombre_fichero" de un cliente con identificador "id_cli" en la lista local de ficheros del repositorio.

public void lista_ficheros_borrar_fichero(int id_cli, String nombre_fichero) throws RemoteException;

Borra un fichero de nombre "nombre_fichero" cuyo propietario es el cliente "id_cli" de la lista local de ficheros del repositorio.

public Map<Integer, List<String>> lista_ficheros_obtener_lista() throws RemoteException;

Devuelve la lista local de ficheros almacenados en el repositorio.

public List<String> lista_ficheros_obtener_lista_ficheroscli(int id_cli) throws RemoteException;

Devuelve una lista con los nombres de los ficheros almacenados en el repositorio que son propiedad del cliente "id_cli".

public void set_id_repositorio(int id) throws RemoteException;

Asigna el identificador único al repositorio.

public void crear_directorio(int id_cli) throws RemoteException;

Crea en el disco del repositorio un directorio para el cliente "id_cli".

public List<Fichero> listar_ficheros_metadatos(int id_cli) throws RemoteException;

Devuelve una lista con los meta datos de los ficheros que pertenecen al cliente "id_cli", almacenados en el repositorio.

* **ServicioCloperadorImpl:** Clase que implementa la interface ServicioCloperadorInterface.

* Métodos de la clase ServicioCloperadorImpl:

public boolean Subir_fichero(int id_cli, String nombre_fichero, Fichero FICHERO) throws RemoteException, NotBoundException, FileNotFoundException, IOException;

Sube al repositorio desde el cliente con identificador "id_cli", un fichero "FICHERO" con nombre "nombre_fichero".

public void borrar_fichero(int id_cli, String nombre_fichero) throws RemoteException, NotBoundException;

Borra del repositorio el fichero "nombre_fichero" que es propiedad del cliente "id_cli".

public void set_id_repositorio(int id) throws RemoteException;

Asigna el identificador único al repositorio.

Cliente.

Proyecto que implementa un cliente del sistema de almacenamiento en la nube.

* **Cliente:** Clase que implementa el main del cliente. Autentica el cliente en el sistema, inicia el servicio "disco - cliente" muestra el menú de opciones del cliente.

* Métodos de la clase cliente:

**private static void subir_fichero() throws IOException,
NotBoundException;**

Implementa la opción subir fichero del menú cliente. Llama al servicio gestor del servidor para obtener el repositorio que tiene asignado el cliente, para después llamar al servicio "cliente - operador" del repositorio asignado para realizar la subida del fichero.

**private static void bajar_fichero() throws IOException,
NotBoundException;**

Implementa la opción bajar fichero del menú cliente. Llama al servicio gestor, pasándole el nombre del servicio "disco - cliente", el cual, pasara el servicio gestor al repositorio adecuado para que inicie la descarga del fichero al cliente.

**private static void borrar_fichero() throws IOException,
NotBoundException;**

Implementa la opción borrar del menú cliente. Llama al servicio gestor del servidor para localizar el repositorio que tiene asignado. Seguidamente llama al servicio "cliente - operador" del repositorio pasándole el nombre del fichero que desea borrar.

**private static void listar_ficheros() throws IOException,
NotBoundException;**

Muestra en pantalla un listado de los ficheros que el cliente tiene almacenados en el repositorio que tiene asignado.

**private static void listar_clientes() throws IOException,
NotBoundException;**

Muestra en pantalla una lista de los clientes conectados al sistema.

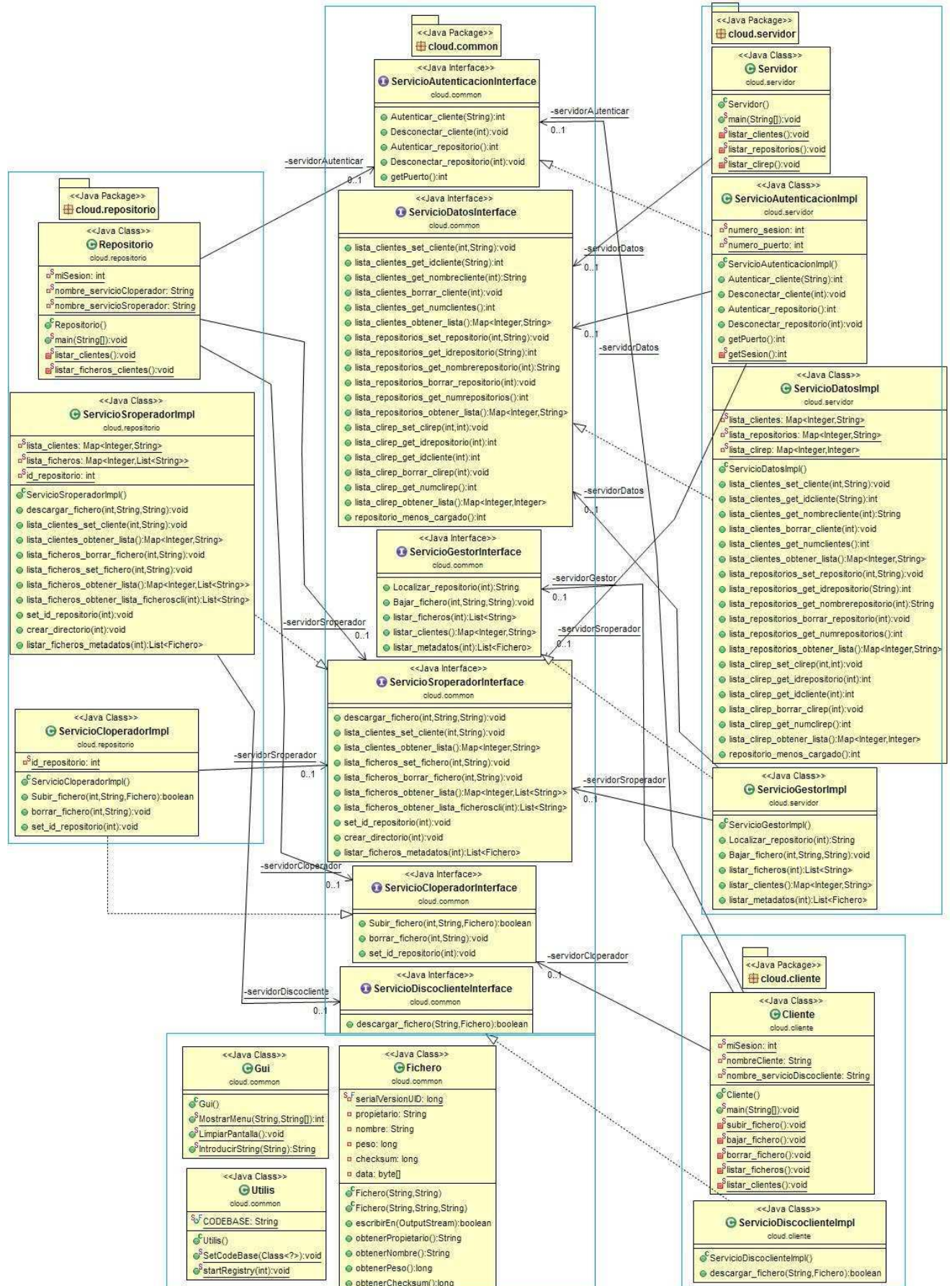
* **ServicioDiscoclienteImpl:** Clase que implementa la interface ServicioDiscoclienteInterface.

* Métodos de la clase ServicioDiscoclienteImpl:

**public boolean descargar_fichero(String nombre_fichero, Fichero fichero)
throws RemoteException;**

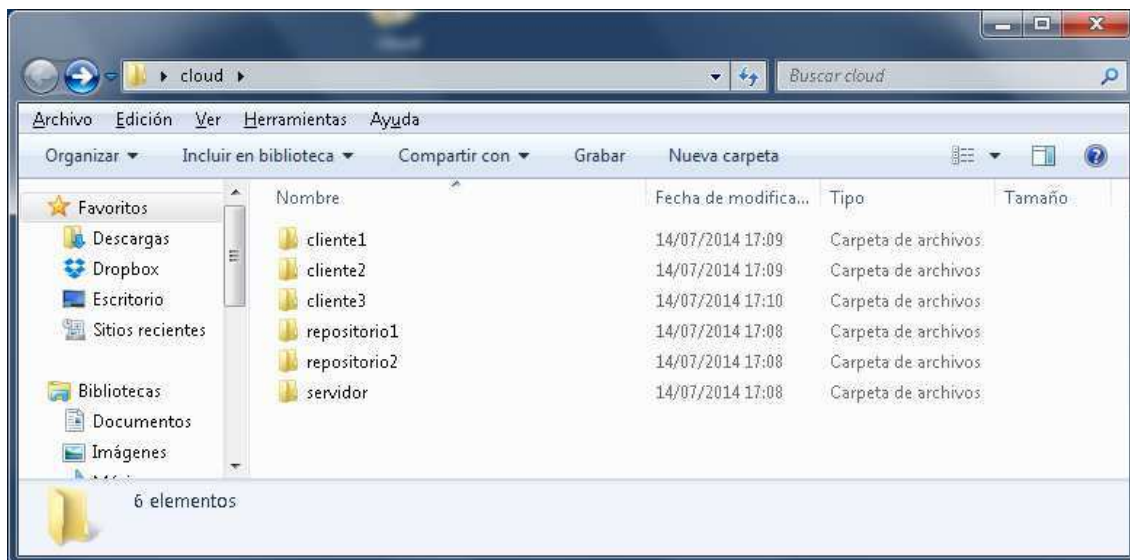
Descarga el fichero "Fichero" con nombre "nombre_fichero" desde el repositorio asociado al cliente, hasta el disco local del cliente.

DIAGRAMA DE CLASES.



EJEMPLOS DE FUNCIONAMIENTO.

Seguidamente, se muestra la ejecución del sistema, iniciando el servidor, dos repositorios y tres clientes. Para ello, los programas se van a localizar en una carpeta "cloud" en el escritorio de windows. Dentro de "cloud", cada uno de los actores se localizara en una subcarpeta, "servidor" para el servidor, "repositorio1" para el primer repositorio, "repositorio2" para el segundo repositorio, "cliente1" para el primer cliente, "cliente2" para el segundo cliente y "cliente3" para el tercer cliente.

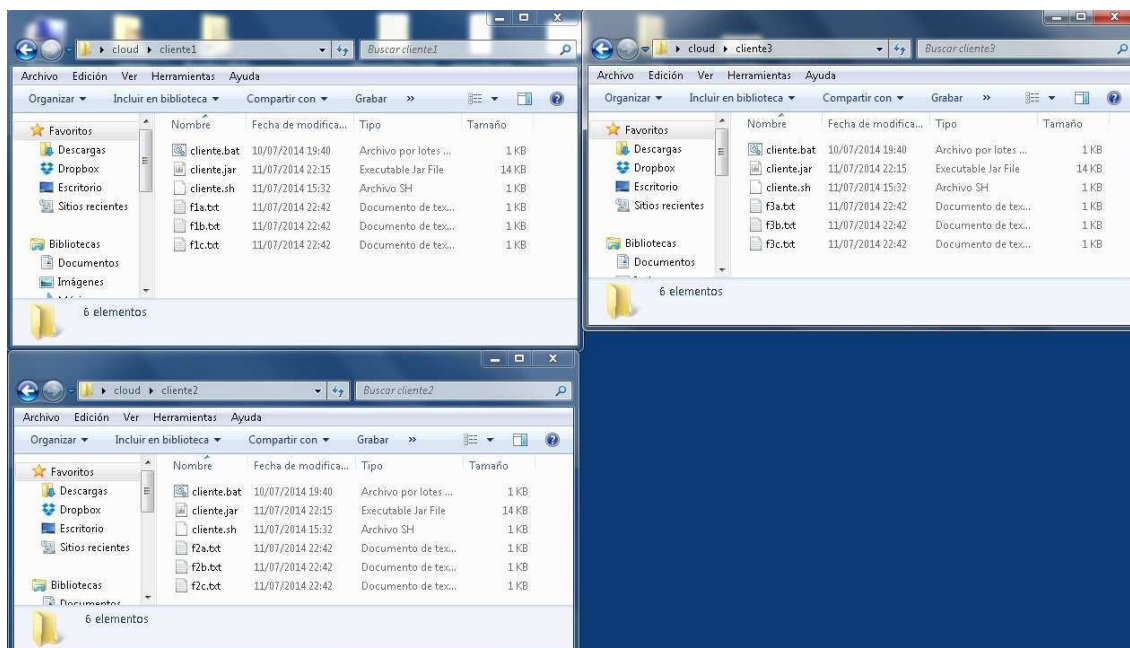


Cada cliente dispone, dentro de su carpeta, de tres archivos.

El cliente1 dispone de los archivos "f1a.txt", "f1b.txt" y "f1c.txt".

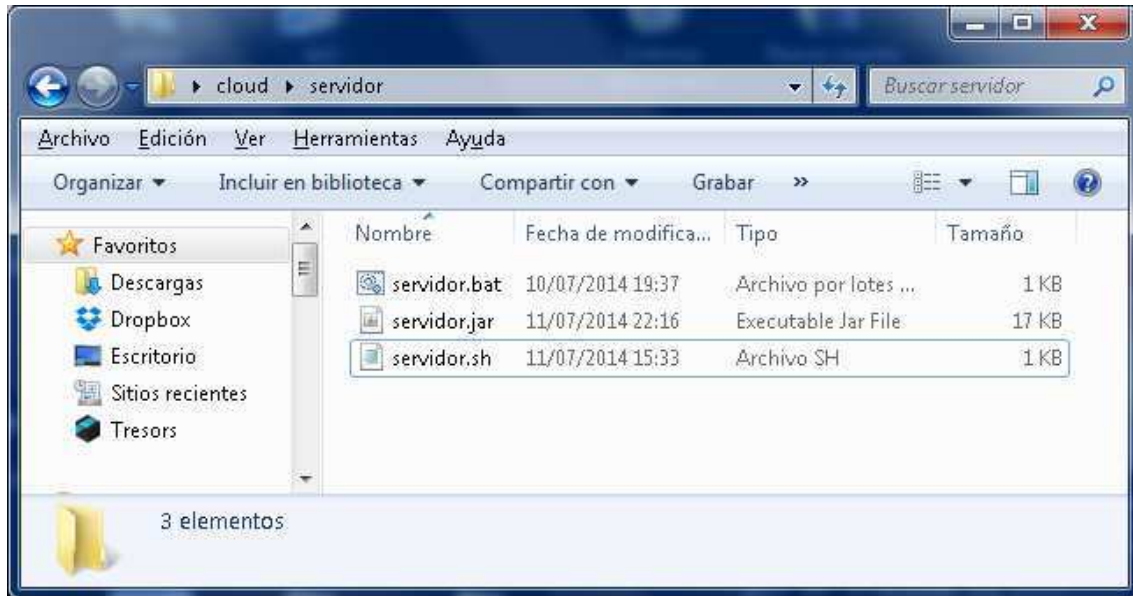
El cliente2 dispone de los archivos "f2a.txt", "f2b.txt" y "f2c.txt".

El cliente3 dispone de los archivos "f3a.txt", "f3b.txt" y "f3c.txt".

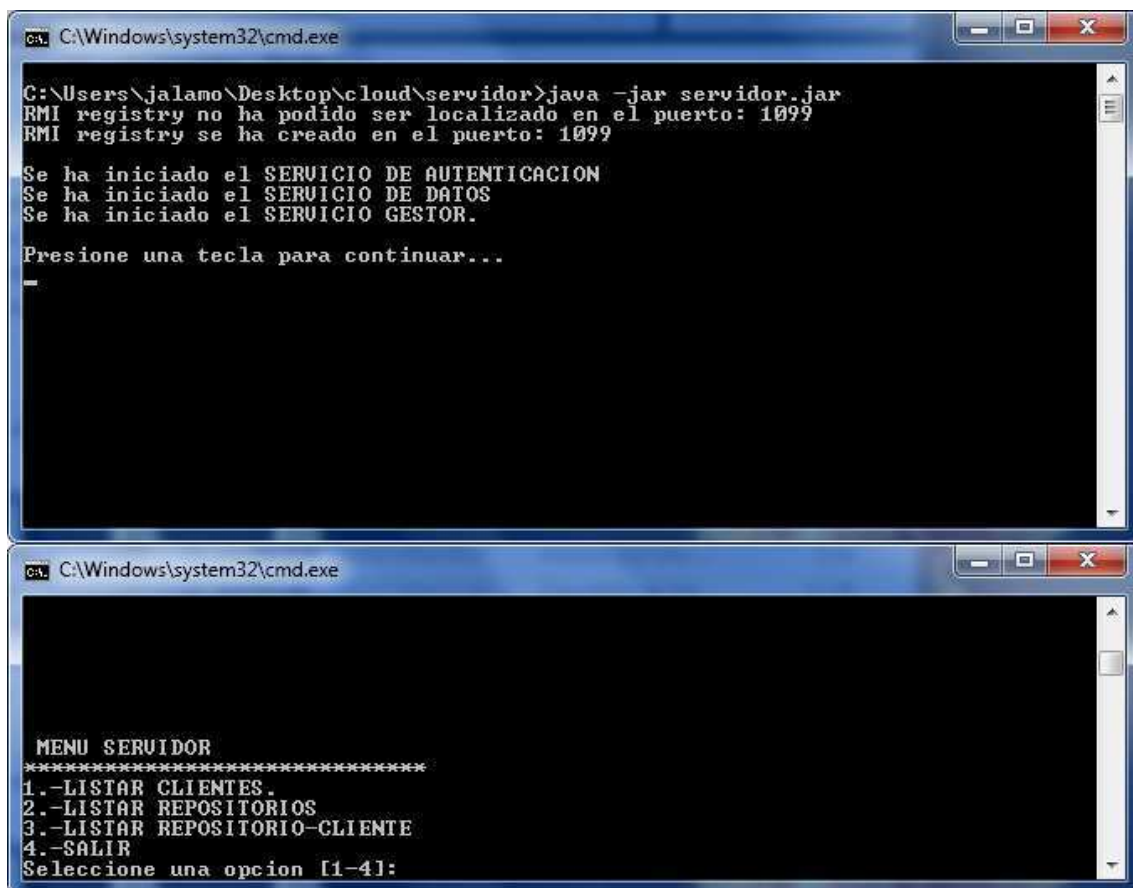


En primer lugar iniciamos el servidor, para ello, en Windows, tecleamos desde un terminal de comandos, en el directorio donde se encuentra el servidor, "servidor.bat", o

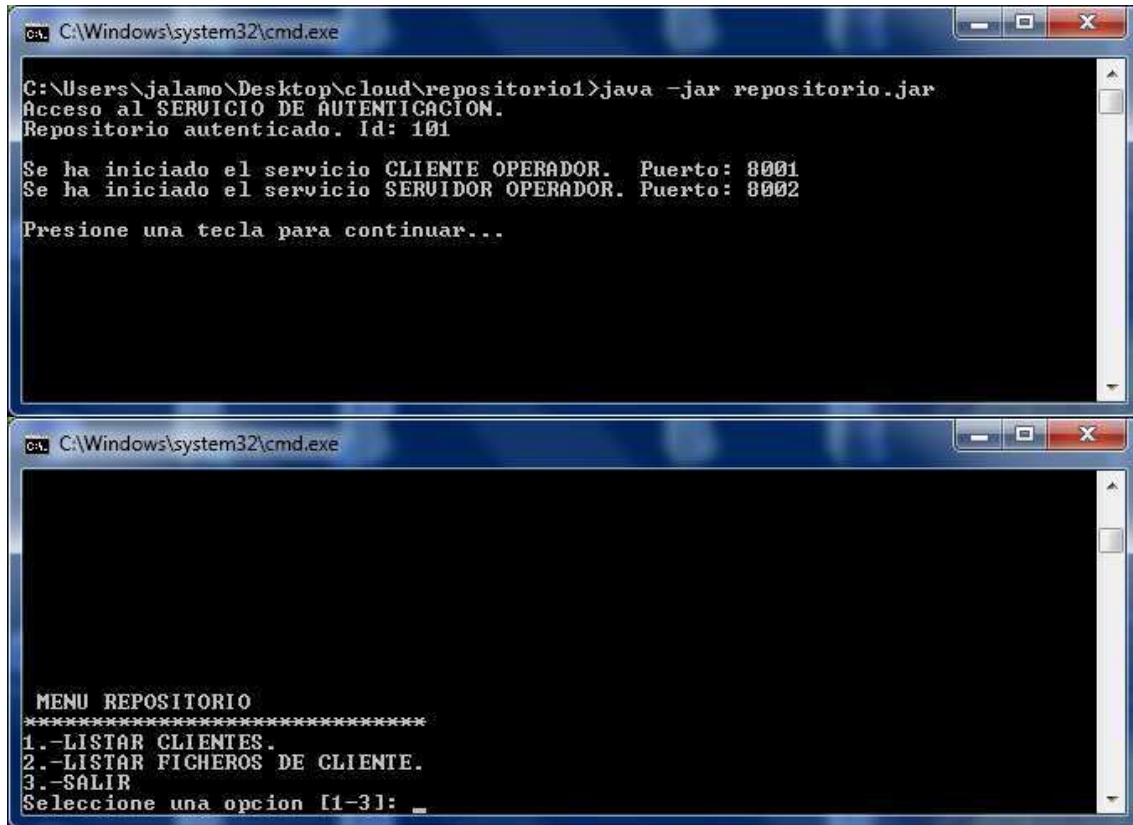
bien hacemos doble clic con el botón izquierdo del ratón sobre el archivo "servidor.bat". Si lo ejecutamos desde linux, tecleamos desde un terminal de comandos, en el directorio donde se encuentra el servidor, "servidor.sh", o bien hacemos doble clic con el botón izquierdo del ratón sobre el archivo "servidor.sh".



Una vez iniciado el servidor, lanzara los servicios de autenticación, datos y gestor, y mostrara el menú del servidor.



De igual forma, lanzamos los dos repositorios, "repositorio.bat" para windows y "repositorio.sh" para linux. Cada repositorio se inicia desde su carpeta, "repositorio1" y "repositorio". Una vez iniciado el repositorio, se autenticara en el servidor e iniciara los servicios "cliente - operador" y "servidor - operador". Por ultimo, se muestra el menú del repositorio.



The image shows two screenshots of a Windows command prompt window. The top screenshot shows the execution of the repository program, which authenticates and starts two services. The bottom screenshot shows the repository menu.

```
C:\Windows\system32\cmd.exe

C:\Users\jalamo\Desktop\cloud\repositorio1>java -jar repositorio.jar
Acceso al SERVICIO DE AUTENTICACION.
Repositorio autenticado. Id: 101

Se ha iniciado el servicio CLIENTE OPERADOR. Puerto: 8001
Se ha iniciado el servicio SERVIDOR OPERADOR. Puerto: 8002

Presione una tecla para continuar...

C:\Windows\system32\cmd.exe

MENU REPOSITORIO
*****
1.-LISTAR CLIENTES.
2.-LISTAR FICHEROS DE CLIENTE.
3.-SALIR
Seleccione una opcion [1-3]: _
```

Por ultimo, iniciamos los tres clientes, cada uno desde su carpeta, "cliente1", "cliente2" y cliente3". De igual forma que con el servidor y los repositorios, los clientes los iniciamos mediante los ficheros "cliente.bat" para windows y "cliente.sh" para linux. Una vez iniciado un cliente, pide que se introduzca un nombre de usuario por pantalla, (no pide clave, ya que no se ha implementado autenticación por contraseña). Se introducirá para cada uno de los clientes los nombres "Cliente1", "Cliente2" y "Cliente3". Seguidamente, el cliente inicia el servicio "disco cliente", muestra su identificador único y por ultimo muestra el menú de cliente.



The image shows a screenshot of a Windows command prompt window. It shows the execution of the client program, which authenticates and starts a service.

```
C:\Windows\system32\cmd.exe

C:\Users\jalamo\Desktop\cloud\cliente1>java -jar cliente.jar
Acceso al SERVICIO DE AUTENTICACION.
Acceso al SERVICIO DE GESTION.

Introduzca nombre de CLIENTE: : Cliente1

Se ha iniciado el servicio DISCO CLIENTE. Puerto: 8005

Cliente autenticado. Id: 103

Presione una tecla para continuar...
```

```
C:\Windows\system32\cmd.exe

C:\Users\jalamo\Desktop\cloud\cliente2>java -jar cliente.jar
Acceso al SERVICIO DE AUTENTICACION.
Acceso al SERVICIO DE GESTION.

Introduzca nombre de CLIENTE: : Cliente2
Se ha iniciado el servicio DISCO CLIENTE. Puerto: 8006
Cliente autenticado. Id: 104
Presione una tecla para continuar...
_
```

```
C:\Windows\system32\cmd.exe

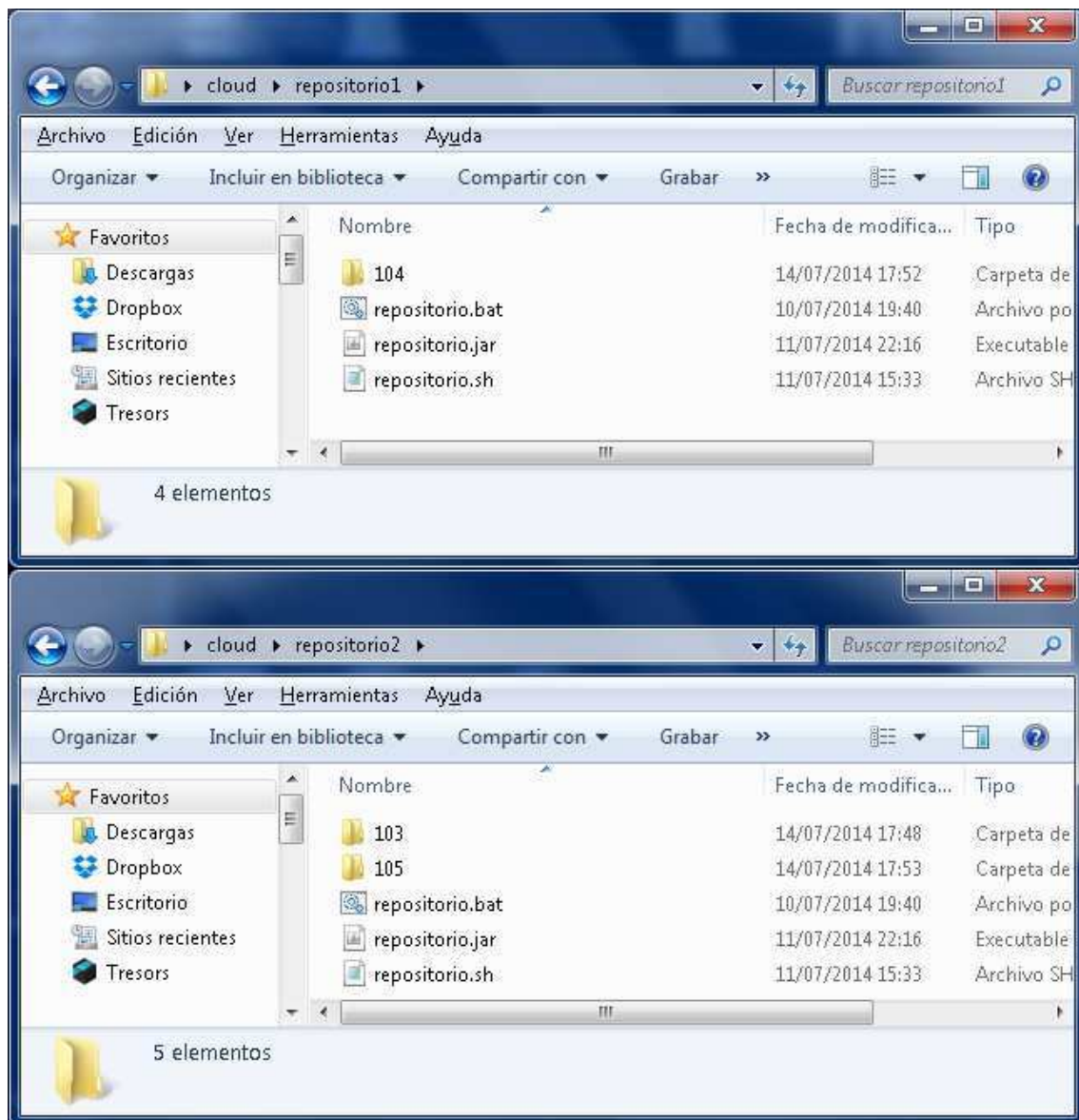
C:\Users\jalamo\Desktop\cloud\cliente3>java -jar cliente.jar
Acceso al SERVICIO DE AUTENTICACION.
Acceso al SERVICIO DE GESTION.

Introduzca nombre de CLIENTE: : Cliente3
Se ha iniciado el servicio DISCO CLIENTE. Puerto: 8007
Cliente autenticado. Id: 105
Presione una tecla para continuar...
```

```
C:\Windows\system32\cmd.exe

MENU CLIENTE
*****
1.-SUBIR FICHERO.
2.-BAJAR FICHERO
3.-BORRAR FICHERO
4.-LISTAR FICHEROS
5.-LISTAR CLIENTES
6.-SALIR
Seleccione una opcion [1-6]: _
```

Cada vez que se inicia un cliente y se autentica correctamente, se le asigna un repositorio, el cual crea automáticamente una carpeta para el cliente con el nombre del identificador único de cliente. Por tanto, podemos observar como dentro de las carpetas de cada repositorio se han creado las carpetas correspondientes a los tres clientes iniciados.

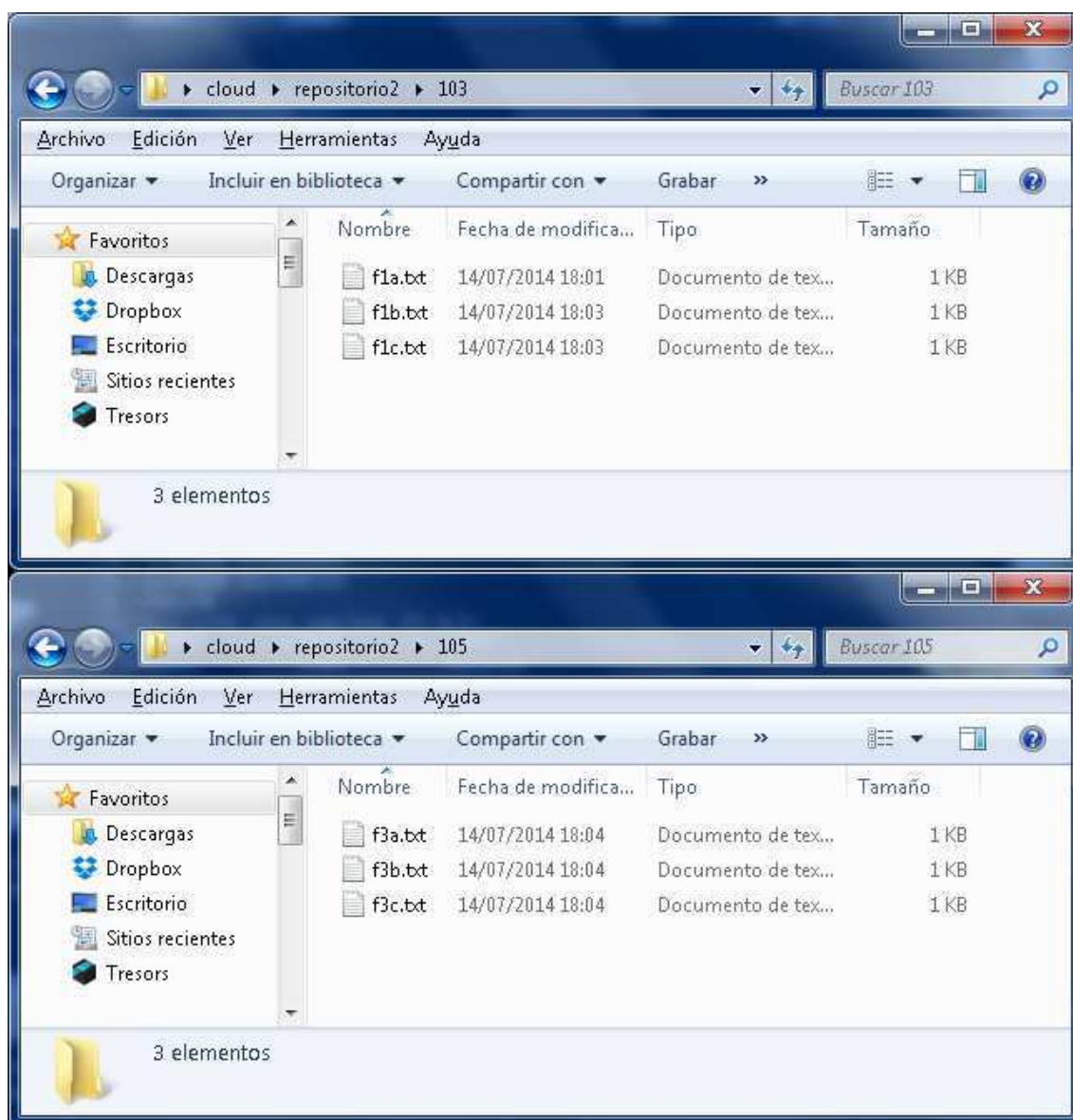


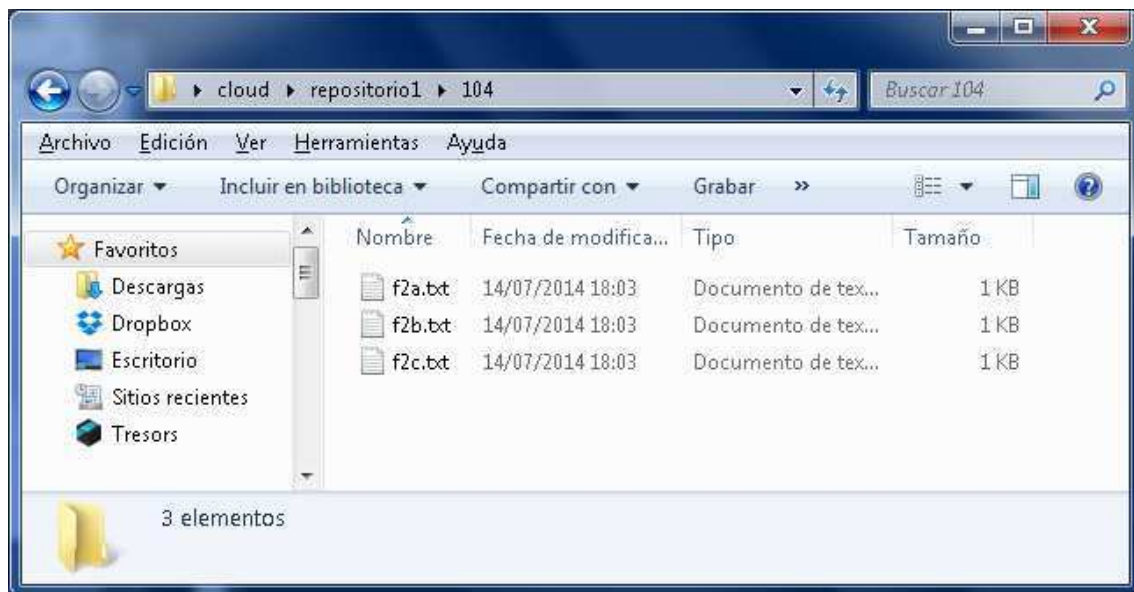
El cliente 1 subirá sus tres archivos al repositorio que le ha sido asignado.



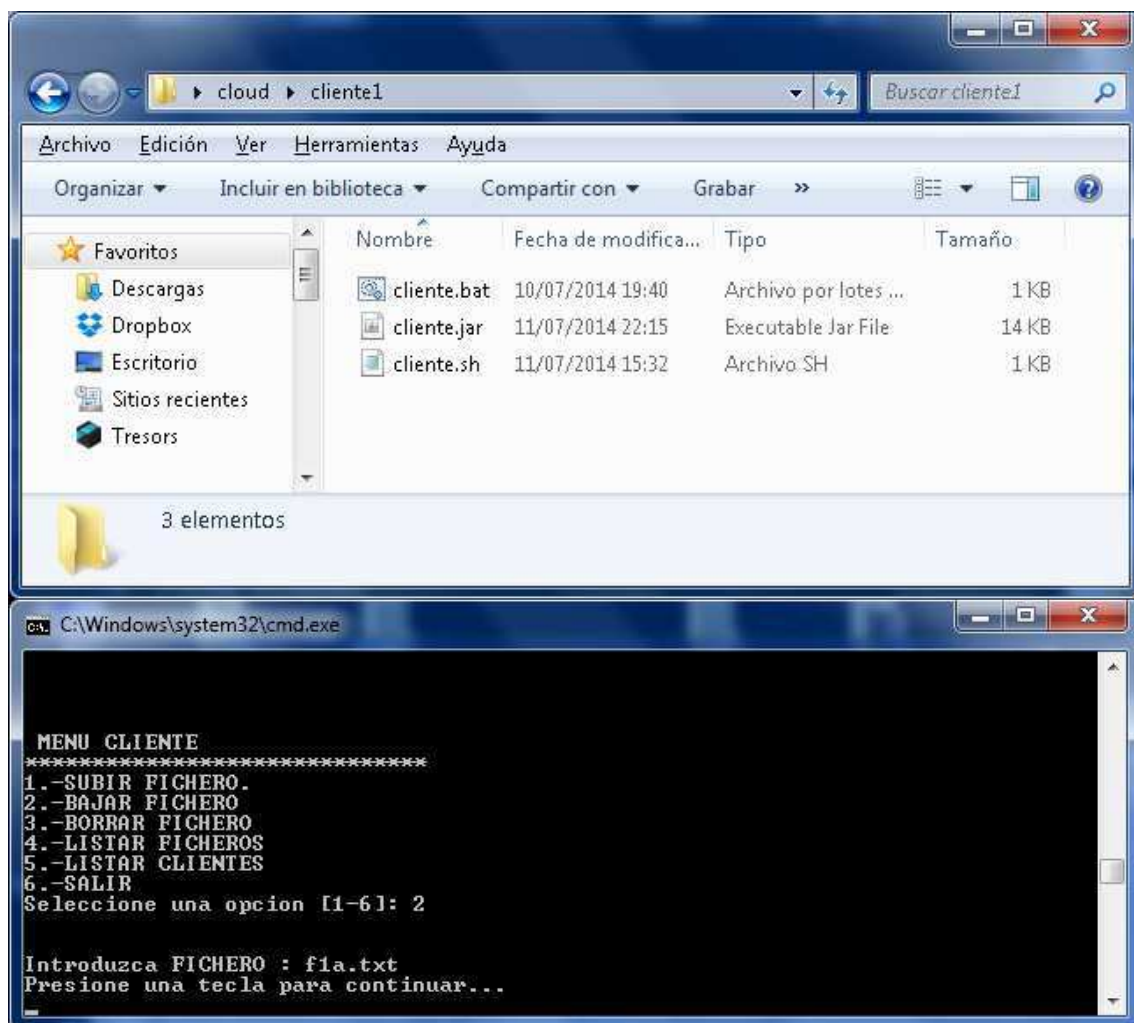
Una vez subidos los tres archivos, comprobamos que se ha grabado en la correspondiente carpeta del repositorio. Hacemos lo mismo con los demás clientes,

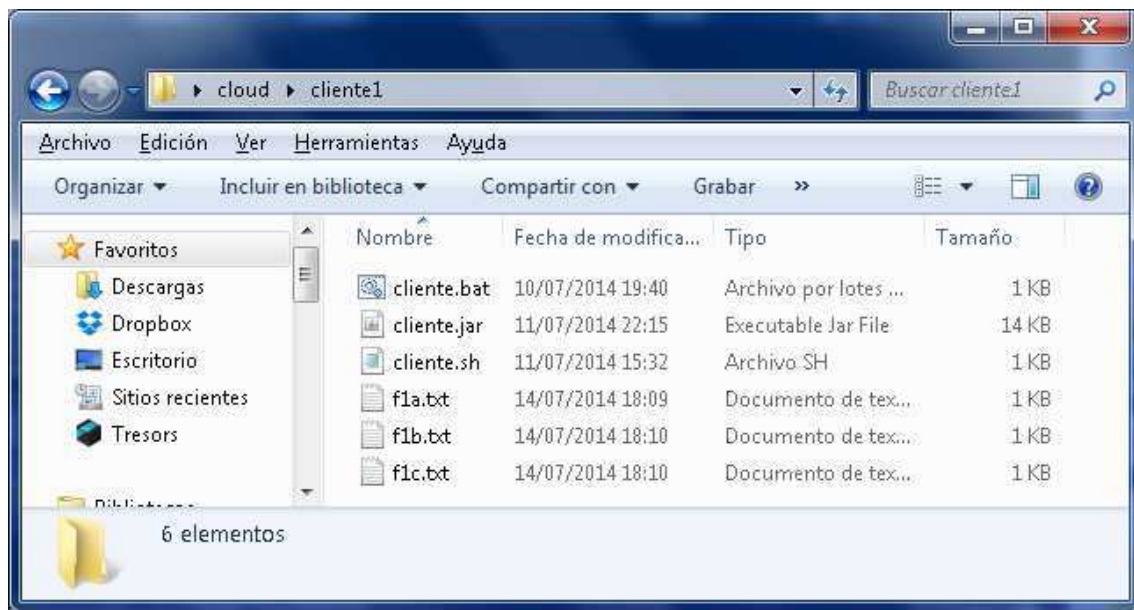
comprobando que todos los archivos se han grabado en sus correspondientes carpetas de sus repositorios.



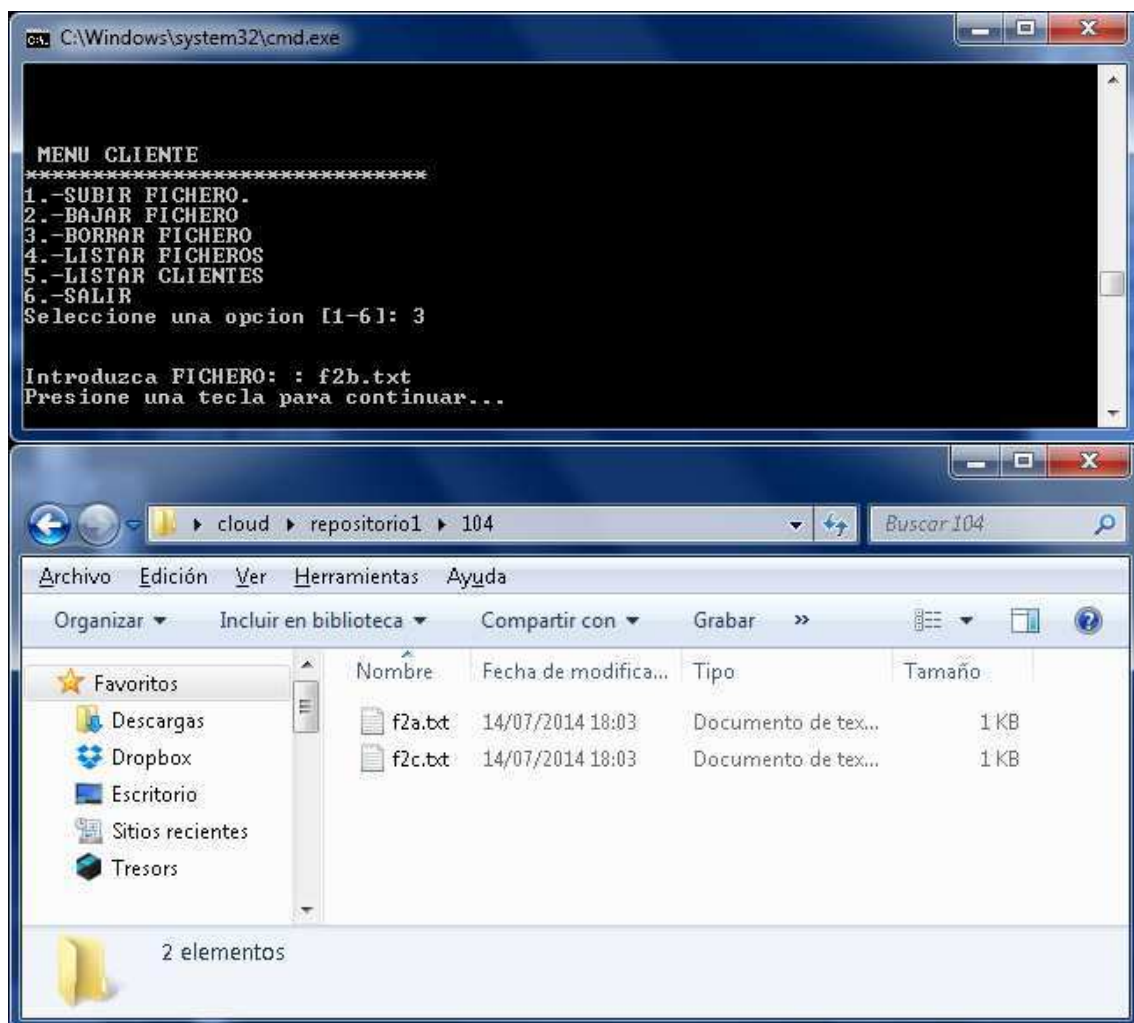


Para comprobar la descarga de archivos, borramos los archivos de la carpeta del primer cliente, y procedemos a descargar los tres archivos del repositorio, comprobando después que estos han sido descargados a la carpeta del cliente.





Seguidamente procedemos a borrar un archivo con el segundo cliente, en concreto, borraremos el fichero "f2b.txt", y comprobamos que se elimina de la carpeta del repositorio que lo almacena.



Si hacemos un listado de los ficheros del segundo cliente, podemos comprobar que efectivamente solo existen los ficheros "f2a.txt" y "f2c.txt".



```
C:\Windows\system32\cmd.exe


MENU CLIENTE
*****
1.-SUBIR FICHERO.
2.-BAJAR FICHERO
3.-BORRAR FICHERO
4.-LISTAR FICHEROS
5.-LISTAR CLIENTES
6.-SALIR
Seleccione una opcion [1-6]: 4

Listado de FICHEROS remotos.
-----
f2a.txt          28 bytes.
f2c.txt          7 bytes.

Hay un total de 2 FICHEROS remotos.

Presione una tecla para continuar...
```

Por ultimo, hacemos un listado de los clientes conectados al sistema.



```
C:\Windows\system32\cmd.exe

MENU CLIENTE
*****
1.-SUBIR FICHERO.
2.-BAJAR FICHERO
3.-BORRAR FICHERO
4.-LISTAR FICHEROS
5.-LISTAR CLIENTES
6.-SALIR
Seleccione una opcion [1-6]: 5

Listado de CLIENTES conectados al sistema.
-----
ID: 103  NOMBRE: Cliente1
ID: 104  NOMBRE: Cliente2
ID: 105  NOMBRE: Cliente3

Hay un total de 3 CLIENTES conectados al sistema.

Presione una tecla para continuar...
```

A continuación, comprobamos el funcionamiento de las opciones de los repositorios. Se hace, con ambos repositorios un listado de los clientes que tienen asignados. Comprobamos que un repositorio tiene asignado un cliente y el otro repositorio dos clientes.

```
C:\Windows\system32\cmd.exe

MENU REPOSITORIO
*****
1.-LISTAR CLIENTES.
2.-LISTAR FICHEROS DE CLIENTE.
3.-SALIR
Seleccione una opcion [1-3]: 1

Lista de CLIENTES asociados al REPOSITORIO 101.

ID: 104 NOMBRE: Cliente2
Presione una tecla para continuar...

C:\Windows\system32\cmd.exe

MENU REPOSITORIO
*****
1.-LISTAR CLIENTES.
2.-LISTAR FICHEROS DE CLIENTE.
3.-SALIR
Seleccione una opcion [1-3]: 1

Lista de CLIENTES asociados al REPOSITORIO 102.

ID: 105 NOMBRE: Cliente3
ID: 103 NOMBRE: Cliente1
Presione una tecla para continuar...
```

Seguidamente hacemos un listado de los ficheros de clientes que tiene almacenados cada repositorio. Comprobamos que un repositorio tiene un total de dos ficheros, los correspondientes al cliente con identificador 104 y el otro repositorio tiene un total de 6 ficheros, los correspondientes a los clientes con identificadores 105 y 103.

```
C:\Windows\system32\cmd.exe

MENU REPOSITORIO
*****
1.-LISTAR CLIENTES.
2.-LISTAR FICHEROS DE CLIENTE.
3.-SALIR
Seleccione una opcion [1-3]: 2

Lista de FICHEROS de CLIENTES.

CLIENTE - ID: 104      NOMBRE: Cliente2      FICHERO: f2a.txt      28 byte
s.
CLIENTE - ID: 104      NOMBRE: Cliente2      FICHERO: f2c.txt      7 bytes
.

Hay un total de 2 Ficheros.
Presione una tecla para continuar...
```



```
Ca. C:\Windows\system32\cmd.exe

MENU REPOSITORIO
*****
1.-LISTAR CLIENTES.
2.-LISTAR FICHEROS DE CLIENTE.
3.-SALIR
Seleccione una opcion [1-3]: 2

Lista de FICHEROS de CLIENTES.

CLIENTE - ID: 105      NOMBRE: Cliente3      FICHERO: f3a.txt      28 byte
s.
CLIENTE - ID: 105      NOMBRE: Cliente3      FICHERO: f3b.txt      7 bytes
CLIENTE - ID: 105      NOMBRE: Cliente3      FICHERO: f3c.txt      7 bytes
CLIENTE - ID: 103      NOMBRE: Cliente1      FICHERO: f1a.txt      28 byte
s.
CLIENTE - ID: 103      NOMBRE: Cliente1      FICHERO: f1b.txt      7 bytes
CLIENTE - ID: 103      NOMBRE: Cliente1      FICHERO: f1c.txt      7 bytes
.

Hay un total de 6 Ficheros.

Presione una tecla para continuar...
```

Seguidamente comprobamos el funcionamiento del menú del servidor.
En primer lugar, hacemos un listado de los clientes que hay conectados al sistema.

```
Ca. C:\Windows\system32\cmd.exe

MENU SERVIDOR
*****
1.-LISTAR CLIENTES.
2.-LISTAR REPOSITORIOS
3.-LISTAR REPOSITORIO-CLIENTE
4.-SALIR
Seleccione una opcion [1-4]: 1

LISTA DE CLIENTES

Hay 3 CLIENTES conectados
Id: 103 Nombre: Cliente1
Id: 104 Nombre: Cliente2
Id: 105 Nombre: Cliente3

Presione una tecla para continuar...
=
```

Hacemos un listado de los repositorios que hay conectados al sistema.

```
Ca. C:\Windows\system32\cmd.exe

MENU SERVIDOR
*****
1.-LISTAR CLIENTES.
2.-LISTAR REPOSITARIOS
3.-LISTAR REPOSITARIO-CLIENTE
4.-SALIR
Seleccione una opcion [1-4]: 2

LISTA DE REPOSITARIOS

Hay 2 REPOSITARIOS conectados
Id: 102 Nombre: repositorio102
Id: 101 Nombre: repositorio101
Presione una tecla para continuar...
```

Por ultimo, listamos las asociaciones entre clientes y repositorios, comprobando que al repositorio con identificador 101 se le ha asignado al cliente con identificador 104 y que al repositorio con identificador 102 se le han asignado dos clientes, los clientes con identificadores 103 y 105.

```
Ca. C:\Windows\system32\cmd.exe

MENU SERVIDOR
*****
1.-LISTAR CLIENTES.
2.-LISTAR REPOSITARIOS
3.-LISTAR REPOSITARIO-CLIENTE
4.-SALIR
Seleccione una opcion [1-4]: 3

LISTA DE CLIENTES - REPOSITARIOS

Hay 3 CLIENTES - REPOSITARIOS asociados
CLIENTE - ID: 103      NOMBRE: Cliente1      REPOSITARIO - ID: 102  NOMBRE:
repositorio102
CLIENTE - ID: 104      NOMBRE: Cliente2      REPOSITARIO - ID: 101  NOMBRE:
repositorio101
CLIENTE - ID: 105      NOMBRE: Cliente3      REPOSITARIO - ID: 102  NOMBRE:
repositorio102
Presione una tecla para continuar...
```

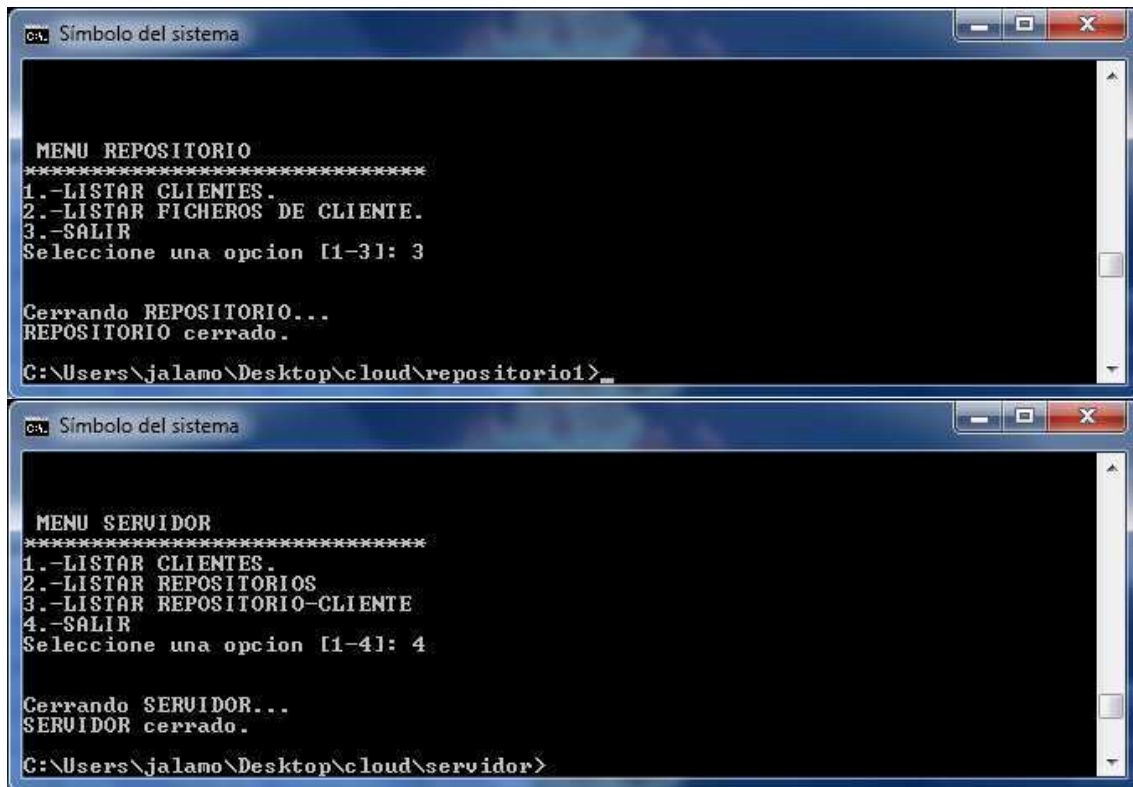
Por ultimo, cerramos los tres clientes, los dos repositorios y el servidor.

```
Ca. Símbolo del sistema

MENU CLIENTE
*****
1.-SUBIR FICHERO.
2.-BAJAR FICHERO
3.-BORRAR FICHERO
4.-LISTAR FICHEROS
5.-LISTAR CLIENTES
6.-SALIR
Seleccione una opcion [1-6]: 6

Cerrando CLIENTE...
CLIENTE cerrado.

C:\Users\jalamo\Desktop\cloud\cliente1>
```



```
ca. Símbolo del sistema
MENU REPOSITORIO
*****
1.-LISTAR CLIENTES.
2.-LISTAR FICHEROS DE CLIENTE.
3.-SALIR
Seleccione una opcion [1-3]: 3

Cerrando REPOSITORIO...
REPOSITORIO cerrado.
C:\Users\jalamo\Desktop\cloud\repositorio1>

ca. Símbolo del sistema
MENU SERVIDOR
*****
1.-LISTAR CLIENTES.
2.-LISTAR REPOSITORIOS
3.-LISTAR REPOSITORIO-CLIENTE
4.-SALIR
Seleccione una opcion [1-4]: 4

Cerrando SERVIDOR...
SERVIDOR cerrado.
C:\Users\jalamo\Desktop\cloud\servidor>
```

CONCLUSIONES Y MEJORAS.

Se ha implementado un sistema básico de almacenamiento de ficheros en la nube, de manera que se ha diseñado de tal forma que el cliente no sea consciente de la distribución del sistema, solo debe conectar con el servidor, pero no sabe como se gestiona el tratamiento de los ficheros. Otro punto a destacar es la implementación del sistema de repositorios de manera que la subida y bajada real de los ficheros se haga entre el cliente y los repositorios, dejando al servidor como mediador o gestor de dichas acciones, evitando de esta forma la sobrecarga del servidor. De esta forma, para poder aumentar el espacio de almacenamiento en la nube o no cargar demasiado un repositorio, simplemente se ha de lanzar otro nuevo repositorio en otra máquina y registrarlo en el servidor. De esta forma, si aumentan los clientes, podrán aumentarse el número de repositorios y el sistema seguirá siendo efectivo.

No se ha implementado ningún tipo de seguridad ni autenticación de los usuarios para no complicar en exceso el programa, puesto que la práctica trata de la programación y funcionamiento de un sistema distribuido.

Por tanto, como mejoras, se podría comenzar por implementar un sistema de seguridad comenzando por la autenticación de usuarios mediante claves de acceso al sistema, así como cifrado de datos.

Otra mejora sería que el usuario pudiese crear un árbol de directorios a partir de su carpeta privada en el repositorio, pudiendo de esta forma ordenar y poder mantener de una forma más clara sus archivos en la nube.

También sería interesante ofrecer al usuario la opción de poder crear una carpeta en la nube cuyo contenido (ficheros y subcarpetas) estén cifradas mediante una clave de cifrado que no se almacene en el servidor ni en ningún otro lado del sistema. Aunque, esta opción, haría que la responsabilidad del cifrado recaiga sobre el cliente, es decir, mediante esta opción, el cliente cifraría el archivo o archivos a subir a la nube y los mandaría ya cifrados mediante una clave que solo y solo el usuario sabe y que no se

almacena en ninguna parte del sistema. El repositorio recibiría archivos cifrados. Por otro lado, esta opción plantea el peligro de que el usuario pierda o no recuerde la clave de cifrado, ya que al no estar guardada, no podría recuperarse y por tanto, aunque el cliente pudiese recuperar los archivos, estos no podrían ser descifrados.

Otra posible mejora, sería que el cliente pudiese obtener una dirección url del archivo que desee y poder enviarla a otro cliente para que este tenga acceso al fichero, sería una forma de compartir ficheros. También se podría implementar una carpeta especial dentro de la carpeta del usuario en el repositorio, que fuese una carpeta publica, de manera que todos los archivos del usuario que se almacenen en esta carpeta publica sean de dominio publico y cualquier cliente del sistema pudiese acceder a el.

Otra mejora a implementar sería que las listas de datos para gestionar el sistema se almacenasen en disco, de manera que si el servidor, los repositorios o los clientes fallan o se desconectan de forma accidental o intencionada, cuando vuelvan a iniciarse puedan saber el estado del sistema y recuperar información de los clientes, repositorios y las asociaciones entre clientes y repositorios, de esta forma, un cliente podría conectarse, desconectarse y volver a conectar y seguir accediendo a sus ficheros.