75.42 Taller de Programación I

# TP N°5: Archivos Ubicuos

*Grupo 04*

*"Manual de Proyecto"*

# Índice

# 1. Ingregrantes

El grupo se encuentra conformado por los siguientes alumnos:


- Belén Beltran (91718) - *belubeltran@gmail.com*
- Fiona Gonzalez Lisella (91454) - *dynamo89@gmail.com*
- Federico Martín Rossi (92086) - *federicomrossi@gmail.com*


# 2. Enunciado

Se propone realizar un servicio similar al del conocido producto *DropBox*[1], que se denominará *Archivos Ubicuos (AU)*. Este servicio debe permitir que los distintos usuarios tengan, cada uno en su computadora, un directorio (que a los fines de la aplicación se denominará "directorio de AU") cuyo contenido debe sincronizarse automáticamente con el directorio de AU del usuario enotras computadoras.

## 2.1. Descripción general

El servicio estará compuesto por las siguientes aplicaciones:

- **Aplicación cliente**: corre en las computadoras de los usuarios. Debe contar con la posibilidad de configurar el acceso al servicio de AU (usuario y contraseña) así como también el directorio que se sincronizará. Dado que para este desarrollo no se contará con un host fijo de contacto para el servidor, se permitirá además configurar el IP del servidor al que el cliente debe conectarse.De acuerdo a las decisiones de arquitectura que realice el grupo, también se deberán configurar números de puertos, intervalo de polling, etc. La aplicación deberá contar con una pequeña interfaz gráfica para efectuar estas configuraciones, y deberá soportar su persistencia en disco;
- **Aplicación servidor**: corre en una computadora determinada. Se encarga de mantener el registro de usuarios, de computadoras de dichos usuarios con sus estados de conexión, y de archivos. Además, almacena una copia de los mismos, para que cuando se conecta un cliente pueda sincronizarse. Esta aplicación debe permitir la realización de configuraciones (tales como puertos, ruta donde almacenar archivos) en un archivo del tipo properties que se ubicará en un lugar previamente conocido;
- **Aplicación monitor**: corre en la misma computadora que la aplicación servidor y permite realizar las mismas configuraciones que se encuentran en el archivo de properties del servidor en forma gráfica, y generar un gráfico de cantidad de bytes almacenados en función del tiempo (actualizado cada un período a determinar por el grupo). Además, contiene la funcionalidad necesaria para gestionar altas, bajas y modificaciones de los usuarios que acceden al servicio.

## 2.2. Restricciones

Cada usuario podrá tener un solo directorio de AU en una computadora determinada. La ruta a este directorio, así como las credenciales de autenticación en el servicio, se almacenarán según criterio del grupo.

Las aplicaciones deben tener un mecanismo de logging (*log4cpp*, *syslog* o similar) que permita conocer los eventos más relevantes que han sucedido para una eventual sesión de debugging o una auditoría sobre las acciones realizadas.

La detección de cambios respecto de la copia almacenada en el servidor es una tarea compleja, razón por la cual, el grupo podrá realizar propuestas al respecto, aunque como condición inicial se espera que se base en un algoritmo de hashing conocido (tal como *MD5* o *SHA* - DropBox emplea *SHA256* como parte de su técnica de almacenamiento), con posibles optimizaciones en base a nombre, fecha, tamaño, etc.

Para esta versión del servicio no se requieren funcionalidades de carpetas compartidas, encriptación de los archivos propiamente dichos, transferencia peertopeer entre clientes, ni posibilidad de ver historial de archivos. Por otro lado, si se requiere el particionamiento de archivos en bloques, además de que debe considerarse que si un cliente se desconecta en medio de la transferencia de un archivo, el resultado debe ser consistente.

### 2.3. Características opcionales

Sólo se requiere sincronizar el usuario actualmente logueado en la computadora (y se puede asumir que será solo uno). No se requiere almacenar subdirectorios en forma recursiva (puede asumirse que todos los archivos se encontrarán a nivel del directorio de au), aunque es un agregado opcional que puede implementarse (podría utilizarse el path completo como nombre del archivo en las comunicaciones e índices).

El grupo deberá proponer el modo de ejecución de la aplicación cliente. En caso de decidir opcionalmente la generación de un daemon (para por ejemplo iniciar automáticamente el cliente junto con el sistema), debe darse la opción alternativa de ejecución manual a fines de facilitar que los ayudantes correctores eviten alterar las configuraciones de sus computadoras de prueba.

## 3. División de tareas

El proyecto se dividió en tres partes: Interfaz, seguridad y modelo de sincronización. La división de tareas se realizó de la siguiente forma:

- Fiona: Encargada de interfaz.
- Federico: Encargado de modelo de sincronización.
- Belén: Encargada de seguridad y modelo de sincronización.

## 4. Evolución del proyecto

### 4.1. Inicio

En los inicios del proyecto, se realizó un diagrama PERT tentativo que mostraba la secuencia posible de pasos para realizar el proyecto. A continuación se ve la figura de este: FIGURA Al poco tiempo, este diagrama fue descartado, ya que era demasiado lineal y no presentaba suficientes etapas de desarrollo en paralelo que permitieran alcanzar el objetivo del proyecto en 8 semanas. Se debió optar por un esquema de solución con más ramas en paralelo y que considere realizar la interfaz gráfica al mismo tiempo que el modelo.

### 4.2. Primera Etapa

Se usó como esqueleto base para el proyecto la aplicación cliente-servidor generada en el trabajo práctico Nº4.

Como primeros pasos para la resolución del problema se obtuvo la lista de archivos presentes en el directorio del cliente. Se logró obtener una lista que refleje solamente los nombres de los archivos presentes, evitando los directorios. Luego se diseñó un esquema del cliente con las posibles clases necesarias y las interrelaciones entre estas (este esquema fue acomodándose a las necesidades que iban surgiendo a medida que el proyecto tomaba forma). También se agregó soporte de contraseñas para inicio de sesión básico (sin un nivel de seguridad correcto).

En lo sucesivo se realizaron mejoras en el loggin, el nivel de seguridad, y el estado de conexión del cliente con el servidor.

En cuanto al desarrollo de interfaz gráfica, se realizó la instalación de la librería gtkmm 3.0 y del programa de desarrollo Glade, se modificaron los archivos Makefile para soporte de la librería y se realizaron las primeras pruebas de interfaz básica para el cliente.

Se desarrolló un prototipo de interfaz de loggin con posibilidad de establecer configuración sobre el cliente.

Se diseñó un esquema para el servidor, primero armando un diagrama tentativo de la posible disposición de clases y se implementó un template sin consideración de archivos y con un manejo total sobre memoria, para realizar pruebas de transmisión de datos. Con esta base establecida, se fueron agregando de a poco funcionalidades al servidor.

Se decidió separar la actualización inicial del cliente, en la cual se conecta al servidor y sincroniza todos los archivos de la de inspección general, regulada por un intervalo de polling. Se decidió que si ocurren modificaciones de algún archivo del cliente offline, se revierten los cambios volviendo a la versión que contiene el server.

Se detectó que la clase manejador de archivos sería necesaria tanto para el cliente como para el servidor y se hizo común a ambos.

Se avanzó sobre el envió y recepción de archivos.

### 4.3. Segunda Etapa

Se investigó y realizaron las primeras pruebas sobre HMAC para la verificación de usuarios y el almacenamiento de claves.

Se comenzó con el diseño de la aplicación monitor, en un primer momento integrada a la aplicación server y con posibilidad de trabajar únicamente de manera local.

Se realizaron pruebas para trabajar con interfaz gráfica con varios hilos e interacción del usuario.

### 4.4. Última Etapa

Se modificó la aplicación monitor, para que esta sea lo más remota posible y pueda ser iniciada por separado del servidor. Se le agregó la funcionalidad de contener un gráfico con las estadísticas de "cantidad de bytes almacenados" vs. "tiempo".

Se continuó con más pruebas de los casos especiales de sincronización entre servidor y cliente, detectandose posibles errores.

## 5. Inconvenientes encontrados

En la puesta en ejecución de este proyecto, nos hemos encontrado con varios problemas o inconvenientes. A continuación se da una lista con una breve descripción de estos:

- Uno de los problemas con los que nos encontramos fue el de no tomar en consideración la sincronización por bloques de los archivos al comienzo. Se decidió dejar para más adelante, pensando que sería de fácil adaptación, pero luego no lo fue. Se tuvieron que realizar muchas modificaciones para lograr incorporar la sincronización de a bloques.
- Otro problema que enfrentamos fue el de no considerar la fecha de modificación como otra verificación sobre los cambios en los archivos, antes de realizar el hash de cada contenido y comparar con los hash viejos. De haber considerado la fecha de modificación, sería más eficiente el uso de recursos.
- Un inconveniente no menor que tuvimos, fue el de analizar los casos posibles de interacción entre usuarios y servidor. Se pudieron detectar casos simples y se logró adaptar el programa para que actúe de forma consistente en cada uno, pero luego surgieron muchos casos especiales no considerados de antemano. Y se volvió una tarea difícil lograr que actúe de forma predecible y estable en cada uno de estos casos.
- Otro problema, fue el de considerar enviar los archivos en formato hexadecimal. Así, se está duplicando el tamaño del archivo, y resulta completamente ineficiente a usos prácticos. Derivaron otros problemas a partir de este, ya que la conversión a hexadecimal es muy lenta y generaba conflictos en la detección de modificaciones en el directorio.

## 6. Análisis de puntos pendientes

En esta sección se quiere analizar qué puntos se pueden ampliar y mejorar de este proyecto. A continuación, se da una lista detallando qué aspectos se mejorarían y cuales se incluirían en el proyecto, de tener la posibilidad en el futuro:

- Una optimización que debería realizarse es la de enviar los archivos en binario sin previa conversión a hexadecimal. Se están consumiendo recursos innecesariamente para realizar esta acción, por lo que debería ser descartada esta implementación. Para realizar esto, se debería analizar la posibilidad de envío de mensajes en paquetes de datos con la longitud fija o con un argumento que simbolice la longitud del paquete.
- Otra mejora de eficiencia sería la de considerar el caso en que se renombra un archivo como tal, y no como si fuese una baja seguida de una alta de archivo. Aquí se están perdiendo muchos recursos, ya que se deben enviar los cambios al servidor de dos archivos, y no de uno sólo, en cuyo caso no debería enviarse el contenido del archivo, sino solamente su nombre.

- Una posible mejora podría ser la de la inclusión de la fecha de modificación en la detección de cambios del directorio. En este momento, se están calculando los hash de los archivos y se están comparando con los que se tienen en el servidor. Este método no es tan práctico, como antes comparar la fecha de modificación, y si esta es diferente, ahí sí realizar la comparación de los hash.
- Una mejora en cuanto a la robustez de la aplicación, sería agregarle métodos extras de seguridad. Actualmente se está implementando un algoritmo que permite comprobar autenticidad e integridad de los datos (hmac). Se está utilizando la clave del usuario para la creación de la firma que se envía adjunta al mensaje, y esto no es muy seguro, ya que cualquiera puede interceptar el mensaje de inicio de sesión y obtener la clave. Podría incluirse un protocolo, para el envío de información más segura y el encriptado de los mensajes. Hay muchas opciones viables de seguridad que pueden ser consideradas.
- Podría considerarse el agregado de funcionalidades tales como el inicio de la aplicación con el sistema operativo y que esta corra de fondo.
- Un agregado interesante sería el de aceptar directorios anidados en las carpetas que se sincronizan. Actualmente se saltean los directorios dentro de directorios y se envían solamente los archivos que están a un nivel de profundidad. Este es un caso muy particular, y sería bueno considerar incluir estos cambios a futuro.
- Un adicional de funcionalidad que podría implementarse, es el de permitir la configuración del el ancho de banda que se quiere que el programa utilice. Esta opción es implementada por la aplicación tomada como referencia (Dropbox) y puede mejorar el rendimiento de la aplicación.
- Una funcionalidad que debería ser agregada es la de aplicar seguridad fuerte a la comunicación entre las aplicaciones monitor y servidor, ya que en este momento no poseen.
- Una mejora a realizar en la aplicación monitor es permitir que la misma pueda consultar en permanente el estado del servidor. Actualmente si el servidor se da de baja, la aplicacion lo muestra pero si luego este se da de alta, la aplicacion no lo detecta.
- Otra funcionalidad que puede agregarse sería la de poder ver el archivo de log en forma remota, desde la aplicación monitor.

# 7. Herramientas

A continuación se listan las herramientas auxiliares que han sido utilizadas a lo largo de la realización del proyecto:

- **GIT**[1], controlador de versiones (`http://git-scm.com/`);
- **Glade**, editor de interfaz gráfica (`http://glade.gnome.org/`);
- **Sublime Text 2**, editor de texto (`http://www.sublimetext.com/2`);
- **Gedit**, editor de texto (`http://projects.gnome.org/gedit/`);
- **Valgrind**, herramienta de depuración de problemas de memoria y rendimiento de programas (`http://www.valgrind.org/`);
- **GNU Debugger**, depurador estándar para el compilador GNU (`http://www.gnu.org/software/gdb/`);
- **LaTeX**, sistema de composición de textos (`http://www.latex-project.org/`).

[1]Se ha utilizado el servicio de control de versiones *GitHub*[2]. Puede accederse al repositorio del grupo ingresando en `https://github.com/federicomrossi/7542-tp-final-grupo04`.

# 8. Conclusiones

A lo largo del desarrollo de este proyecto aprendimos la importancia del establecimiento de buenos diseños de base, considerando las posibles modificaciones que se pueden tener en el futuro. Además, es importante la implementación de prototipos, que permitan analizar incrementalmente las funcionalidades que se van agregando a la aplicación. También resulta importante definir un protocolo que permita un desarrollo continuo e integrable.

Es necesario obligarse a poner metas intermedias y a intentar cumplirlas dentro del plazo establecido, o dentro de un rango razonable de tiempo. Además se debe ir avanzando en la implementación de a una funcionalidad por vez, dando así lugar a la realización de pruebas incrementales que permitan avanzar con pasos firmes.

# Referencias


[1] DropBox, `http://www.dropbox.com/`

[2] GitHub, `https://github.com/`