Práctica

Se desea crear un servidor de ficheros que mediante conexiones TCP los usuarios puedan subir archivos de pequeño tamaño y descargarlos desde su cuenta. Se deberá escribir también un cliente que por línea de comandos se podrá comunicar con el servidor. Las operaciones básicas que podrá solicitar el cliente al servidor son:

- *listado*: Saca una lista de los archivos en el servidor y el número de bytes disponibles.
- *subir*: Sube un archivo al servidor.
- *descargar*: Descarga un archivo del servidor y lo almacena en la carpeta en la que se esté ejecutando el cliente.
- *borrar*: Borra un archivo del servidor.

La comunicación se hará por línea de comandos. Una sesión del cliente podría ser la siguiente:

```
$ node cliente.js pepe 12345 subir archivo.txt
Archivo subido correctamente: “archivo.txt”
$ node cliente.js pepe 12345 listado
Listado de archivos en el servidor:
archivo.txt 123 bytes
Espacio disponible 5230 bytes
$ node cliente.js pepe 12345 descargar archivo.txt
Archivo descargado
$ node cliente.js pepe 12345 borrar archivo.txt
Archivo borrado del servidor
$ node cliente.js pepe 12345 listado
Listado de archivos en el servidor:
Espacio disponible 5353 bytes
```

Como se puede ver en cada comando se debe indicar el usuario y la contraseña del mismo. A continuación la operación a realizar. Según la operación habrá que añadir el nombre del archivo a subir, borrar o descargar.

Para no complicar el diseño de la aplicación se tendrán las siguientes limitaciones en el servidor:

1. Los usuarios, contraseñas y los archivos que han subidos, se almacenarán en un archivo json en el servidor. De cada archivo se guardarán su nombre y contenido. Por ejemplo:

```
[
  {
    'usuario': 'pepe',
    'password': '12345',
    'cuota': 5353,
    'archivos': [
      {
        'nombre': 'archivo1.txt',
        'contenido': 'Contenidos del archivo1'
      },
      {
        'nombre': 'archivo2.txt',
        'contenido': 'Contenidos del archivo2'
      }
    ]
  },
  {
```

```
    'usuario': 'lolo',
    'password': '12345',
    'cuota': 10000,
    'archivos': [
      {
        'nombre': 'archivo1.txt',
        'contenido': 'Contenidos del archivo1'
      },
      {
        'nombre': 'archivo2.txt',
        'contenido': 'Contenidos del archivo2'
      }
    ]
  }
]
```

2. El servidor leerá el archivo JSON al iniciarse y lo volverá a guardar cada vez que un usuario suba o borre un archivo.
3. El tamaño de los archivos se calcularán en el servidor a partir de la longitud de la cadena de texto que se reciba.
4. Se podrá indicar la “cuota” de cada usuario que será el tamaño en bytes del espacio que tiene el usuario para almacenar archivos. Si al subir un archivo, se supera la cuota, el servidor no debe almacenar un archivo y mandar un mensaje de error al cliente.
5. Cuando el cliente solicite al servidor el listado de archivos, el servidor calculará el tamaño de cada archivo a partir de la longitud de la cadena de texto que esté almacenada en el servidor. También calculará el espacio que le queda disponible al cliente.
6. La comunicación entre cliente y servidor será mediante conexiones TCP en las que se enviarán mensajes en formato JSON entre cliente y servidor. Los mensajes pueden ser textos en JSON. Por ejemplo, el mensaje JSON que envía el cliente para subir un archivo podría ser de la siguiente forma:

```
{
      ‘usuario’: ‘pepe’, ‘password’: ‘12345’, ‘accion’: ‘subir’,
      ‘archivo’: ‘ejemplo.txt’, ‘contenido’: ‘En un lugar de La Mancha...’
}
```

   La respuesta del servidor, podría ser:

```
{ ‘respuesta’: ‘Archivo subido correctamente’}
```

7. El servidor responderá al cliente con mensajes de texto también en JSON.
8. En el caso de que las credenciales sean incorrectas el servidor devolverá un mensaje de “Acceso denegado” al cliente y el cliente deberá mostrarlo.
9. Cuando se solicite al cliente subir un archivo, lo deberá leer desde el disco y enviarlo al servidor.
10. Cuando se solicite al cliente bajar un archivo, lo deberá descargar desde el servidor y guardarlo con el mismo nombre en el disco.
11. Para simplificar, los nombres de archivo que se le pasen al cliente no podrán ser rutas complejas, sólo el nombre del archivo y el archivo debe estar en la misma carpeta que el cliente a la hora de hacer el envío.

Se valorará:

1. (4 ptos) Que el servidor haga correctamente las 4 operaciones descritas, con las limitaciones indicadas.
2. (4 ptos) Que el cliente haga correctamente las 4 operaciones descritas, con las limitaciones indicadas.
3. (0,5 ptos) El servidor debe almacenar el archivo JSON cada vez que se suba o borre un archivo usando asincronía.
4. (0,5 ptos) Se deben tener los usuarios “pepe” y “lolo” en el servidor con contraseña “12345”.
5. (1 pto) Se debe tener un proyecto en NodeJS para el servidor y un comando en el proyecto de forma que se ejecuten test que comprueben las operaciones al ejecutar los comandos:

```
$ npm run test-subir
$ npm run test-bajar
$ npm run test-listado
$ npm run test-borrar
```