# *Udrive*

## *Trabajo práctico*

| | |
|---|---|
| **Objetivos** | ● Afianzar los conocimientos adquiridos durante la cursada.<br>● Poner en práctica la coordinación de tareas dentro de un grupo de trabajo.<br>● Realizar un aplicativo de complejidad media con niveles aceptables de calidad y usabilidad. |
| **Instancias de Entrega** | **Checkpoint 1:** clase 8 (dd/mm/aaaa).<br>**Checkpoint 2:** clase 11 (dd/mm/aaaa).<br>**Entrega:** clase 14 (dd/mm/aaaa). |
| **Criterios de Evaluación** | ● Empleo de buenas prácticas de programación<br>● Empleo de test unitarios<br>● Coordinación de trabajo grupal.<br>● Planificación y distribución de tareas para cumplir con los plazos de entrega pautados.<br>● Cumplimiento de todos los requerimientos técnicos y funcionales.<br>● Facilidad de instalación y ejecución del sistema final.<br>● Calidad de la documentación técnica y manuales entregados.<br>● Buena presentación del trabajo práctico y cumplimiento de las normas de entrega establecidas por la cátedra |

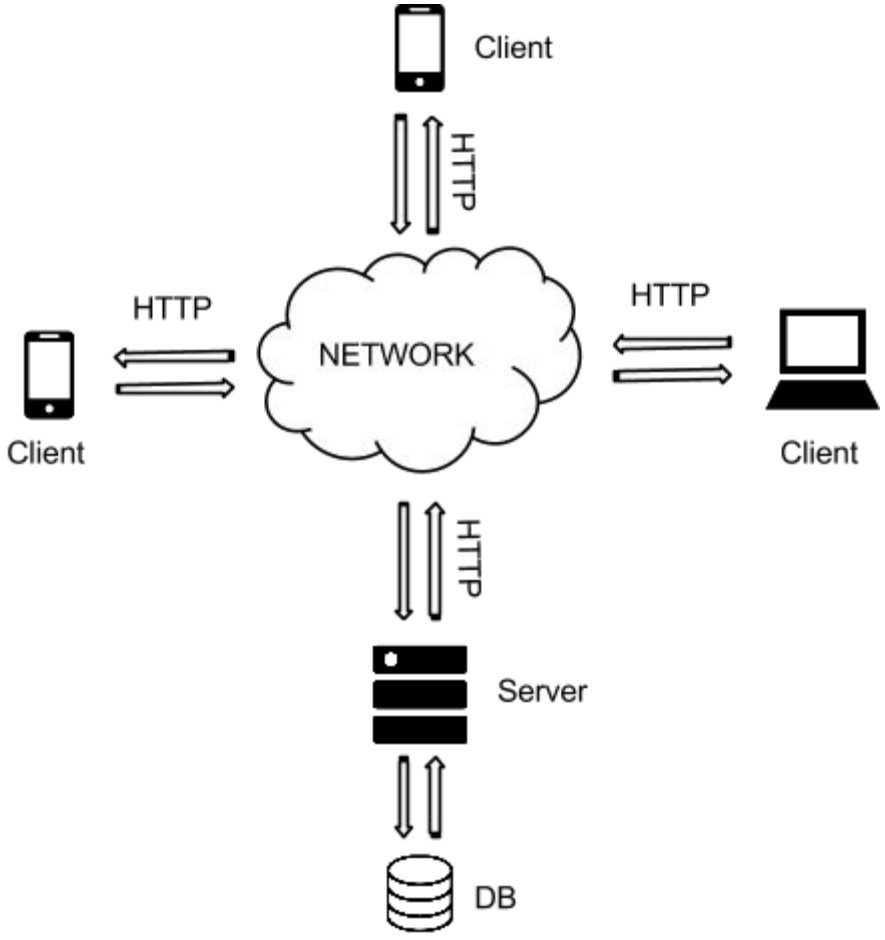
# Índice

# Introducción

Debido al auge de las aplicaciones para almacenamiento remoto (cloud storage) y su integración con la web y dispositivos móviles, la empresa AppMaker© nos ha encargado el desarrollo de una aplicación de almacenamiento remoto.
Esta empresa busca posicionarse en el mercado de aplicaciones de almacenamientos en la nube buscando conectar esta aplicación con sus otras aplicaciones que poseen gran cantidad de usuarios.

La aplicación constará de dos componentes:

- Un servidor, el cual será el responsable del procesamiento, almacenamiento y el delivery de las solicitudes de los usuarios
- Un cliente, el cual será responsable de visualizar el almacenamiento remoto asociado al usuario que se encuentra conectado



# Aplicaciones Requeridas

## Servidor

Se trata de una aplicación linux por consola destinada a mantenerse en ejecución por períodos prolongados de tiempo. Esta aplicación debe brindar una interfaz para la comunicación de los diferentes clientes que se conecten a la misma. Para la interfaz de comunicación se utilizará Restful API [1], que definirá la forma de las solicitudes y respuestas de los diferentes servicios que brindará el servidor.

### Servicio de autenticación

El servidor dispondrá de un servicio para la autenticación de los clientes. Este servicio consta de una solicitud de autenticación, que viene junto con las credenciales del usuario. La respuesta a la solicitud es un token (identificador) de la sesión del usuario.

### Servicio de registración de usuarios

El servidor debe disponer de un servicio que permita la registración de nuevos usuarios en el sistema.

### Servicio de administración de archivos remotos

El servidor debe disponer de un servicio que permita la administración de los archivos remotos. Por administración se entiende como la posibilidad de realizar alta, baja o modificaciones sobre los archivos pertenecientes a un usuario.

#### Alta

El pedido de alta de un archivo remoto consiste en la subida al servidor del archivo físico para que el mismo pueda almacenarlo. Este pedido de alta debe estar acompañado por metadatos asociado al archivo para permitir una correcta administración del mismo.  Estos metadatos deben al menos contener:

- Nombre archivo
- Extensión
- Última fecha de modificación (dato no completado al momento de alta)
- Usuario que realizó la modificación (dato no completado al momento de alta)
- Etiquetas
- Propietario (dato no completado al momento de alta)

#### Baja

El pedido de baja de un archivo remoto realizará una baja lógica del archivo en el servidor. Esto quiere decir que no eliminará el mismo del servidor sino que sólo desactiva al mismo (elimina sus metadatos) para que este no pueda ser accesible desde el cliente.

#### Actualización

El pedido de actualización del archivo posee dos tipos de actualizaciones: lógica y física. La actualización lógica hace referencia a poder modificar los metadatos asociados al archivo. Por otro lado, la actualización física debe actualizar el archivo en el servidor.

**Consulta**
El pedido de consulta sobre un archivo debe devolver los metadatos asociados.

**Descarga**
El pedido de descarga de un archivo remoto consiste en la descarga física del mismo.

### Servicio de búsqueda de archivo remotos

El servidor debe permitir realizar búsquedas sobre los metadatos de los archivos remotos. Estas búsquedas deben admitir las búsqueda por los diferentes metadatos:

- Búsqueda por etiquetas
- Búsqueda por extensión
- Búsqueda por nombre
- Búsqueda por propietario de archivo

### Servicio de papelera y recuperación de borrado

El servidor  debe permitir poder consultar los archivos que fueron borrados (colocados en la papelera), dando la posibilidad de recuperarlos mediante una operación de recupero de archivo.

### Servicio para compartir archivos

El servidor debe permitir poder compartir archivos entre clientes. Para esto el cliente debe especificar el archivo a compartir y la lista de usuarios con los cuales desea compartir. Todos los usuarios que puedan acceder al archivo tienen los mismos permisos sobre el mismo.

### Almacenamiento de archivos

El servidor debe persistir los archivos remotos que son solicitados por el usuario.

### Cuotas por usuario

El servidor debe permitir administrar el manejo quotas por usuario para controlar la capacidad de archivos a almacenar por este. La cuota asignada al usuario formará parte de la información del perfil de usuario.

### Servicio de administración de perfil de usuario

El servidor proveerá un servicio en donde el usuario podrá actualizar los datos asociados a su perfil, y los demás usuarios podrán consultar su estado. Estos datos consisten en:

- Nombre
- Email
- Foto de perfil
- Última ubicación del usuario: esta ubicación del usuario se actualizará cuando el mismo realice actualización sobre los archivos sobre los cuales tiene acceso

### Log

El servidor debe constar con un sistema de log en donde se registren los eventos que se generen durante la ejecución del mismo. El sistema de log debe permitir configurar el nível de los eventos que desean registrar. Estos níveles son:

| Nivel | Condiciones |
|---|---|
| Error | Condición de falla catastrófica, el sistema no puede funcionar. (criterio de las 2 a.m.)<br>Condición que haga que la aplicación no pueda ejecutar una funcionalidad.<br>Ejemplo:<br>No es posible conectarse con la base de datos |
| Warn | Cualquier condición anómala que afecte el funcionamiento del sistema, pero no impida la funcionalidad básica<br>Ejemplos:<br>Uso de APIs deprecadas<br>Mal uso de APIs |
| Info | Cualquier acción correspondiente a un caso de uso iniciada por el usuario o el sistema.<br>Información que permita trazar el historial de las entidades.<br>Ejemplos:<br>Conexión a la base de datos exitosa<br>Conexión de nuevo cliente |
| Debug | Información de contexto que permita resolver un problema técnico. Debe ser útil incluso sin el código fuente<br>Ejemplo:<br>Datos de login para la DB |
| Trace | Seguimiento de valores de variables, condiciones del soft. En la mayoría de los casos no se encuentra la salida estándar disponible para la impresión de condiciones del software. |

## Cliente

El sistema cliente posee una interfaz gráfica que permitirá visualizar el almacenamiento remoto del usuario que se encuentra utilizando la aplicación.

### Autentificación

Para que el cliente pueda enviar y recibir conversaciones el mismo debe autentificarse con el servidor. Para esto el cliente debe permitir al usuario ingresar usuario y contraseña. Se deberá diseñar una pantalla de ingreso a la aplicación.

### Registración

En caso de que la persona que desee utilizar la aplicación no tenga un usuario registrado la aplicación debe permitir poder registrar un usuario nuevo.

### Interfaz gráfica

- Visualización de Drive
- Administración de carpetas y archivos

**Visualización de Drive**

La aplicación debe permitir visualizar el directorio compartido (Drive) asociado al usuario. Dentro de este directorio se deben visualizar todos los archivos sobre los cual el usuario tiene acceso (los propios y los compartidos por otro usuario).
Dentro de la visualización se permitirá:

- Agregar nuevo archivo
- Crear una nueva carpeta
- Actualizar un archivo existente
- Eliminar un archivo/carpeta

**Administración de carpetas**

La aplicación debe permitir la administración sobre las carpetas y archivos del Drive. Las tareas de administración son:

- Compartir archivo/carpeta
- Cambiar metadatos asociados al archivo carpeta

### Log

El cliente debe constar con un sistema de log en donde registro los eventos que se generen durante la ejecución del mismo. El criterio de los niveles de log a utilizar es el mismo definido para la aplicación servidor.

### Instalación de aplicación

La instalación de la aplicación cliente debe ser un proceso simple y que ayude a un usuario inexperto a poder realizar la instalación de la misma sin mayores inconvenientes.

## Despliegue servidor

Para poder realizar una correcta administración de las dependencias que requiera la aplicación servidor, y para poder simplificar el proceso de despliegue (deploy) del mismo. Se deberá utilizar Docker http://docker.io/ como herramienta.

## Documentación

Es condición necesaria para la aprobación del trabajo práctico la entrega de una documentación formal de carácter profesional.

## Pruebas

El desarrollo de la aplicación se deberá adaptar a los estándares de calidad utilizados por AppMaker©. Dentro de estos estándares se encuentran:

- Pruebas unitarias [3]
- Métricas (ej. code coverage) [4]
- Respetar estándar para estilo de codificación

## Integración continúa

Para poder realizar una integración de todos los componentes del servidor debe utilizarse un servidor de integración continua para poder realizar una integración automática de estos componentes. En caso de utilizarse GitHub puede utilizarse Travis CI (https://travis-ci.org/)

## Restricciones

- El servidor debe ser desarrollado en C/C++. El servidor debe soportar múltiples conexiones en simultáneo.
- Para la compilación de la aplicación servidor se deberá utilizar CMake [5]
- La aplicación debe desarrollarse en Java utilizando el SDK de Android [6]
- Para la documentación técnica del trabajo práctico se utilizará Sphinx [7]
- Para el almacenamiento de la información del servidor (ej. conversaciones, cuentas de usuarios) se deberá utilizar una base de dato de tipo clave-valor de tipo NoSQL[8]. Se recomienda utilizar rocksdb. [9]

## Referencias

- [0] https://www.android.com/
- [1] http://en.wikipedia.org/wiki/Representational_state_transfer
- [2] http://developer.android.com/reference/android/location/LocationManager.html
- [3] http://en.wikipedia.org/wiki/Unit_testing
- [4] http://en.wikipedia.org/wiki/Code_coverage
- [5] http://www.cmake.org/
- [6] http://developer.android.com/sdk/index.html
- [7] http://sphinx-doc.org/
- [8] http://es.wikipedia.org/wiki/Javadoc
- [9] http://www.stack.nl/~dimitri/doxygen/
- [10] http://en.wikipedia.org/wiki/NoSQL
- [11] http://rocksdb.org/
- [12] http://es.wikipedia.org/wiki/Git