75.42 Taller de Programación I

TP N°5: Archivos Ubicuos

Grupo 04

"Documentación Técnica"

Índice

1. Requerimientos de software **1**

2. Descripción general **1**

3. Aplicación *Servidor* **1**
3.1. Clases 1
3.2. Diagramas UML 2
3.3. Descripción de archivos protocolos 3

4. Aplicación *Cliente* **3**
4.1. Clases 4
4.2. Diagramas UML 4
4.3. Descripción de archivos protocolos 4

5. Aplicación *Monitor* **4**
5.1. Clases 5
5.2. Diagramas UML 5
5.3. Descripción de archivos protocolos 5

6. Programas intermedios y de prueba **5**

7. Código Fuente **5**

1. Requerimientos de software

Se listan a continuación los distintos requerimientos mínimos y necesarios para poder compilar, desarrollar, probar y depurar las aplicaciones que conforman al proyecto:

- *Sistemas operativos*: GNU/Linux (x86 y x86-64, distribuciones Linux basadas en RPM y DEB);
- *Controlador de versiones*[1]: GIT (`http://git-scm.com/`);
- *Compilador*: g++ (`http://gcc.gnu.org/`);
- *Herramientas*: Make (`http://www.gnu.org/software/make/`).

2. Descripción general

El proyecto se encuentra dividido en tres aplicaciones principales: *cliente*, *servidor* y *monitor*. El corazón central del servicio se encuentra establecido sobre el servidor, ya que las demás aplicaciones tienen como tarea mantenerse en contacto con este. Tal comunicación se realiza a travéz de sockets, permitiéndose así la ejecución remota de los programas, instanciados en distintos equipos.

Para lograr tal fin, se han identificado ciertos módulos que dotarán de distintos tipos de funcionalidades a nuestras aplicaciones. Además de ello, y no menos importante, se establecerá la existencia de un protocolo conocido por los tres entes, de manera de poder entenderse entre si. Cabe señalar que la aplicación monitor no es capaz de interactuar con la aplicación cliente, sino que simplemente todas interactuan con el servidor.

3. Aplicación *Servidor*

Al ser la parte central y motor del servicio, el servidor deberá constar de un conjunto de módulos los cuales permitirán dividir las distintas ocupaciones y eventos que deben atenderse en este.

Como avance de lo que será la explicación detallada del propósito a cumplir por cada clase, veamos los tipos de eventos y como son manejados a grandes razgos por el servidor.

En primer lugar, el servidor atenderá conexiones entrantes. Estas últimas son derivadas a un ente que se encargará de mantener una interacción con el cliente que se encuentra al otro lado del canal de comunicación. El primer paso que se realiza con este usuario es el inicio de sesión. Esto le permitirá identificarse y ser derivado de acuerdo a la credencial (nombre de usuario) presentada.

Cada usuario se encuentra vinculado a un ente que denomiaremos *carpeta*. Al producirse una conexión entrante, cada usuario es derivado a la carpeta que le corresponde, agrupándose así en cada uno de los directorios las conexiones vinculadas a un mismo nombre de usuario.

Llegado a este paso, es decir, una vez que una conexión es derivada a la carpeta correspondiente, se inicia la sincronización con el host de dicha comunicación. Esto se logra a través del intercamboo de mensajes los cuales llegan a cada ente que representa una conexion, y los cuales son todos depositados en un ente *receptor*.

Cada carpeta posee un módulo *sincronizador* el cual se ocupa de tomar uno a uno los mensajes provenientes del receptor, los procesa y da la orden de realizar una acción en base a esto último. Entre estas acciones se encuentran: recibir un archivo, recibir una notificación de archivo eliminado, recibir partes correspondientes a un archivo modificado, etc.

Al llegar un mensaje que solicita agregar un archivo, modificarlo o eliminarlo, se deriva con dicha notificación en el módulo *manejador de archivos*, quien se encarga de lidiar con los archivos físicos.

3.1. Clases

Pasaremos ahora a detallar las clases que conforman el conjunto de módulos de la aplicación servidor. Se recomienda al lector que a medida que avance en cada descripción, visualice en los diagramas de clases mostrados posteriormente ya que esto sumará en el entendimiento de como es que se relacionan entre sí los entes.

- *Servidor*: es la encargada de estar a la escucha de nuevas conexiones y de derivarlas a ConexionCliente;

[1] Este requerimiento es de caracter opcional ya que solo es necesario en caso de desear clonar el proyecto desde el repositorio del grupo.

- *ConexionCliente*: primeramente se ocupa de que el usuario se identifique mediante el inicio de sesión. De ser válidado correctamente este último, se deriva al objeto de ConexionCliente pertenenciente a dicha comunicación hacia el AdministradorDeClientes. En caso de no ser válido el inicio de sesión, la conexión se auto destruye;
- *AdministradorDeClientes*: su tarea se limita a derivar a los clientes a sus respectivas carpetas lógicas, como así también es quien recibe las ordenes de dar de baja a un cliente de una carpeta y destruirlo;
- *Validador*: es quien se encarga de validar los datos que envía el usuario para iniciar sesión en su carpeta;
- *Carpeta*: los objetos de esta clase son administrados por AdministradorDeClientes. Estas representan la carpeta a nivel lógico de cada usuario. El Administrador se encarga de crear una sola de estas carpetas para un grupo de conexiones que se han loggeado con el mismo nombre de usuario. Cada carpeta posee asociado un conjunto de modulos con los cuales realizará emisiones y recepciones hacia los distintos objetos ConexionCliente de la carpeta, sincronizará y procesará mensajes entrantes y salientes y realizará el manejo de los archivos fisicos de la carpeta sobre el servidor. Las clases que representan estos módulos son: *Actualizador*, *Receptor*, *Emisor*, *Sincronizador* y *ManejadorDeArchivos*.

3.2. Diagramas UML

En la *Figura 1* se muestra un diagrama de clases del corazón del servidor. En este se pueden notar las relaciones que hacen posible la derivación de un cliente hacia una carpeta una vez iniciada la sesión.

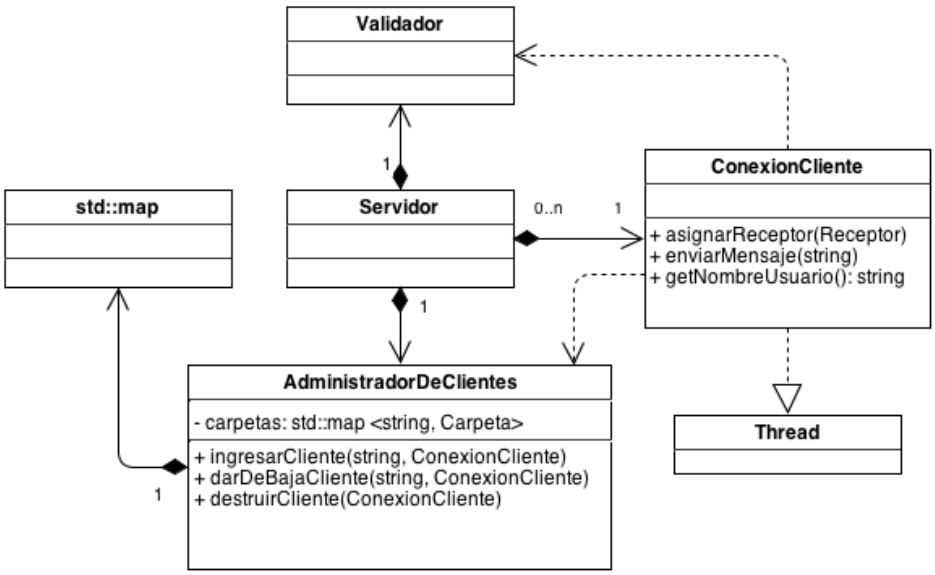

Figura 1: *Diagrama de clases principal del servidor.*

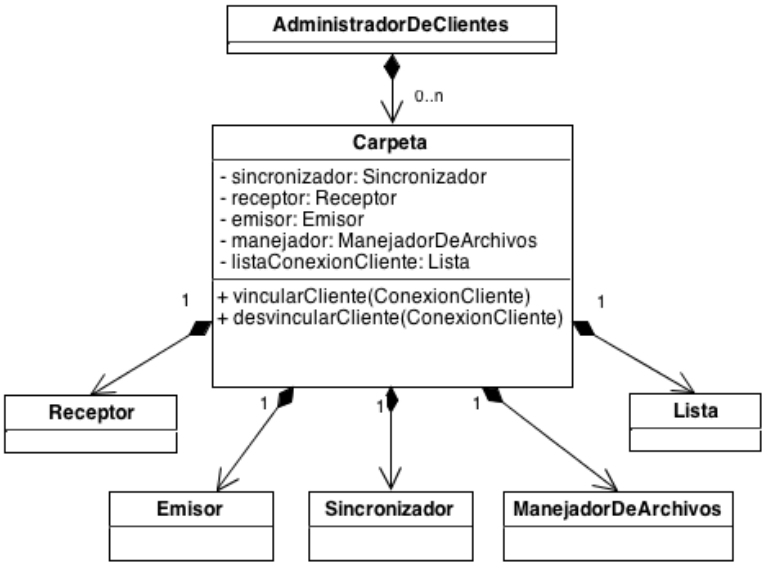

Figura 2: *Diagrama que muestra la jerarquía de clases principales.*

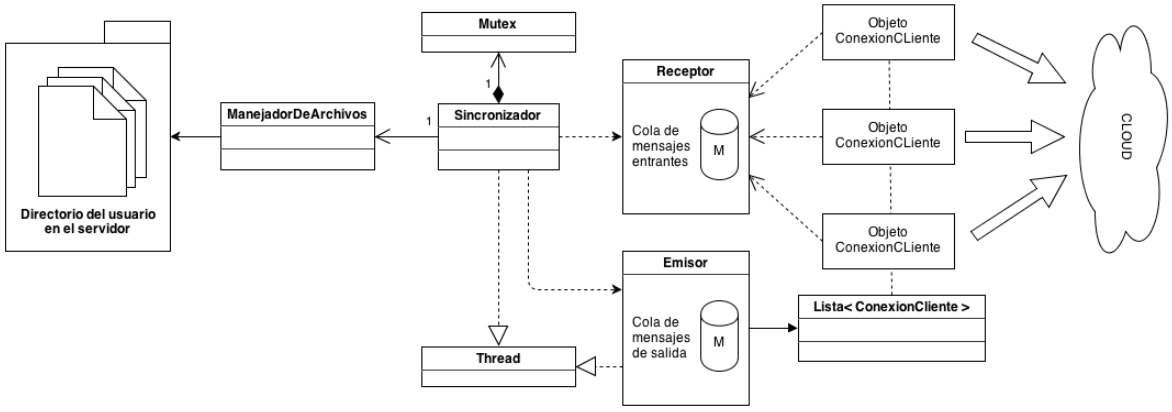

Figura 3: *Diagrama de representación de una Carpeta con clientes conectados.*

3.3. Descripción de archivos protocolos

[Colocar texto aquí]

4. Aplicación *Cliente*

[Colocar texto aquí (descripción general)]

4.1. Clases

[Colocar texto aquí]

4.2. Diagramas UML

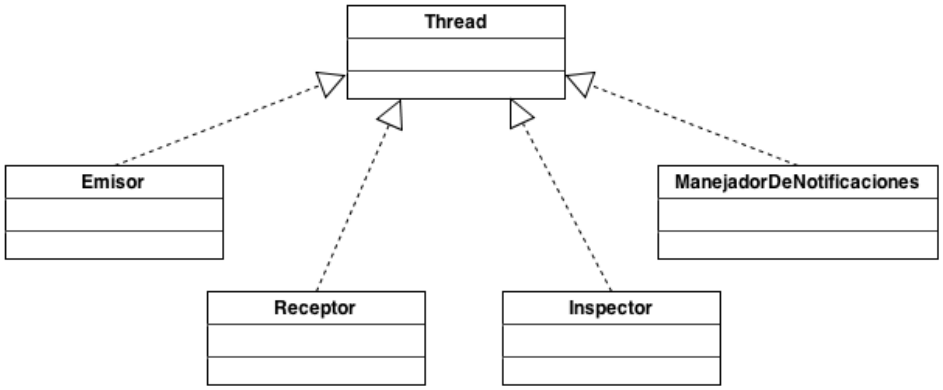

Figura 4: *Diagrama de clases que indica aquellas que son un Thread.*

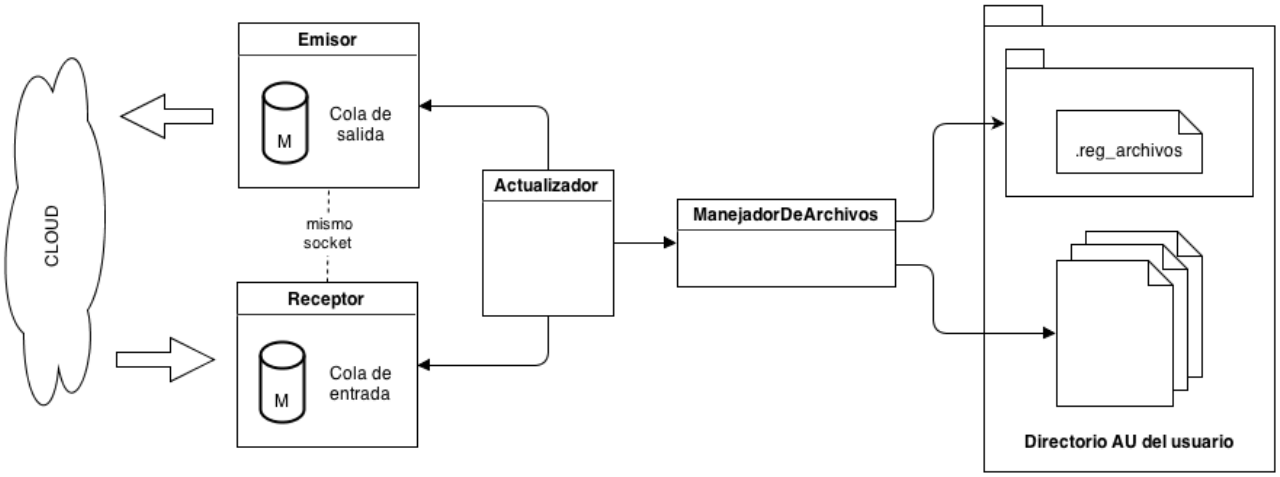

Figura 5: *Diagrama de clases que muestra de que forma se relacionan en el proceso de actualización inicial.*

4.3. Descripción de archivos protocolos

[Colocar texto aquí]

5. Aplicación *Monitor*

[Colocar texto aquí (descripción general)]

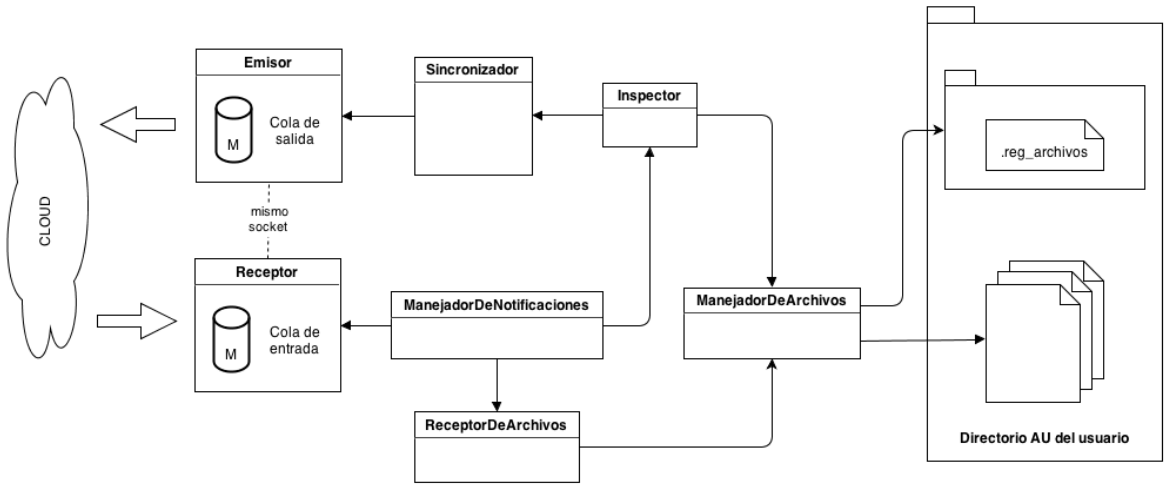

Figura 6: *Diagrama de clases que muestra de que forma se relacionan en el proceso de sincronizacion.*

5.1. Clases

[Colocar texto aquí]

5.2. Diagramas UML

5.3. Descripción de archivos protocolos

[Colocar texto aquí]

6. Programas intermedios y de prueba

[Colocar texto aquí]

7. Código Fuente

[Colocar texto aquí]