Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

ALMACENAMIENTO AUTOMÁTICO EN LA NUBE DE MEGA

Autor: Alejandro de Bona Moliz

Director: Jorge Dávila Muro

MADRID, JUNIO 2018

ÍNDICE DE CONTENIDOS.

INDICE DE FIGURAS.

RESUMEN.

Este trabajo de fin de grado nace por mi interés en tener un nuevo acercamiento a la seguridad informática y a los servicios de cloud computing vistos en la carrera. La propuesta de realizar una aplicación/servicio que comprende desde, el uso de bases de datos, hasta desarrollo con APIs, añadiéndole un componente de criptografía mediante diversos métodos, fue una motivación para poder desarrollar muchas de las competencias obtenidas durante mis estudios universitarios.

El objetivo principal de este trabajo es diseñar, implementar y poner a prueba una aplicación que permita el almacenamiento y recuperación en la nube de almacenamiento de MEGA.

Se trata de dos aplicaciones, una para el almacenado y otra para la recuperación, que permitirán, a cualquier usuario que disponga de un ordenador con sistema operativo Windows, mantener una copia segura de sus archivos en la nube de MEGA y posteriormente descargar el archivo, sin que el usuario tenga que intervenir más que en elegir el fichero deseado, la aplicación hará el resto. Las aplicaciones no requieren instalación y están listas para usarse, solo será necesario configurar el servicio a gusto del usuario.

El aspecto de la seguridad se aplica principalmente a la subida y descarga de los ficheros, siendo el usuario el único que podrá conocer y hacer uso de los ficheros almacenados, permitiéndole saber en todo momento que los ficheros que subió son exactamente los mismos que está descargando.

ABSTRACT.

This final degree project was born out of my interest in having a new approach to computer security and the cloud computing services seen during my studies. The idea of making an application/service that incorporate the use of databases, development with APIs, adding a cryptography component through multiple methods, was a motivation to develop many of the skills taken during my university studies.

The main goal of this project is to design, implement and test an application that allows storage and recovery in the storage cloud of MEGA.

These are two applications, one for storage and the other one for recovery, which will allow any user with a Windows system keeping a secure copy of their files in the MEGA cloud, and then download the file. Without the user having to intervene just for choosing the file wanted, the application will do the rest. The applications do not require installation and are ready to be used, it will only be necessary to configure the service as the user likes.

The security side is mainly applied to the upload and download of the files, being the user the only one who will be able to know and use the uploaded files, allowing him to know at all times that the files he uploaded are exactly the same ones that he is downloading.

1. INTRODUCCIÓN.

El almacenamiento de ficheros en la nube es una situación muy cotidiana actualmente. De hecho, la informática como la conocemos se está orientando a entornos de cloud computing, a las empresas les supone un ahorro gracias a la flexibilidad y disponibilidad que aporta este sistema. Para el usuario convencional permite tener sus archivos siempre disponibles y en todos sus dispositivos a la vez. Todo esto supone un reto para la ciberseguridad puesto que estos archivos se encuentran en la red, al alcance de cualquiera y debemos actuar en consecuencia. En el trabajo a desarrollar trataremos con un servicio de almacenamiento en la nube, MEGA.

Para empezar, haremos una breve cronología para entender como hemos llegado a esta situación.

El concepto de "nube" empieza a fraguarse en los años 50, en referencia a compartir de forma concurrente un recurso computacional entre muchos usuarios. Alrededor de 1970, la agencia DARPA dio origen al sistema de redes que posteriormente se terminaría convirtiendo en el Internet que conocemos actualmente [1].

En 1999, uno de los pioneros en la computación en la nube fue Salesforce.com, que introdujo el concepto de entrega de aplicaciones empresariales a través de una página web. Ya en 2007, un servicio llamado Dropbox llevo el almacenamiento en la nube al alcance de cualquier persona. A este servicio le fueron saliendo competidores con la misma esencia.

Entre ellos estaba Kim Dotcom, un hacker obsesionado con la privacidad. Fundo MEGA en 2013, aun que actualmente se encuentra desvinculado. Este servicio está implementado mediante una arquitectura de servidores y un encriptado automático RSA de 2048 bits, ha esto se le suma que todos los archivos que se suban a MEGA se cifran antes de subirlos y se almacenan cifrados, según los términos de MEGA, los archivos se almacenan cifrados al igual que la contraseña con AES-128, lo que impide que puedan tener conocimiento del contenido de los ficheros o de la contraseña. Ambos se almacenan por separado.

Todo esto hace presagiar que nuestros archivos estarían seguros. Pero expertos en el sector [2], la propia desconfianza en el almacenamiento de las claves, incluso el propio Kim [3], hacen dudar de que esto sea cierto. Por lo que la forma de "asegurarnos" que nuestros datos se almacenan de forma segura es cifrarlos nosotros antes de subirlo. Lo cual se detallará más adelante.

Esto motiva el diseño y desarrollo de una aplicación que permita al usuario aprovechar el almacenamiento en la nube de MEGA, mediante los métodos que ellos mismos proporcionan y añadiéndole un extra de seguridad. Mediante un directorio definido la aplicación subirá los ficheros del directorio en cuestión con la nube, cifrándolos como

paso intermedio. Posteriormente podremos descargarlos obteniendo una copia del original.

Para el desarrollo de la aplicación se usará la API [4] oficial de MEGA, que contiene los métodos que se pueden usar con su interfaz, intentando usar el menor número de librerías adicionales posible, con ello se pretende conseguir una implementación lo más independiente, facilitando el mantenimiento. Con esto también se quiere conseguir una aplicación multiplataforma con las menores modificaciones posibles.

Los objetivos principales para este proyecto son:

- Conocer el proceso que conlleva diseñar y desarrollar una aplicación mediante buenas prácticas de programación para su fácil comprensión.
- Desarrollar una aplicación modular, fácil de ampliar y mantener de sus funciones.
- Aprender e implementar la API que aporta el servicio.
- Desarrollar una aplicación multiplataforma.
- Comprender los requisitos de la aplicación y la mejor forma de implementarla.
- Manejo de base de datos, incluyendo el desarrollo de pruebas de integridad propias y cifrados externos a MEGA.
- Permitir al usuario elegir diferentes opciones de configuración.
- La instalación y el uso de la aplicación debe ser lo más sencillo posible.

2. ESTADO DEL ARTE.

Este capítulo se versará las distintas tecnologías utilizadas para el desarrollo del proyecto.

2.1. SQL.

Es un lenguaje de consultas estructuradas que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos [5].

Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

Figura 1. Logo SQL.

2.2. MySQL.

Es un gestor de bases de datos relacionales de código abierto, que fue lanzado en 1995. Esta entre los tres gestores de bases de datos más usados del mundo, junto con Oracle y Microsoft SQL Server, además es el más usado de código abierto. [6]

Entre sus características principales [7] destacan:

- Velocidad y robustez. Portabilidad de sistemas.
- Permite persistencia.
- APIs disponibles en múltiples lenguajes.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Soporta gran cantidad de tipos de datos para las columnas.
- Gran capacidad de almacenamiento y escalabilidad.

Figura 2. Logo MySQL.

2.3. C++.

Es un lenguaje de programación diseñado a principios de los años 80, este amplia el lenguaje de programación C, siendo un lenguaje orientado a objetos. [8]

Entre sus características destacan:

- Una gran biblioteca de funciones.
- Un lenguaje muy versátil y portable.
- Está recomendando para el desarrollo de software reutilizable, por lo tanto, reduce el esfuerzo desarrollando un software eficiente.
- Es un lenguaje independiente de la máquina.
- Se puede usar constructores de alto nivel y para actividades de bajo nivel.
- Detección de errores mejorable.

Figura 3. Logo C++.

2.4. SDK MEGA.

Un SDK por sus siglas en inglés "Software Development Kit" es el conjunto de herramientas de software necesarias para desarrollar aplicaciones. Particularmente usaremos el SDK que proporciona MEGA [9]. Consiste en código y documentación que permite utilizar la funcionalidad de la API de MEGA, las cuales usaremos para desarrollar la aplicación.

2.5. Visual Studio 2017

Es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft [10]. Un IDE es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Visual Studio soporta múltiples lenguajes y nos permite depurar, generar y publicar aplicaciones para Android, iOS, Windows, la Web y la nube.

Figura 4. Logo Visual Studio 2017.

2.6. Crypto++.

La API de MEGA se apoya en diversas librerías externas, entre ellas crypto++, una librería de código abierto compuesta de algoritmos criptográficos. Mediante esta librería realizaremos todas las funciones criptográficas.

Figura 5. Logo Crypto++.

3. DISEÑO

3.1. Aplicación.

En este capítulo trataremos la elección de las diferentes herramientas y el diseño de la aplicación.

La elección del lenguaje se fundamenta en varios pilares:

- El SDK de MEGA esta implementado en C++, y aun que tiene adaptaciones en múltiples lenguajes, algunas están realizadas por desarrolladores externos, además, la versión nativa estará siempre en un punto más avanzado y estable. Por lo que usar C++ facilitará el desarrollo conjuntamente con la API.

- Es un lenguaje robusto y versátil, que se ha usado y se continúa usando en multitud de sistemas operativos, software o utilidades. Esto facilita el acceso a gran cantidad de documentación.

- Existen compiladores de C++ para diferentes sistemas operativos, lo cual representa una ventaja en cuestión de portabilidad. Es posible compilar nuestro código en diferentes plataformas.

- Los programas nuevos pueden ser desarrollados en menos tiempo porque se puede rehusar el código

- Es rápido y eficiente, además posee un manejo de memoria más fácil y transparente.

Para el desarrollo usaremos Visual Studio 2017 en su versión C++, este nos permite desarrollar fácilmente una solución modular, una fácil importación de las librerías necesarias, incluido el SDK, y facilita la depuración del proyecto.

Como se mencionó en la introducción uno de los objetivos era hacer la aplicación lo más modular posible. Por lo que se tomó el siguiente diseño de la solución

La solución se compone de 3 proyectos: “Downloader”, “Uploader”, “Utiles”. Los dos primeros corresponden a las aplicaciones de bajada y subida de ficheros respectivamente, son completamente independientes la una de la otra, esto nos permitirá ejecutar una aplicación sin necesidad de hacer uso de la otra. En ambos proyectos se ha separado el método principal (main) del resto de la aplicación.

Mientras que ambos referencian a “Utiles”, contiene los métodos que son comunes a ambos proyectos.

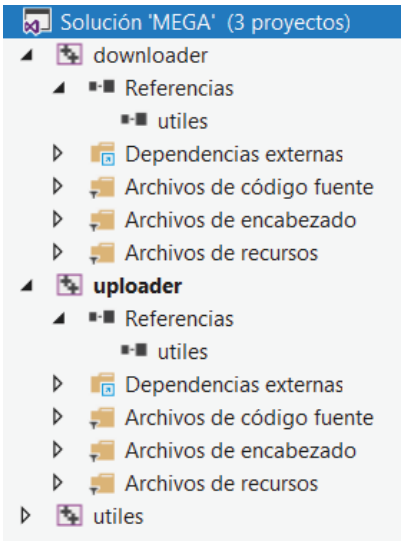

Figura 6. Diseño de la solución.

Se ha introducido una particularidad en el diseño para la subida de ficheros, por la que los ficheros se almacenaran en carpetas de hasta 20000 ficheros, una vez llegado al máximo se creará una nueva carpeta. Las carpetas serán numeradas empezando por el "1" y se incrementara en 1 tanto como sea necesario. Esto se debe a la limitación que presentan muchos gestores y al visualizar los ficheros.

La aplicación de descarga también permite el borrado de ficheros. Una vez elijamos un fichero tendremos la opción de borrar o descargar el fichero.

Cada aplicación tiene su propio fichero de log en el podremos consultar los pasos que siga la aplicación, así como los errores que pudieran producirse.

3.2. Fichero de configuración.

```
[MEGA]
MAIL = correoMega@alumnos.upm.es
PASSWORD = TFG2018
[BBDD]
HOSTNAME = Servidor:3306
USERNAME = tfg2018
PASSWORD = tfg2018
SCHEMA = BBDD
[PATHS]
DIRECTORIOSUBIDA = C:\TFG\Subida\
DIRECTORIOTMP = C:\TFG\TMP\
DIRECTORIODESCARGA = C:\TFG\Descargas\
[ALGORITMOS]
;Funciones Hash disponibles: SHA1, SHA224, SHA256, SHA384, SHA512,
;TIGER, WHIRLPOOL, RIPEMD160, RIPEMD320, RIPEMD128, RIPEMD256, MD5
HASH = TIGER
;Algoritmos de cifrado disponibles: AES, IDEA, DES, BLOWFISH, RC5
CIFRADO = IDEA
```

Figura 7. Ejemplo fichero Config.ini.

Las aplicaciones están diseñadas de forma que cualquier usuario pueda poner en un fichero .ini [11] todos los parámetros de configuración de la aplicación, este será "Config.ini" [*Figura 7]*. Sólo tiene que abrirlo con un editor de texto y añadir los parámetros necesarios. Aquí encontraremos cuatro secciones:

- MEGA. En esta sección están los parámetros para hacer el login en nuestra cuenta de MEGA. MAIL y PASSWORD.

- BBDD. Toda la configuración de la conexión a la base de datos está bajo esta sección. Entre ellos tenemos el HOSTNAME, para la dirección del servidor; USERNAME y PASSWORD, credenciales para acceder al servidor; y por último SCHEMA, el nombre de la base de datos en la cual almacenaremos todos los datos de los ficheros

- PATHS. Aquí se encontaran todos los directorios que usarán las aplicaciones, deben tener el formato que se muestra en el fichero de ejemplo:

  C:\TFG\Tipo

  DIRECTORIOSUBIDA. En el que ubicaremos los ficheros que queramos subir. Este directorio borra los ficheros una vez cifrados los ficheros para evitar tener copias indeseadas. DIRECTORIOTMP. Como paso intermedio a la subida el fichero cifrado se ubicará en está carpeta hasta completar la subida. Como las aplicaciones son independientes estos directorios solo son necesarios configurarlos para el uso del uploader.

  Por el contrario, el DIRECTORIODESCARGA solo será necesario configurarlo en el downloader. En el encontraremos todos los ficheros que descarguemos con la aplicación.

  Las rutas de los directorios deben ser únicas para cada parámetro.

- ALGORITMOS. Las aplicaciones usan funciones *HASH* [12] y algoritmos de *CIFRADO*, para garantizar la integridad de los ficheros y para cifrarlos respectivamente. Estos se podrán elegir entre una amplia lista. Como se explico en el apartado anterior, tanto las funciones Hash como los algoritmos de cifrado provienen de la librería Crypto++. Estos parámetros solo necesitarán ser establecidos para la aplicación de subida.

  Entre las funciones Hash podemos encontrar: SHA1, SHA224, SHA256, SHA384, SHA512, TIGER, WHIRLPOOL, RIPEMD160, RIPEMD320, RIPEMD128, RIPEMD256, MD5.

Los algoritmos de cifrado disponibles son: AES, IDEA, DES, BLOWFISH, RC5. Todos ellos se usan en modo CBC [13].

3.3. Base de datos.

El diseño de base de datos es muy sencillo, consiste en una única tabla, que almacena todos los datos de los ficheros que subamos, para poder descargarlos posteriormente. Pese ha su sencillez es clave en la aplicación puesto que contiene todos los datos para poder descargar los ficheros. En caso de que se borraran los registros de la base de datos perderíamos cualquier posibilidad de recuperarlos, porque, aunque físicamente si estarían en la nube, no sabríamos cual son ni como descifrarlos. En este caso veríamos lo mismo que cualquier usuario que tuviera acceso solo a nuestra cuenta de MEGA, ficheros con datos incomprensibles.

El lenguaje escogido ha sido SQL porque para un sistema tan sencillo el sistema tradicional funciona perfectamente, nuestras consultas en base de datos se basan en un INSERT y varias SELECT, por lo que con SQL es muy fácil almacenar los datos de los ficheros.

En cuando al gestor se ha elegido MySQL principalmente por:

- Es una base de datos gratuita. Al ser de código abierto, no tiene coste.
- Su facilidad de uso y rapidez.
- Pocos requerimientos y eficiencia de memoria.

Para mostrar la completa independencia de las aplicaciones se ha creado una base de datos en red usando el servidor MySQL de db4free.net. Gracias al cual podemos demostrar la modularidad del servicio, al tener la base de datos en red y dos aplicaciones independientes, podremos tener cada una en una máquina diferente funcionando simultáneamente subiendo ficheros en uno y bajándolos en otro. Pero se puede usar cualquier gestor de base de datos, solo debemos ejecutar el script “Creación_Tabla.sql” que se adjunta con la aplicación, en la base de datos que deseemos, la misma que hayamos configurado en el “Config.ini”, esto creará una tabla t_ficheros que almacenará toda la información de los ficheros:

- Nombre: Se trata del nombre original del fichero.
- Nombre cifrado: Nombre generado aleatoriamente con una longitud de 256 caracteres, es el nombre que tendrá el fichero en la nube.
- Path: Carpeta a la que se subirá el fichero en la nube.
- Hash_pre: Resultado de aplicar la función Hash elegida antes de cifrar el fichero, se almacena en hexadecimal.
- Hash_post: Resultado de aplicar la función Hash elegida después de cifrar el fichero, se almacena en hexadecimal.

- Clave: Parámetro de 16 bytes que se usa el algoritmo de cifrado como clave.
- IV: Vector de inicialización [14]. Parámetro de 16 bytes utilizado por el algoritmo de cifrado.
- Algoritmo_hash: Función Hash utilizada.
- Algoritmo_cifrado: Algoritmo cifrado usado.
- Size: Tamaño del fichero.
- Fecha_modificación: Es un parámetro que se introduce automáticamente cuando se realiza la inserción con los datos del fichero, indica de cuando son los dato.

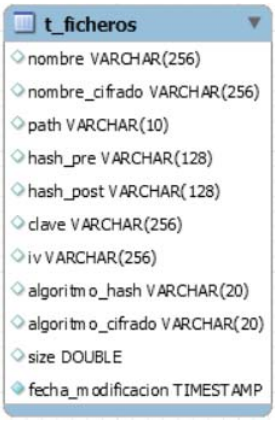

Figura 8. Estructura tabla t_ficheros.

4. DESARROLLO.

En esta sección veremos el desarrollo de la aplicación siguiendo el diseño expuesto en el apartado anterior.

4.1. Aplicación.

4.1.1. Utiles.

Para comentar el desarrollo de la aplicación empezaremos por el proyecto común a las dos aplicaciones, contiene métodos que se usan en ambos proyectos. Está dividido en 3 ficheros que en función del contenido:

4.1.1.1. Utiles.cpp

Contiene los métodos comunes a ambas aplicaciones, que no se categorizarán. Contiene una clase objeto Fichero que permitirá agrupar todos los datos del fichero en sus comunicaciones con la base de datos.

También se encarga de carga los parámetros de configuración del fichero Config.ini. Este método se ejecuta solo una vez y es nada más arrancar la aplicación. Obtiene todos los parámetros necesarios para la aplicación que vayamos a usar y los guarda en un diccionario a modo de caché con todos los elementos. Esto permitirá tener siempre disponibles los parámetros introducidos, evitando constantes llamadas al fichero y manteniendo la coherencia durante todo el tiempo que se este ejecutando la aplicación. Cualquier cambio en el fichero Config.ini no será efectivo hasta que se reinicie la aplicación.

Como paso previo a la carga del parámetro en la caché se comprueba que el parámetro exista y tenga el formato correcto. En caso de no serlo devolverá un error describiendo los parámetros incorrectos. Esta función permite también determinar que aplicación se esta ejecutando, esto conlleva que solo se comprobarán los parámetros necesarios para cada aplicación, omitiendo el resto.

Existe un método más, una función de ofuscación que se encarga de generar una cadena de caracteres alfanuméricos aleatorios. Como se ha comentado anteriormente se busca dificultar cualquier posible análisis o lectura de los ficheros. En nuestro caso será una cadena de 256 caracteres.

4.1.1.2. Bbdd.cpp

En este apartado encontraremos todos los métodos que se encargan de la comunicación con la base de datos.

Para comprobar que los parámetros de base de datos introducidos en el Config.ini son correctos, dispone de un método que se encarga de probar la conexión con el servidor a modo de test. En caso de no poder realizar correctamente la conexión devolverá un error al tratar de realizarla.

Tiene dos métodos para la obtención de datos de los ficheros:

- Uno que se encarga de recoger toda la información de los ficheros cuyo nombre coincida con el introducido, en la aplicación de descarga, este devolverá los nombres de los ficheros, que coincidirán entre ellos, así como sus tamaños y las fechas en las que se subieron.
- El segundo se encarga de traer todos los datos necesarios para descargar, o borrar, el fichero que se haya seleccionado.

En caso de seleccionar la opción de borrado, existe un método que simplemente borra el registro que contiene dicho fichero. Este proceso es irreversible, por ello solicitará al usuario una confirmación para eliminarlo.

Hay un caso más en el que se realiza una conexión con el servidor, es para insertar los datos de los ficheros que se suban a la nube, estos serán agrupados en un objeto Fichero que será el cual recoja este método para insertar las credenciales en t_ficheros.

4.1.1.3. Seguridad.cpp

El fichero de seguridad se encarga de los métodos criptográficos. En el caso del uploader tanto las funciones hash, como los algoritmos de cifrados, se seleccionarán en el fichero Config.ini:

- Funciones Hash: Tomará la elección de función, o la almacenada, y la aplicará sobre el fichero obteniendo como valor de retorno el cálculo resultante.

- Algoritmo de cifrado: Utilizada solo por el uploader, genera un par de clave y vector de inicialización, estas serán usadas por el algoritmo elegido para cifrar el fichero, almacenando las claves para su posterior almacenado en base de datos.

- Algoritmo de descifrado: Utilizado solo por el downloader, tomará las claves y el algoritmo almacenados en base de datos y descifrará el fichero.

4.1.2. Uploader.

Esta aplicación se encargará de subir los ficheros a nuestra cuenta de la nube de MEGA.

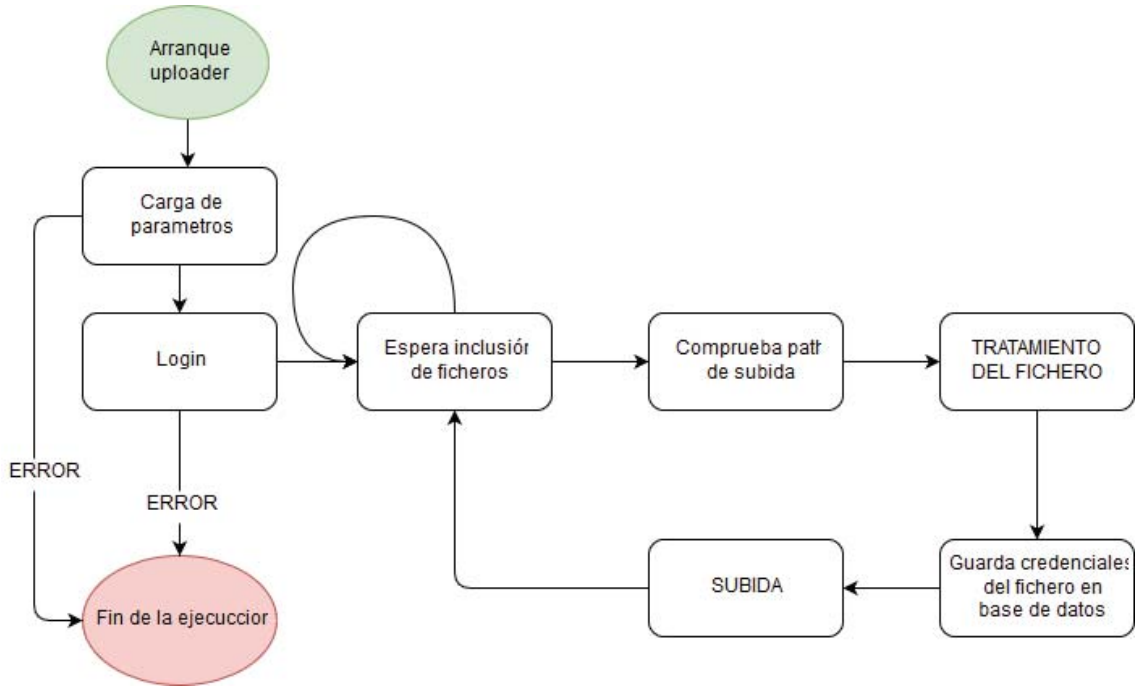

Figura 9. Diagrama de flujo uploader.

Lo primero que sucede al arrancar la aplicación es la cargar de parámetros, como se expuso anteriormente se obtienen del fichero Config.ini y se almacenan para que estén disponibles durante toda la ejecución. En caso de que no se carguen correctamente, se indicará el parámetro o los parámetros que producen el error, acto seguido terminará la ejecución.

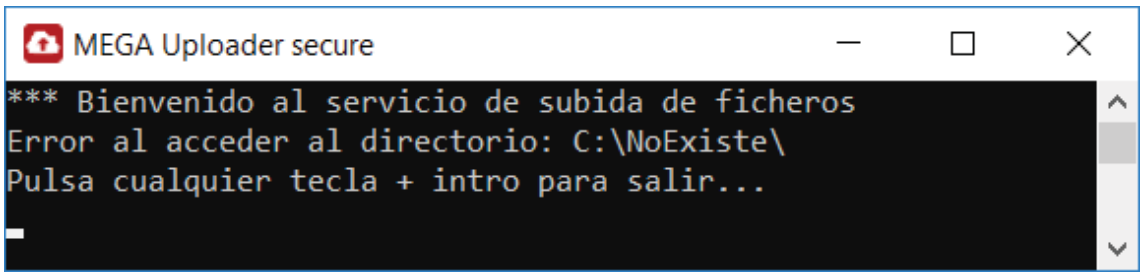

Figura 10. Pantalla error de parametrización.

Una vez se han cargado los parámetros se realiza el login.

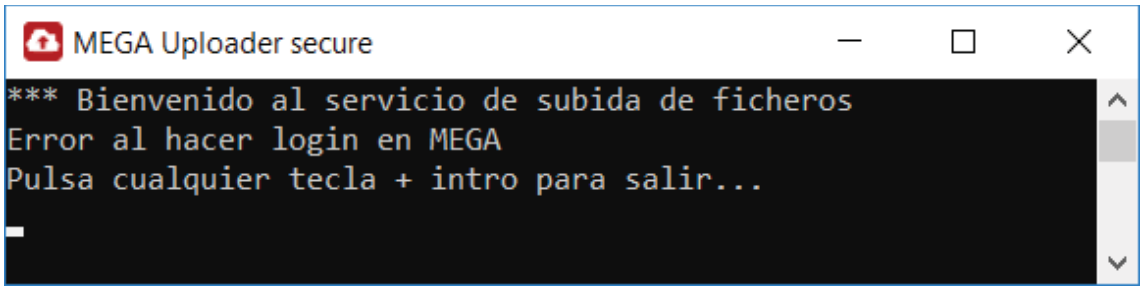

Figura 11. Pantalla error login en MEGA.

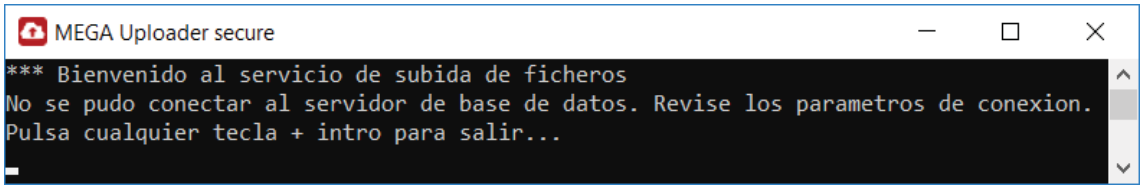

Figura 12. Pantalla error conexión al servidor de base de datos.

Una vez hecho el login comenzará con la monitorización del directorio configurado. Como consecuencia cualquier fichero que se meta en este directorio será detectado por la aplicación y comenzará el proceso de subida.

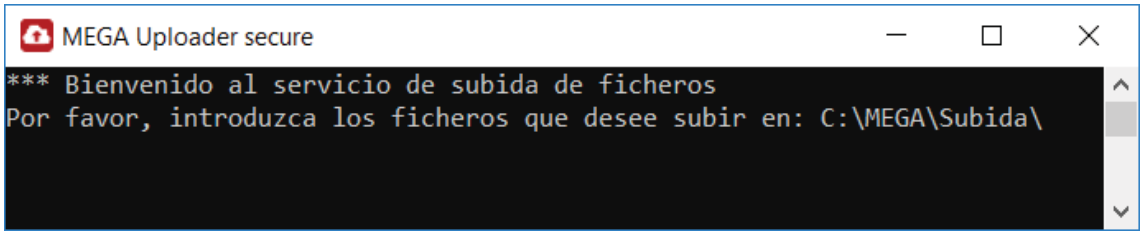

Figura 13. Pantalla de monitorización del directorio de subida.

Se pueden añadir tantos ficheros como se quiera y en cualquier momento en el que se encuentre la aplicación. Aun que la subida se realizará individualmente, los ficheros restantes se encolaran esperando su turno. Para una correcta detección de los ficheros deben introducirse directamente en la carpeta de subida, la aplicación no contempla la subida de carpetas.

Después de la detección del fichero el siguiente paso será comprobar el directorio de la nube de MEGA que corresponde al nuevo fichero, este parámetro se almacenará en un objeto Fichero creado al principio del proceso.

Figura 14. Estructura MEGA.

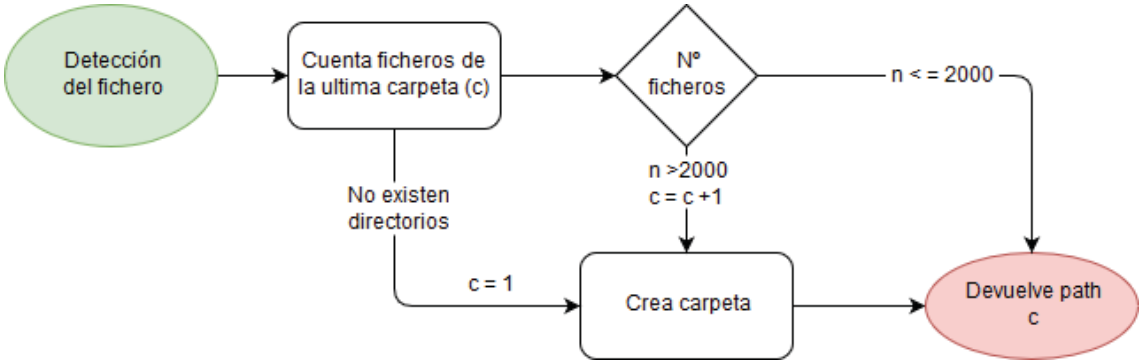

Figura 15. Diagrama de flujo comprobación directorio subida.

Sabiendo el directorio del fichero al que se subirá el fichero, se procederá a tratar el fichero que se subirá.

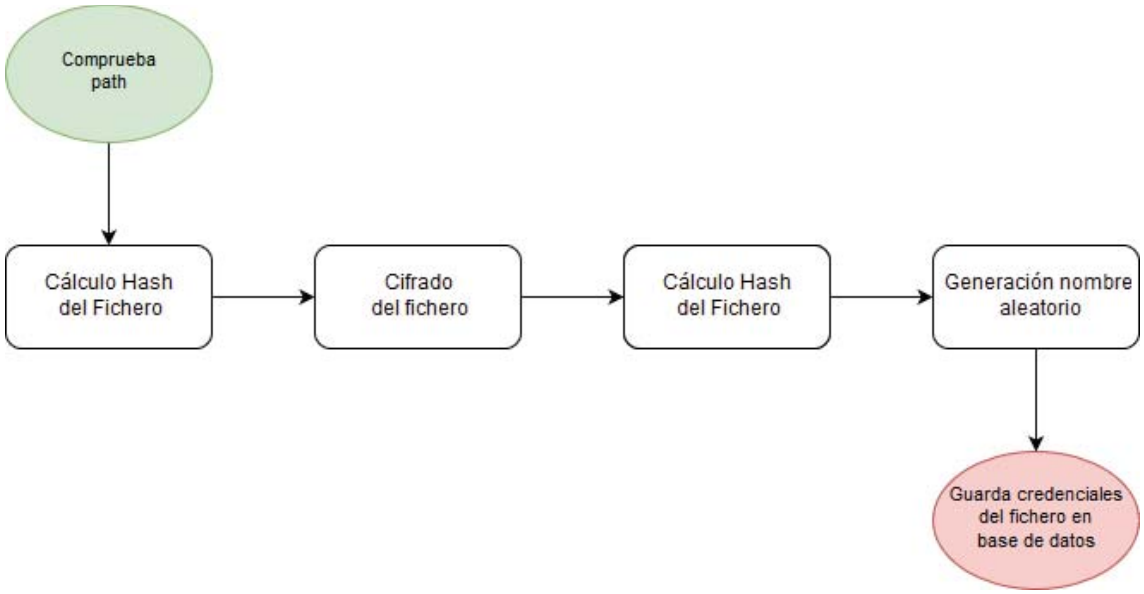

Figura 16. Diagrama de flujo tratamiento fichero subida.

El tratamiento del fichero consiste en cuatro pasos:

- Cálculo del Hash: Con el la función Hash definida en el fichero de configuración, la aplicación calculará el Hash y el resultado lo guardará en él objeto de tipo fichero.

- Cifrado del fichero: Utilizará el algoritmo establecido en el fichero de configuración para determinar el algoritmo con el que se cifrará. Previo al cifrado genera una clave y un vector de inicialización que usará el algoritmo. Estos valores se almacenarán en el objeto para poder descifrarlo durante el proceso de descarga. Al cifrar el fichero se ocultan también los metadatos [15] del fichero. Estos proporcionan información sobre el fichero y estos podrían revelar el contenido del fichero y muchos otros valores.

- Cálculo del Hash: Se volverá a calcular el Hash esta vez del fichero cifrado, para certificar la integridad del fichero y, en caso de fallar, saber en qué momento ha sido alterado. Se llevará acabo con la misma función Hash que el anterior cálculo y como en los pasos anteriores este resultado también se almacenará.

- Generación del nombre aleatorio: Se generará una cadena de 256 caracteres que sustituirán al nombre original del fichero.

En este punto ya tendremos todos los parámetros del fichero que se va a subir, todos ellos están recogidos en el objeto, este se pasará al método que almacenará las credenciales del fichero en base de datos, será la base de datos la encargada de insertar la fecha y hora de subida del fichero.

	nombre	nombre_cifrado	path	size	algoritmo_hash	algoritmo_cifr
	megasync2.jpeg	t%HWGwP*XbTT3Knpo$YdHoMdv1...	6	0.033402	SHA512	IDEA
	MEGA01.mp4	anvezAzOTI!DA6tJ5tpeZAv$Oa%0...	6	24.267315999999997	MD5	IDEA
	MEGA01.mp4	^b^eHP3VSP^upob*u08TiN0z5a3...	6	1151.586562	SHA512	AES
	Imagen1-600x420.jpg	TSJfupiw2h49WsROR5*a8nv^CDU...	6	0.064288	SHA512	BLOWFISH
	pruebas53.zip	@a20a%$AsFB2%Jv$dVu*S7^VXt...	7	0.40658999999999995	MD5	BLOWFISH
	videoPrueba.mp4	7GZ0a2sxTP!GexWsWNO7oEAPUP...	7	321.547276	SHA512	IDEA

Figura 17. Ejemplo registros base de datos.

Por último, se llevará a cabo la subida del fichero mediante el método proporcionado por la API de MEGA.

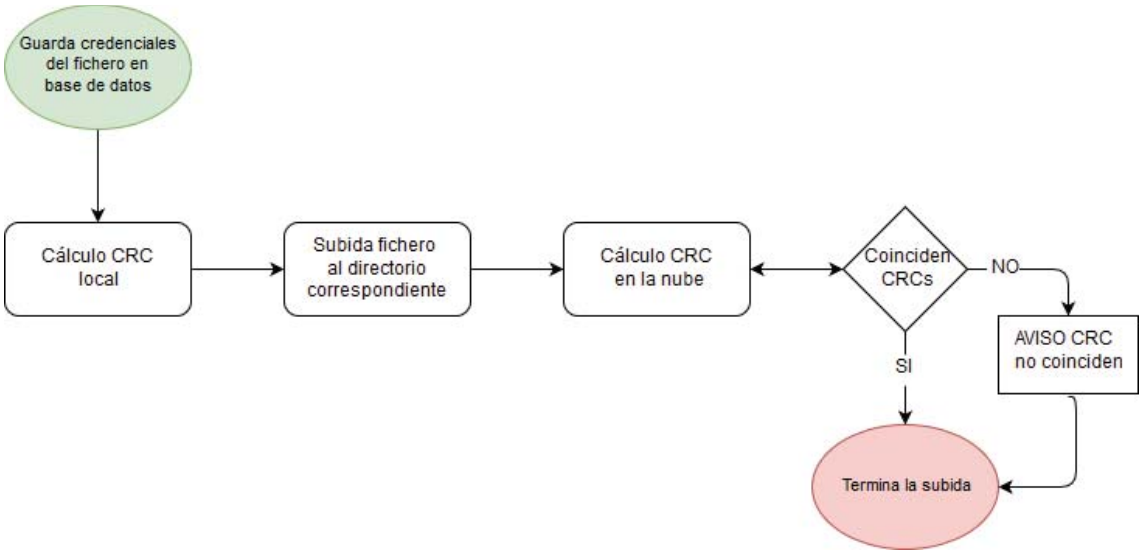

Figura 18. Diagrama de flujo subida fichero.

Para garantizar la correcta subida del fichero se usará un código de detección de errores llamado CRC [16], verificación por redundancia cíclica, que permite detectar cambios accidentales en los datos.

Para ello calcularemos el código que genera el fichero en local, después del cálculo se procederá a subir el fichero, este proceso cuenta con una barra de progreso que nos indica el estado de la subida.

Una vez termine el proceso de subida comprobaremos de nuevo el CRC que genera el fichero que, almacenado en la nube, en caso de que coincidan la subida habrá sido correcta, en caso negativo, la aplicación devolverá un mensaje avisando de la situación.

Al finalizar se eliminará el fichero de la maquina para evitar que se pueda tener conocimiento de los ficheros que se encuentran en MEGA.

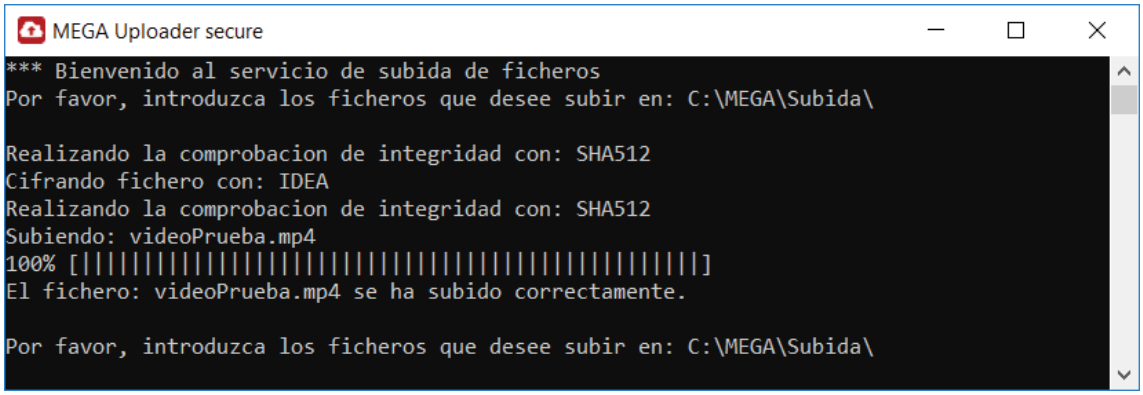

Figura 19. Pantalla subida fichero completada.

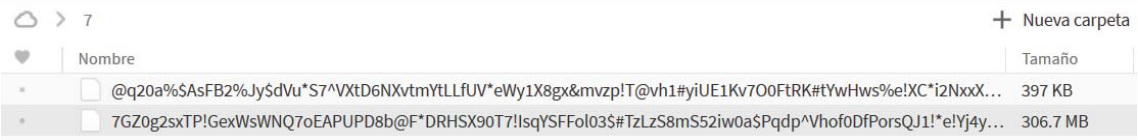

Figura 20. Registros fichero MEGA.

Cualquier cierre de la ventana, ya sea porque se ha presionado la "X" situada en la esquina superior derecha, un cierre de sesión, o cualquier otro caso en el que se cierre la ventana, conllevará un cierre de sesión de la cuenta, para minimizar los riesgos de que se pueda quedar una conexión con nuestra cuenta.

Cualquier error que se produzca después de comenzar la monitorización de ficheros no parará la aplicación, simplemente devolverá el error y volverá a solicitar la inclusión de ficheros.

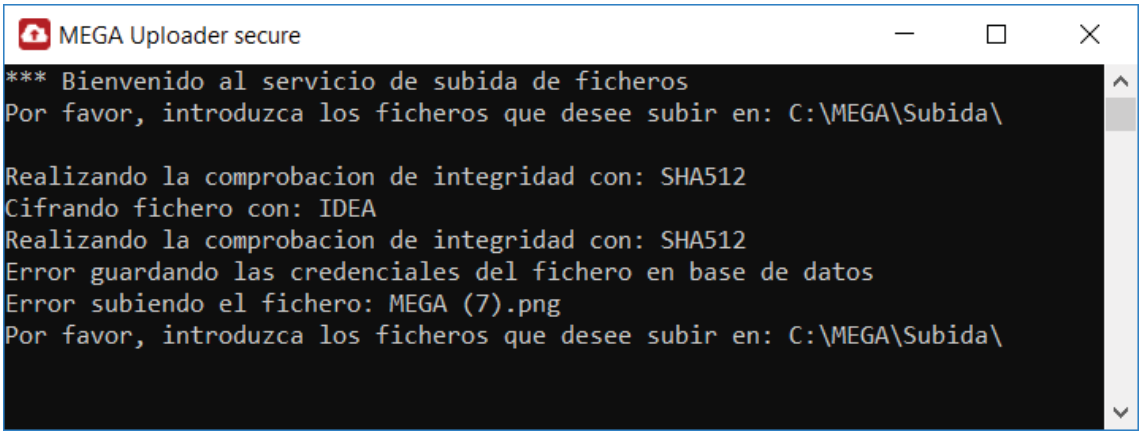

Figura 21. Pantalla error durante la subida.

4.1.3. Downloader

El downloader se encarga, como su propio nombre indica, de las descargas de los ficheros, pero además también ofrece la opción de borrar los ficheros almacenados. Los ficheros que se usarán por esta aplicación deberán haber sido subidos previamente mediante la aplicación uploader, cualquier fichero subido de otro modo será ignorado por la aplicación.

Se trata de un proceso lineal como podemos observar en el siguiente diagrama:

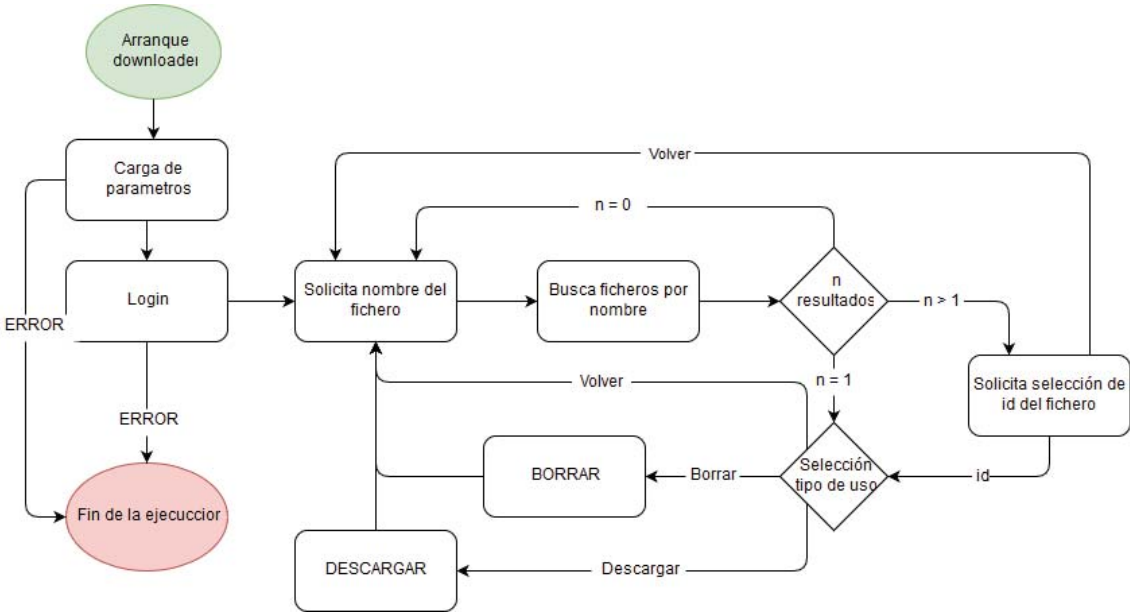

Figura 22. Diagrama de flujo downloader.

Lo primero que sucede al arrancar la aplicación es un proceso idéntico al arranque de la aplicación de subida, carga de parámetros e inicio de sesión. Pero en este caso no será necesario cargar los parámetros de la sección de algoritmos.

Una vez cargados los parámetros correctamente y realizado el inicio de sesión, la aplicación solicitará que introduzcamos el nombre del fichero que deseemos descargar.

El nombre debe ser el mismo que introdujimos durante la subida, incluida la extensión, y no es sensible a mayúsculas y minúsculas.

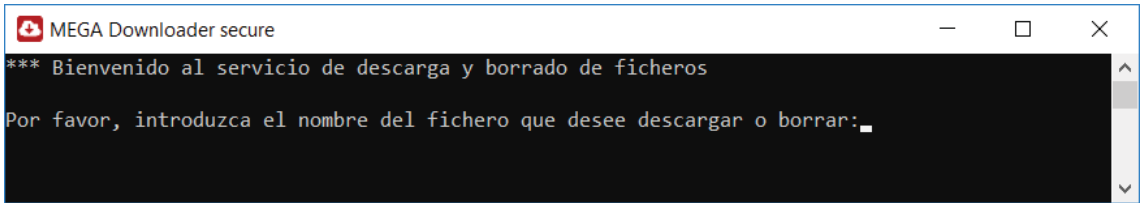

Figura 23. Pantalla solicitud de fichero.

En esta situación podemos encontrarnos tres casos:

- Que el nombre introducido no exista en la base de datos, en cuyo caso la aplicación nos lo indicará y solicitará otro nombre.

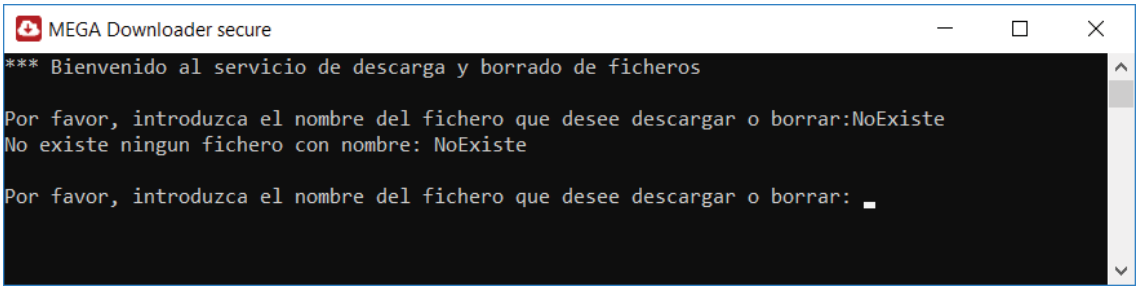

Figura 24. Pantalla búsqueda fichero inexistente.

- Que exista más de un registro en la base de datos que coincida con el nombre introducido, en este caso nos mostrará los registros cuyos nombres coincidan, con un id asociado. Mediante el id seleccionaremos el fichero deseado. También existe la posibilidad de Volver para seleccionar otro fichero.

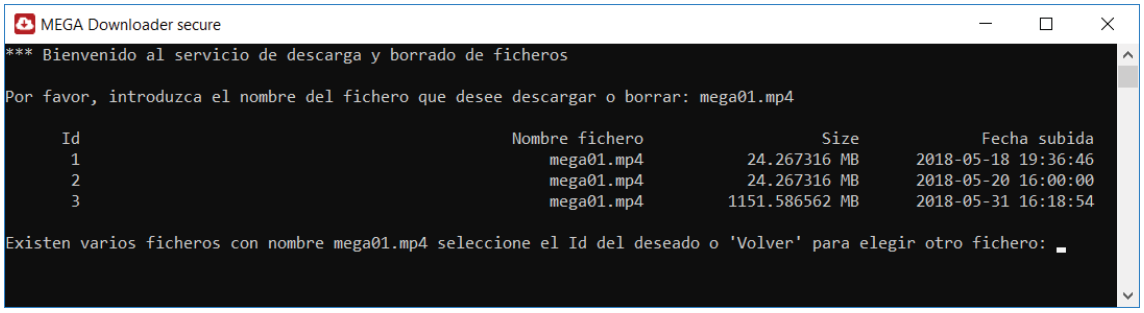

Figura 25. Pantalla multiples registros.

- Que solo exista un registro que coincida con el nombre introducido, en cuyo caso no hará falta seleccionar el fichero.

A continuación, en los casos en los que ya dispongamos del fichero a usar, nos dará la posibilidad de descargar el fichero, de borrarlo o de volver al principio del proceso y buscar otro fichero.

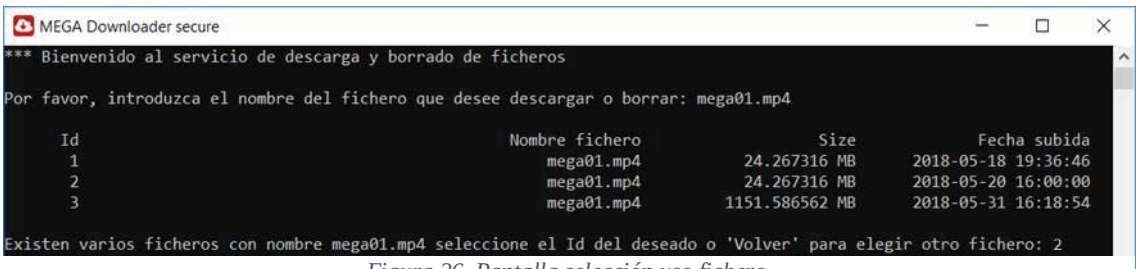

Figura 26. Pantalla selección uso fichero.

4.1.3.1. Borrar.

La opción de borrar permite eliminar cualquier registro de forma permanente, este proceso se realiza en dos pasos:

- Borrado del registro en la base de datos.
- Borrado del fichero en MEGA, borrará permanentemente el fichero de la nube.

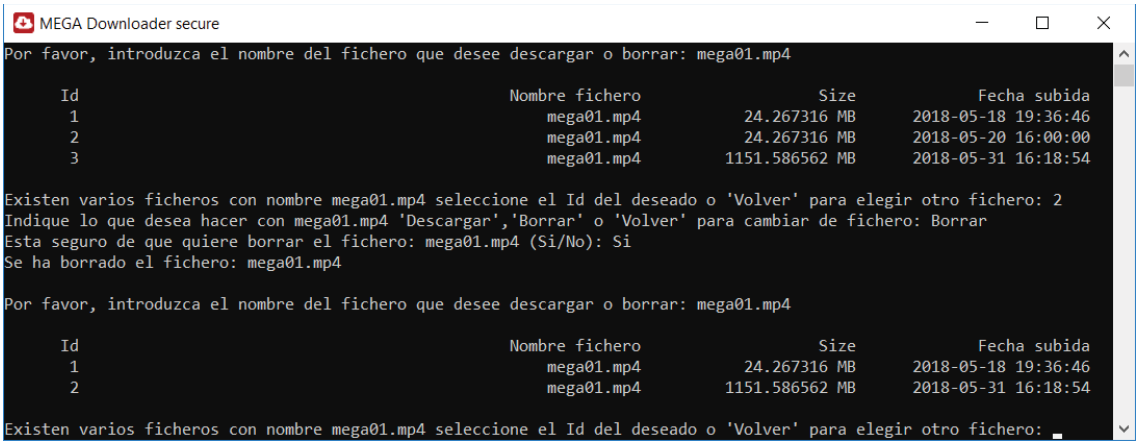

Figura 27. Pantalla borrado de fichero.

Podemos observar que el fichero ya no está disponible.

4.1.3.2. Descargar.

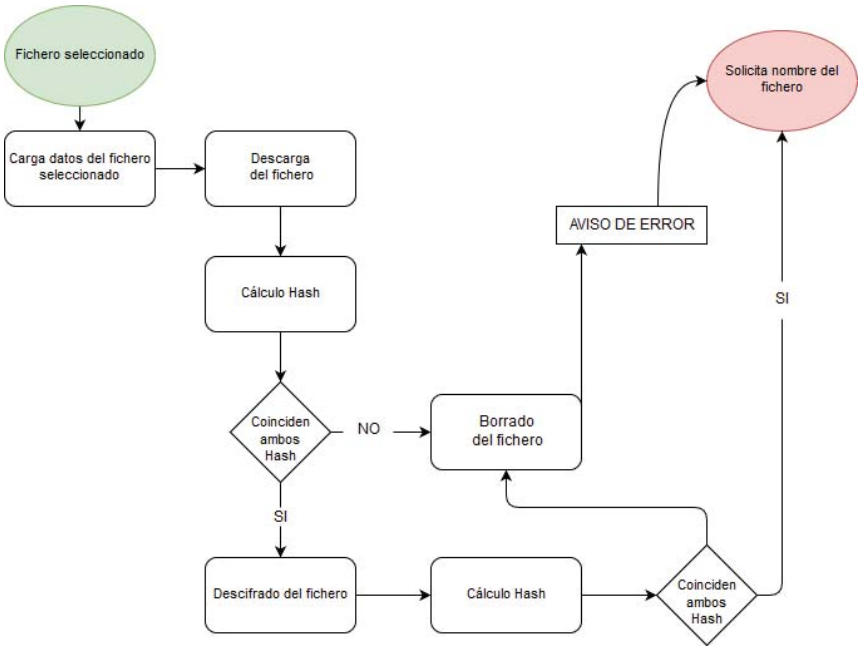

Figura 28. Diagrama de flujo descarga.

La opción de descargar permite obtener una copia del fichero que se encuentra en la nube de MEGA y descifrarlo obteniendo una copia del fichero original.

Una vez seleccionado el fichero, se realizará una nueva consulta a base de datos para obtener los parámetros para realizar la descarga. Tras realizar este proceso el siguiente paso será descargar el fichero de nuestra cuenta de MEGA, haciendo uso del método proporcionado por la API y en el directorio configurado, este estará cifrado todavía. La duración del proceso dependerá del tamaño del fichero y de nuestra conexión, para conocer el progreso dispondremos de una barra que nos indicará el estado de la descarga.

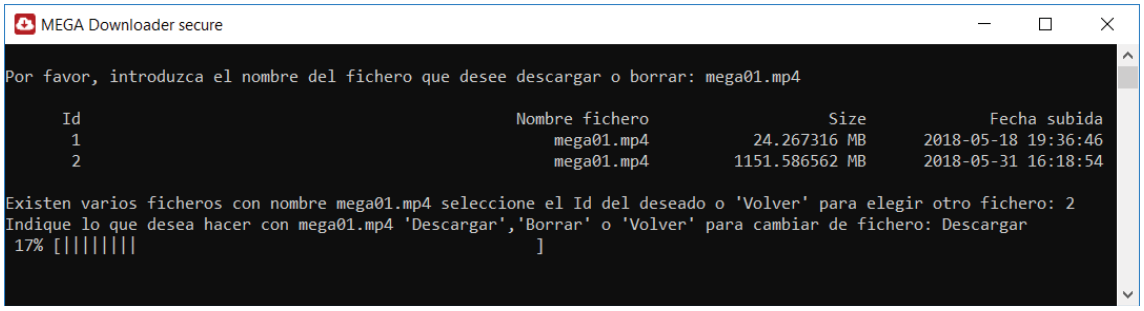

Figura 29. Pantalla descargando fichero.

Para comprobar la integridad del fichero se calculará la función Hash del fichero, esta función será la misma que se empleó durante la subida. Compararemos el resultado con él almacenado en base de datos, en caso de no coincidir, se borrará el fichero y devolverá un error. En caso contrario continuará con el proceso.

En este momento se procederá a descifrar del fichero, como con la función Hash, obtendremos el algoritmo con el que fue cifrado el fichero, y además recuperaremos las claves usadas por este. Con el fichero descifrado volveremos a calcular el hash, esta vez del fichero descifrado, en caso de no coincidir con el almacenado borrará el fichero y devolverá un mensaje de error, en el caso favorable concluirá la descarga del fichero devolviendo un mensaje favorable y solicitando la introducción de un nuevo fichero.

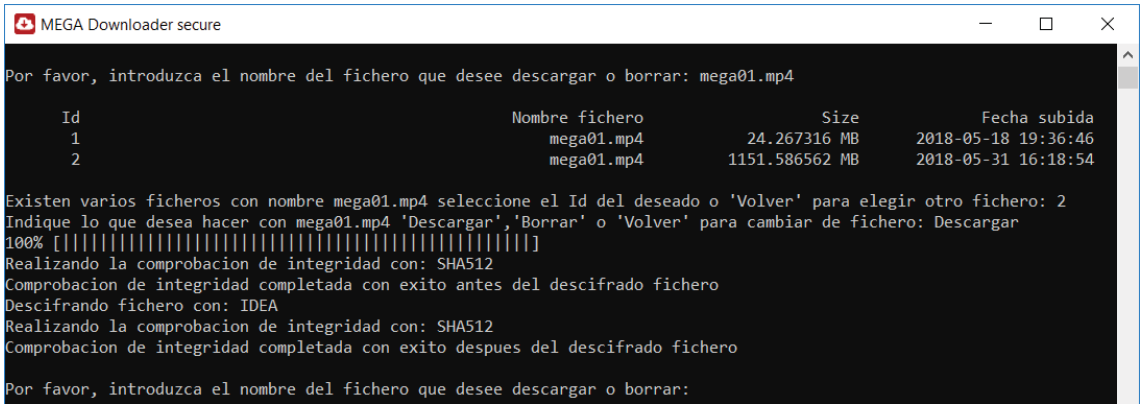

Figura 30. Pantalla descarga completada

En caso de que existiera un fichero con el mismo nombre en la carpeta de descarga lo sobrescribirá, esto sucede para evitar el llenado de la carpeta, por lo que se recomienda usar este directorio de manera temporal.

Al igual que en el uploader cualquier cierre de la ventana provocará un cierre de nuestra sesión en MEGA.

También se continuará con la política de mantener activa la aplicación siempre que esta se haya inicializado correctamente la aplicación.

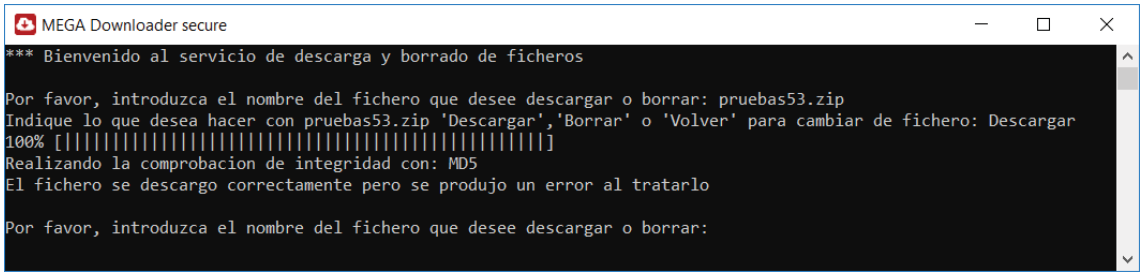

Figura 31. Error en el tratamiento del fichero después de la descarga.

4.2. Base de datos.

Al igual que el diseño el desarrollo de la base de datos ha sido muy sencillo, después de establecer todo lo que iría en la tabla, solo quedaba elaborar un script que la creara, conociendo lo que contendría cada columna solo quedaba elegir los tipos de estas.

Como se ha visto anteriormente el objeto contiene todos los valores que se van a insertar en base de datos, por lo que se ha implementado una configuración en la que no se pueden insertar valores null, esto no debería suceder ya que en caso de que los valores fueran null el error debería producirse antes, pero si se diera el caso devolvería un error al guardar las credenciales del fichero.

La fecha de modificación se inserta automáticamente al almacenar el registro y es la que se recupera a la hora de devolver los ficheros iguales en el downloader.

Por último, se ha utilizado el servidor de bases de datos gratuito que proporciona db4free.net, solo crear la cuenta obtienes unos parámetros para poder conectar con el servidor, solo es necesario ejecutar en el servidor el script de creación de la tabla y añadir los parámetros al fichero config.ini

5. RESULTADOS

En esta sección analizaremos los resultados obtenidos una vez concluido el proyecto.

El más evidente, las aplicaciones. Se han desarrollado dos aplicaciones, una para subidas de ficheros y otra para descargas, estas aplicaciones son completamente portables e independientes, se trata de unos ficheros de apenas unos megas. No requieren instalación, el único requerimiento es configurar el fichero de configuración "Config.ini".

Si se usa un servidor de base de datos en red bastará con ejecutar la aplicación deseada y seguir los pasos que se muestran en pantalla. En caso contrario debemos tener también instalado el gestor de bases de datos correspondiente.

La aplicación se ha diseñado para que su mantenimiento sea muy sencillo, la separación en múltiples métodos nos permitirá saber en cual se produce un hipotético fallo, además, la ampliación del diseño de la aplicación sería muchos más sencilla.

Estos eran los objetivos principales del proyecto, a los que se les suman:

- Implementar la API de MEGA.

- Minimizar el uso de librerías externas, solo ha sido necesario añadir una librería, y esta es la del conector MySQL. Aprovechando las librerías de las que ya hace uso MEGA para su propia implementación.

- El desarrollo se ha realizado de forma que su entendimiento sea lo más sencillo posible, documentando las partes que puedan ser más complejas.

- Todo el tratamiento del fichero se realiza sin conocimiento de MEGA lo que hace que no intervengan para nada en el fichero original, solo se usa como servicio de almacenamiento.

6. CONCLUSIONES.

Una vez finalizado este trabajo de fin de grado podemos valorarlo.

El trabajo de investigación me ha permitido observar la situación actual de los gestores de ficheros en la nube, de cómo funcionan y de las dudas acerca del tratamiento que hacen de los ficheros que se suben a sus servidores.

Esta investigación, junto con la documentación, es clave para realizar un diseño apropiado minimizando las modificaciones que pueda sufrir este durante la fase de desarrollo, al conocer las necesidades de la aplicación podemos establecer los requisitos de la esta.

En el aspecto de la seguridad, se han aplicado los conocimientos obtenidos durante la carrera, buscando las herramientas apropiadas para garantizar, en la medida de lo posible, un correcto y seguro almacenado de los ficheros.

Implementar una API o un SDK, es muy sencillo. Los métodos desarrollados que aporta MEGA, en este caso, permite multitud de funcionalidades, por lo que, mediante un estudio previo de estos, permite determinar los métodos que debemos usar para hacerla funcionar de manera eficiente.

Con todos los pasos previos realizados empezar con el desarrollo resulta más sencillo y, sobre todo, evita las constantes modificaciones en el código. Las modificaciones que haya serán más fáciles de realizar si se implementa el código de forma modular, limpia y documentada. Un aspecto clave del desarrollo es la capacidad para depurar el código, lo que nos permitirá localizar los errores que se produzcan para posteriormente corregirlos.

Por último, una buena capacidad de abstracción del proyecto permite comprender las debilidades de la aplicación, así como los puntos de mejora.

7. TRABAJOS FUTUROS.

Toda aplicación tiene margen para la mejora, ya sea para implementar nuevas funcionalidades o para fortalecer las debilidades. En esta aplicación se podrían realizar mejoras como:

- Una aplicación que cifrara las contraseñas de nuestras cuentas para después ponerlas en el config.ini. Aun que el fichero se encuentra en local y depende de la seguridad de los equipos en los que se encuentre, tener las contraseñas en un fichero de texto plano supone un riesgo a la seguridad. Esta aplicación cifraría la contraseña con una clave desconocida por cualquier usuario, pero conocida por nuestras aplicaciones, que sustituiría a la que se encuentra en el fichero. Esto evitaría el conocimiento de las contraseñas por parte de cualquier persona simplemente mirando el fichero.

- Descargas y subidas múltiples y simultáneas, actualmente ambos procesos se producen individualmente y linealmente, la posibilidad de realizar los procesos mediante hilos que permitan la subida simultanea de los ficheros reduciría los tiempos que estos conllevan, además, en el caso de las descargas permitiría introducir nombres de múltiples ficheros a descargar.

- La posibilidad de subir directorios de ficheros. Esta mejora tiene un impacto menor que las anteriores, permitiría la subida de la estructura de ficheros sin tener que sacarlos de ellos.

Estas entre otras ayudarían ha hacer la solución más robusta y completa.

8. BIBLIOGRAFÍA.

[1] Proyecto DARPA.

https://www.darpa.mil/attachments/(2O15)%20Global%20Nav%20-%20About%20Us%20-%20History%20-%20Resources%20-%2050th%20-%20Internet%20(Approved).pdf

[2] Artículo que cuestiona la seguridad de MEGA.

https://blog.erratasec.com/2013/01/mega-and-encrypted-cloud-deduplication.html

[3] Entrevista a Kim Dotcom en la que responde cuestionando la seguridad de MEGA.

https://yro.slashdot.org/story/15/07/27/200204/interviews-kim-dotcom-answers-your-questions

[4] Artículo que describe que son las APIs.

https://hipertextual.com/archivo/2014/05/que-es-api/

[5] Descripción de SQL.

https://support.office.com/es-es/article/access-sql-conceptos-b%C3%A1sicos-vocabulario-y-sintaxis-444d0303-cde1-424e-9a74-e8dc3e460671#bm1

[6] Ranking de los gestores de bases de datos más usados.

https://db-engines.com/en/ranking

[7] Descripción de MySQL.

http://ftp.tcrc.edu.tw/MySQL/doc/refman/5.0/es/features.html

[8] Libro donde se trata todo lo relativo al lenguaje C++.

The C++ Programming Language: Special Edition (3rd Edition)

[9] SDK de MEGA

https://github.com/meganz/sdk

[10] Descripción de las funcionalidades de Visual Studio.

https://www.visualstudio.com/es/vs/

[11] ¿Qué son los ficheros .ini?

https://es.wikipedia.org/wiki/INI_(extensión_de_archivo)

[12] Descripción y uso de las funciones Hash

https://latam.kaspersky.com/blog/que-es-un-hash-y-como-funciona/2806/

[13] Descripción de los modos de cifrado entre los que se encuentra CBC.

http://ocw.bib.upct.es/pluginfile.php/5310/mod_resource/content/1/MODOS_DE_OPERACION_CIFRADO_BLOQUES.pdf

[14] Descripción de los vectores de inicialización.

https://es.wikipedia.org/wiki/Vector_de_inicialización

[15] ¿Qué son los metadatos?

https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/Archivo_electronico/pae_Metadatos.html

[16] Uso de CRCs.

https://www.ecured.cu/Comprobaci%C3%B3n_de_redundancia_c%C3%ADclica

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Wed Jun 06 21:14:13 CEST 2018
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)