Objetivo:
Implementar un sistema BitTorrent que permita compartir archivos entre sus usuarios, pudiendo obtener los mismos de distintas fuentes en simultáneo.

Alcance:
- Aplicación cliente BitTorrent, con su interfaz.
- Middleware BitTorrent, con protocolos propios de BitTorrent.
- Implementación de sistemas de premios y castigos propio, respetando los protocolos de BitTorrent.

Actores/roles :

Principales:
- Administrador: Se encarga de poner en marcha el sistema.

Secundarios/importantes: *Leecher*, *Seeder*
- *Leecher*: Es el usuario que desea realizar la descarga de un archivo leyendo el contenido del archivo ".torrent" y descargando el mismo por partes desde *Seeders* o *Leechers*. También tiene la responsabilidad de enviar partes del archivo ya descargados a otros *Leechers* que lo requieran.
- *Seeder*: Es el usuario encargado de poner a disposición de los *seeders* y/o *Leechers*, según corresponda, la totalidad de las partes de un archivo asociado a un ".torrent".
- Técnico de Mantenimiento: En caso de errores en el sistema, es a quien se le avise para que éste tome medidas que el sistema en sí no puede.

Restricciones
- La obtención del archivo .torrent no está dentro del alcance del sistema.
- Se debe identificar a cada archivo de forma única.
- Un archivo debe separarse en paquetes de tamaño fijo, con número de paquete únicos.

Requerimientos

<u>Funcionales de Actores</u>

Seeder y Leecher
- Un *Seeder* o *Leecher* podrá abrir la aplicación.
- Un *Seeder* o *Leecher* podrán cerrar la aplicación. Ésta deberá poder realizar un cierre prolijo.
- Un *Seeder* o *Leecher* debe poder configurar la prioridad de subida (o *uploading*) de un archivo.
- Un *Seeder* o *Leecher* puede ver estadísticas de la sesión actual y sesiones anteriores: cantidad de datos descargados, cantidad de datos compartidos.

- Un *Seeder* o *Leecher* podrá volver a abrir la aplicación.
- Un *Seeder* o *Leecher* puede remover un torrent que ya no desee descargar o compartir.

Seeder

- Un *Seeder* debe poder ser fuente de un archivo nuevo (ser *Seeder original*). *Seeder* puede obtener el archivo .torrent que se corresponde con este nuevo archivo.
- Un *Seeder* debe poder dejar de ser fuente de un archivo, ya sea momentáneamente ,por el cierre de la aplicación, o *permanentemente*, por eliminar el archivo .torrent de la aplicación.
- Un *Seeder* debe poder volver a ser fuente de un archivo (siempre y cuando lo siga teniendo), ya sea volviendo a abrir la aplicación (la había cerrado), o volviendo a cargar el archivo .torrent a la aplicación.

Leecher

- Un *Leecher* puede, dado un archivo .torrent, obtener el archivo completo o parte del mismo, siempre que se encuentre disponible.
- Un *Leecher* podrá descargar contenido, y, a la vez, compartir contenido que ya haya obtenido.
- Un *Leecher* puede descargar una parte (o paquete) de un archivo, mediante una identificación única de la misma.
- Un *Leecher* debe poder ver información sobre los usuarios que ya descargaron el archivo previamente.
- Un *Leecher* debe poder ver información sobre los usuarios que se encuentran descargando el archivo actualmente.
- Un *Leecher* debe poder conocer las unidades del archivo que ya han sido obtenidas; de forma tal de reconocer las unidades faltantes.
- Un *Leecher* debe poder, siempre que sea posible, descargar de más de una única fuente (sea *leecher* o *Seeder*)
- Un *Leecher* debe poder, siempre que sea posible, utilizar tanto *Seeder*s como *Leechers* para la descarga de un archivo.
- Un *Leecher* puede ver una estadística de los aportes y descargas realizados, para cada archivo, y mantener un historial del total aportado y descargado durante el último período de tiempo (constante) de uso de la aplicación, para poder implementar el sistema de premios y castigos[1].
- Un *Leecher* puede ver el progreso de la descarga de un archivo (porcentaje ya descargado, tiempo restante estimado a partir de la velocidad observada, identificación de los paquetes ya descargados).

[1] No debe implementarse el sistema de premios y castigos sobre el global del historial, puesto que si un cliente primero aporta mucho, y luego decide no aportar, eventualmente también debe ser castigado.

- Un *Leecher* debe poder dejar de descargar un archivo, ya sea momentáneamente, por el cierre de la aplicación o pausa de la descarga, o *permanentemente*, eliminando el archivo .torrent de la aplicación.
- Un *Leecher* debe poder seguir con una descarga pendiente, desde el punto en el que la dejó (las partes que ya hubiere descargado deben ser tenidas en cuenta), ya sea que la hubiese detenido por cerrar la aplicación, pausado; e inclusive si vuelve a cargar un archivo .torrent que ya hubiese estado descargando previamente.
- Un *Leecher* siempre debe tener la posibilidad de descargar un archivo, si hay disponibilidad del mismo en la red, independientemente del "mal comportamiento" del mismo. Cabe aclarar que su velocidad descarga será correspondiente a su castigo.
- Un *Leecher* debe poder dar prioridad a las distintas descargas, pausar una descarga, o detener una descarga[2].
- Un Leecher descargando un archivo puede cooperar con otro *Leecher*.
- Un *Leecher* coopera cuando obtiene una unidad de otro *Leecher*.
- Un *Leecher* coopera cuando se ponen de acuerdo con otro *Leecher* en solicitar dos unidades distintas; para luego poder compartirlas.

<u>Funcionales del sistema</u>

- Un cierre prolijo incluye que el sistema entienda que el usuario con el archivo original pueda cerrar su aplicación. Si otros usuarios han descargado el archivo, se debe poder seguir compartiendo.
- Un cierre prolijo incluye el cierre de todas las conexiones establecidas entre los pares.
- El sistema debe penalizar malas actitudes (compartir pocos datos, compartirlos a baja velocidad) y premiar buenas actitudes.
- Los premios y penalizaciones deben afectar la velocidad de descarga del usuario. Un premio mejora la velocidad; una penalización la limita.
- Si un cliente no aporta al sistema, debe ser castigado reduciéndose su velocidad de descarga.
- Por el contrario, si un cliente aporta mucho al sistema, debe ser premiado, pudiendo tener una mayor velocidad de descarga.
- Se deberá tener consenso del nivel de comportamiento de un usuario.
- Los *Leechers* deberán consensuar las unidades a descargar.
- La caída de un cliente no debe impedir la descarga de un archivo, exceptuando casos excepcionales (se cayeron todos los clientes con el archivo en cuestión, se cayó todo el sistema, etc…).
- En el caso de caída del *tracker*, debe ser posible realizar la descarga de haber otras fuentes disponibles, aunque fuere a menor velocidad.
- No debe agotarse el ancho de banda de un *Seeder* por descargar uno o varios archivos de él, ya que también debe poder ser un *Leecher* para otros archivos.

[2] Al pausar una descarga, no se cierran las conexiones con los pares de los cuales se realiza la descarga (permite retomar mucho más fácilmente la descarga). Al detener la descarga, se cierran todas las conexiones.

No Funcionales

- Un cliente debe poder ser *Leecher* y *Seeder* simultáneamente, para distintos archivos.
- Se debe garantizar integridad de los datos descargados.
- El sistema debe ser de interfaz abierta.
- La interfaz de usuario debe ser lo más simple posible.
- Ni el sistema, ni los desarrolladores, ni los dueños del mismo son responsables por el contenido de los archivos compartidos, ya sea que violen derechos de autor o alguna ley de regulación informática.
- Mantener compatibilidad con los protocolos establecidos de BitTorrent.
- El sistema debe tener una confiabilidad del 99.9%.
- El sistema debe poder escalar a una cantidad N de equipos. (pensar cuánto puede valer N)

Entidades:

- Usuario: que tiene roles por cada archivo:
  - Leecher
  - Seeder
- Metadata de Archivo.
- Archivo (propiamente dicho).
- Archivo siendo descargado (Archivo Lógico).
- Parte de un Archivo.
- Técnico de Mantenimiento.

Caso de Uso:

Nombre: Descargar Archivo.

Actores:

Primario: Administrador

Secundario: Leecher, Seeder original, Seeder (incluye al original).

En paralelo se ejecutan una indefinida cantidad de seeders y leechers.

0. Administrador inicia el sistema por primera vez.
1. **Seeder original:**
   1.1. El seeder original pide un tracker al sistema.
   1.1. Crea archivo .torrent con dicho tracker.
   1.2. El seeder original se transforma en el primer seeder del archivo (ir a 2.)
2. **Seeder**:
   2.1. El seeder avisa (se registra) al sistema que será seeder (fuente) del archivo.
   2.2. Repetir para cada pedido de descarga de un leecher:
      2.2.1. El leecher pide parte n°x del archivo.
      2.2.2. El seeder envía la parte n°x del archivo al leecher.
   2.3. Fin Repetir (hay que poner que deja de ser seeder si se cierra el programa?)
3. **Leecher:**
   3.1. El leecher avisa (se registra) al sistema del Archivo.
   3.2. El leecher le pide al sistema e listado de leechers del Archivo.

3.3. El leecher le pide a los otros leechers las listas de partes del archivo que ya han descargado.

3.4. Mientras falte alguna parte del archivo por descargar, hacer en paralelo:

3.4.1. Mientras falta alguna parte del archivo por descargar:

3.4.1.1. El leecher envía pedido de descarga de parte n°x del archivo a leecher (que tenga esa parte ya descargada) o seeder.

3.4.1.2. El leecher descarga la parte n°x del archivo de la fuente de 3.4.1.1.

3.4.1.3. Si la parte del archivo descargada no tiene fallas:

3.4.1.3.1. El leecher guarda la parte descargada.

3.4.1.3.2. El leecher avisa al sistema y otros leechers que tiene dicha parte de archivo disponible para ser descargada.

3.4.1.4. Sino:

3.4.1.4.1. El leecher descarta la parte del archivo descargada (la volverá a pedir en la iteración siguiente).

3.4.1.5 Fin si

3.4.2. Fin Mientras

3.4.3. Repetir para cada pedido de descarga, mientras falten partes del archivo por descargar:

3.4.3.1. Otro leecher realiza pedido de descarga de parte n°x del archivo.

3.4.3.2. Si el leecher fuente tiene la parte n°x del archivo descargada:

3.4.3.2.1. El leecher fuente envia la parte n°x del archivo al otro leecher.

3.4.3.3. Sino:

3.4.3.3.1. El leecher fuente le envía mensaje de error al otro leecher.

3.4.3.4. Fin Si.

3.4.4 Fin Repetir.

3.5. Una vez que el leecher ha terminado de descargar todo el archivo (todas sus partes):

3.5.1. El leecher avisa al sistema que dejó de ser leecher del archivo.

3.5.2. El leecher se transforma en seeder del archivo (ir a 2.)

4. Técnico de Mantenimiento:

4.1. Repetir para cada mensaje de error recibido:

4.1.1. Registrar el mensaje de error.

4.1.2. **Ver si hay que agregar alguna acción más**

4.2. Fin Repetir

Diagrama de Casos de Uso

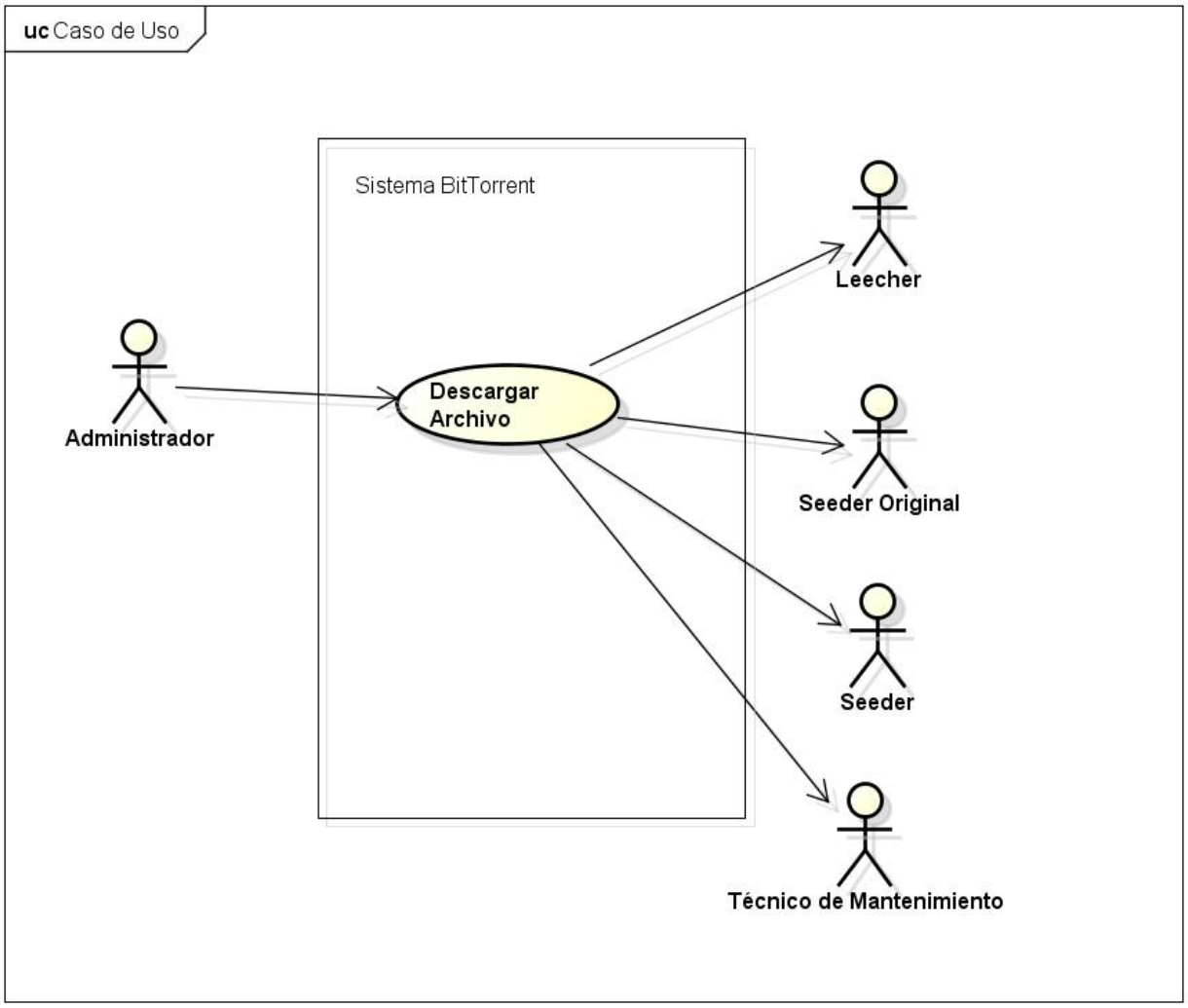

Diagrama de Clases Preliminar