Sincronización de ficheros entre usuarios

Grupo:
Mario Gordillo Ortiz
Adrian Alhama Carrasco

Ciclo:
Administració de Sistemes Informàtics en xarxa (ASIX)

Curso:
2017/2018

MIT License

Descripción del proyecto

Desarrollo de una aplicación web que usa el protocolo P2P para compartir ficheros desde un disco local y streamear el contenido con otras personas en diferentes salas evitando la necesidad de subir el fichero a ningún servidor. Los usuarios podrán comunicarse entre ellos haciendo uso de un chat de texto. Cada sala tendrá ciertos permisos y a los usuarios se les pueden asignar unos permisos específicos. En la sala se podrá compartir y reproducir ficheros de vídeo, audio, imágenes o de texto. También será posible reproducir vídeos de otros servicios como Youtube. En cada sala creada se mostrará una lista de reproducción que permitirá ir compartiendo los diferentes ficheros con los usuarios uno detrás de otro.

Abstract:

Development of web application that uses the P2P protocol to share files from a local disk and stream content with other people in different rooms without the need of uploading the file anywhere. Users can communicate with each other using a text chat. Each room will have certain permissions and users may has assigned specific permissions. In the room a user can share and play video, audio, images or text files and it is also possible to play videos from other services such as YouTube. Every created room there is a playlist that will allow to share different files with the users one after another.

Palabras clave:

web, torrent, stream, p2p, video, playlist, nodeJS, room, javascript, friends, player, peer, sync, users, mongoDB, playemJS, youtube, app.

Índice

1. INTRODUCCIÓN

1.1 DESCRIPCIÓN DEL PROYECTO

El proyecto trata de una aplicación web programada en Node JS que permitirá compartir cualquier tipo de fichero multimedia o documento de texto con otras personas sin necesidad de pasar por el proceso de subida y descarga del mismo. El objetivo es poder sincronizar el recurso a compartir para todos los usuarios de la misma manera, se podrá comunicar con los usuarios y será posible manejar las cosas a placer gracias a permisos que se ofrecen para configurar.

1.2 CARACTERÍSTICAS DEL PROYECTO

Las características de este proyecto son las siguientes:

- Aplicación Web en Node JS
- Uso de diferentes ficheros a compartir como MP3, MP4, MKV, PDF o TXT.
- Conexión entre clientes vía P2P (Peer 2 Peer).
- Creación de torrents a partir de un fichero.
- Salas públicas y privadas.
- Playlist para agregar los ficheros a una sala.
- Chat de texto y de voz entre los diferentes usuarios.
- Asignación de permisos a nivel de sala y usuarios.
- Personalización de los usuarios.
- Reproductor personalizado.

2. OBJETIVOS

El objetivo principal del proyecto es poder compartir archivos locales con otras personas usando una aplicación web para ser compatible con la mayoría de dispositivos. Lo más importante es la sincronización entre las diferentes personas a la hora de visualizar el recurso al mismo tiempo, ya que esta es la finalidad del proyecto.

2.1 OBJETIVOS PRINCIPALES

Los objetivos principales del proyecto, es decir, la base del proyecto son:

- Aprender el lenguaje de programación “JavaScript”.
- Implementar servidor con el uso de NodeJS.
- Aprender el funcionamiento de un cliente Torrent.
- Analizar la creación de Listas de Reproducción.
- Analizar la creación y el manejo de ficheros Torrent.
- Investigar la implementación de un chat de Texto.
- Adquirir más conocimientos durante el desarrollo del proyecto.

2.2 OBJETIVOS SECUNDARIOS

Estos objetivos forman parte de la ampliación del proyecto:

- Aprender a introducir otros servicios por HTTP (Youtube, Vimeo...)
- Investigar la creación de un chat de Voz (voIP).
- Analizar cómo fraccionar un fichero Torrent.

3. REQUISITOS GENERALES

3.1 REQUISITOS PREVIOS

Es importante tener conocimientos mínimos de programación orientada a objetos y saber cómo funciona el lenguaje de programación JavaScript que es el que se usará en el proyecto.

3.2 REQUISITOS DE SOFTWARE

Para poder llevar a cabo el proyecto, se recomienda el uso de un IDE. Hay diferentes IDEs que se pueden usar para programar esta aplicación con código JavaScript como Visual Studio Code, Netbeans o Atom. El software escogido es el IDE de JetBrains que lleva como nombre “WebStorm”, intuitivo, con muchas características y el más cómodo para trabajar en equipo.

3.2.1 TECNOLOGÍAS

A continuación las tecnologías que se utilizarán durante todo el proyecto:

- P2P
- Lenguaje Node JS
- Socket.io
- Librería WebTorrent
- Librería PlayemJS
- Lenguaje Pug

Estas tecnologías han sido escogidas debido a que son las que mejor se adaptan para el desarrollo de la aplicación.

4. TEMPORIZACIÓN Y TAREAS

4.1 PLANIFICACIÓN DE TAREAS

Las tareas que se llevarán a cabo durante el desarrollo del proyecto son las siguientes:

1. Investigar tecnologías a utilizar.
2. Realizar comprobaciones y pruebas básicas.
3. Diseñar conceptualmente la aplicación
    - Diseñar la aplicación web
    - Diseñar el servidor
4. Implementar aplicación
    - Crear un servidor web
    - Implementar sistema de salas
    - Implementar comunicación entre clientes y servidor
    - Implementar streaming con el uso de protocolo P2P
    - Implementar sincronización entre clientes del contenido multimedia
    - Implementar Drag and Drop
    - Generar torrents a partir de ficheros (Magnet Links)
    - Definir diseño visual de la aplicación
        - Estructurar diseño
        - Aplicar diseño
    - Implementar el uso de chat de texto
    - Implementar playlists
    - Implementar usuarios
        - Guardar estado de los usuarios de la diferentes salas
        - Añadir permisos a los usuarios.
    - Añadir soporte para más tipos de ficheros (.pdf, .txt, .png, .jpg....)
    - Implementar base de datos MongoDB para monitorizar información
    - Implementar uso de enlaces de otros servicios (Youtube, etc)
    - Definir salas públicas y privadas
    - Integrar notificaciones en la aplicación.
    - Gestionar tracker para los archivos compartidos.
    - Añadir una capa de seguridad a la aplicación
5. Comprobar aplicación
6. Memoria del proyecto
    - Documentar la memoria
    - Revisar la versión definitiva de la memoria
    - Entregar memoria
7. Preparar la presentación

4.2 DESCRIPCIÓN DE LAS TAREAS PRINCIPALES

TAREA	DESCRIPCIÓN
Investigar tecnologías a utilizar.	Realizar una búsqueda de las principales tecnologías que es posible usar en la aplicación web que cumpla con todos los objetivos.
Realizar comprobaciones y pruebas básicas.	Crear diferentes tests de prueba con las tecnologías seleccionadas para comprobar que pueden ser útiles.
Diseñar conceptualmente la aplicación	Idear el funcionamiento del servidor que controlará la aplicación, y, de la propia aplicación web.
Implementar aplicación	Proceso de realización de la aplicación web con todo el conjunto de características que conlleva. También la implementación del servidor.
Comprobar aplicación	Probar constantemente la aplicación para ver que no existen errores.
Memoria del proyecto	Completar la documentación del proyecto que se irá haciendo durante el transcurso de todo el proyecto.
Preparar la presentación	Preparar los materiales necesarios para realizar la presentación del proyecto.

4.3 DIAGRAMA DE GANTT

El Diagrama de Gantt permite tener una planificación de todas las tareas que forman el proyecto, y, poder visualizar cuándo y quién ha realizado una tarea para así tener controlado los tiempos y repartir las tareas entre los integrantes de manera correspondiente.

En la siguiente imagen se muestran los recursos que participan en la realización de las tareas, es decir, los participantes que llevarán a cabo el proyecto.

Nombre	Nombre corto	Tipo	Grupo	Correo-e	Coste
Mario Gordillo Ortiz	MGO	Trabajo		mgordillo@elpuig.xeill.net	0
Adrian Alhama Carrasco	AAC	Trabajo		aalhama@elpuig.xeill.net	0

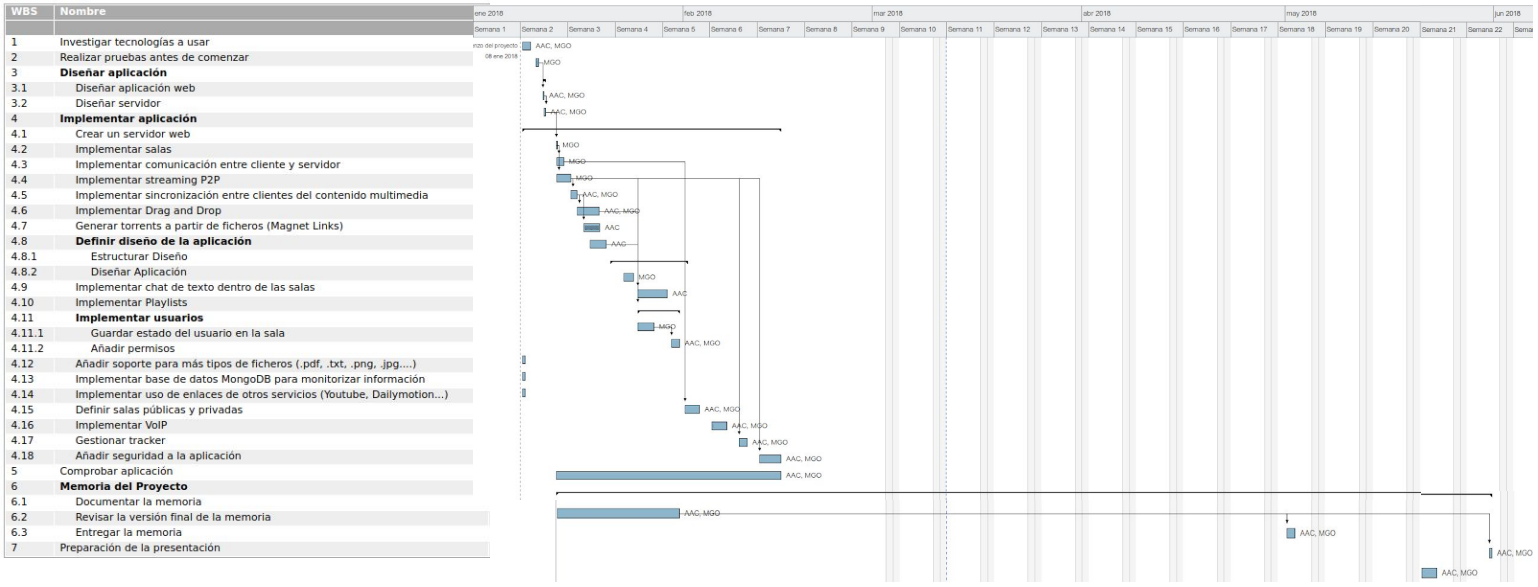
Por otra parte, existen las diferentes tareas y subtareas detalladas con la duración aproximada de cada una de ellas hasta la presentación del proyecto.

Investigar tecnologías a usar	3d
Realizar pruebas antes de comenzar	1d
Diseñar aplicación	**1d 7h**
Diseñar aplicación web	1d
Diseñar servidor	7h
Implementar aplicación	**66d 2h**
Crear un servidor web	2h 15...
Implementar salas	1d
Implementar comunicación entre cliente y servidor	2d
Implementar streaming P2P	2d
Implementar sincronización entre clientes del contenido multimedia	7d
Implementar Drag and Drop	3d
Generar torrents a partir de ficheros (Magnet Links)	3d
Definir diseño de la aplicación	**12d**
Implementar chat de texto dentro de las salas	2d
Implementar Playlists	5d
Implementar usuarios	**6d**
Guardar estado del usuario en la sala	3d
Añadir permisos	3d
Añadir soporte para más tipos de ficheros (.pdf, .txt, .png, .jpg....)	1d
Implementar base de datos MongoDB para monitorizar información	1d
Implementar uso de enlaces de otros servicios (Youtube, Dailymotion...)	1d
Definir salas públicas y privadas	5d
Implementar VoIP	5d
Gestionar tracker	3d
Añadir seguridad a la aplicación	7d
Comprobar aplicación	67d
Memoria del Proyecto	**41d 4h**
Documentar la memoria	36d 4h
Revisar la versión final de la memoria	3d
Entregar la memoria	2d
Preparación de la presentación	5d

El apartado más importante es mostrar la gráfica con las semanas y el tiempo que nos ocuparan las tareas hasta la presentación del proyecto. En la siguiente imagen tenemos la gráfica o el Diagrama de Gantt del proyecto.

Inicio: 8 enero, 2018
Fin: 31 mayo, 2018
Fecha de informe: 11 marzo, 2018

Gráfica Gantt



5. IMPLEMENTACIÓN

5.1 REQUISITOS DE LA APLICACIÓN

La aplicación para poder ejecutarse correctamente deberá tener instalado lo siguiente en la máquina dónde se ejecutará la aplicación (Servidor):

- Node.JS: Versión 7.5 o superior
  https://nodejs.org/es/download/package-manager/
- MongoDB:
  https://docs.mongodb.com/manual/administration/install-community/

Es importante tener la versión indicada de NodeJS o superior para no tener problemas a la hora de ejecutar la aplicación.

5.2 USO DE TECNOLOGÍAS

En la aplicación se hacen uso de muchas tecnologías simultáneamente, para poder conseguir la finalidad de la aplicación:

- **NodeJS:** se usa para para poder crear la aplicación web con JavaScript y permite usar evento en el lado del servidor con un gran rendimiento.
- **Socket.io:** se utiliza para que el servidor se comunique con los diferentes clientes que acceden a la aplicación y poder manejar los eventos en tiempo real.
- **Webtorrent:** se utiliza como cliente P2P para compartir ficheros entre los diferentes usuarios de una sala. (Peer To Peer)
- **PlayemJS:** se usa para poder compartir y reproducir vídeos de otros servicios, principalmente Youtube.
- **Pug:** este lenguaje se utiliza para poder definir la página web con HTML que usa la aplicación JavaScript.

5.3 FORMATOS SOPORTADOS

La aplicación podrá reconocer los formatos que se encuentran en la siguiente lista:

- **Vídeo**: mkv, mp4.
- **Audio:** mp3, flac.
- **Imagen:** png, jpeg, jpg.
- **Otros:** pdf, txt.

La idea principal es poder compartir los ficheros más usados y comunes con el resto de usuarios que pertenezcan a la misma sala. Por otra parte, los navegadores cuentan con limitaciones a la hora de poder interpretar ciertos ficheros. Esto quiere decir, que en el caso que el usuario comparta un archivo que no se reconozca, recibirá una notificación para tener constancia.

Los ficheros ".torrent" que se compartan con los usuarios, el tracker ha de ser compatible con la tecnología "Webtorrent" para que los usuarios se comuniquen sin problema a la hora de descargar el recurso.

5.4 OTROS SERVICIOS

Esta aplicación permite compartir y realizar streaming de otros tipos de servicios diferentes a archivos ".torrent". El principal servicio integrado es "Youtube", el cual permitirá compartir un enlace del mismo con los usuarios que se encuentran en la sala. Para ello se ha usado la librería "PlayemJS" que permite reproducir enlaces de otros servicios.

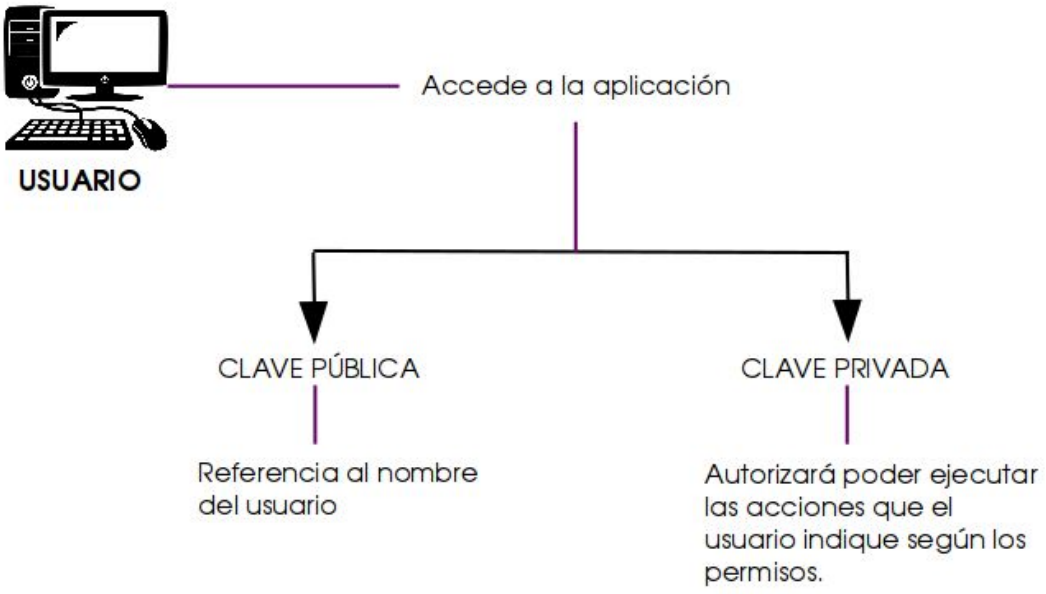
La base es muy similar a la usada con los ficheros ".torrent", los enlaces reproducidos serán sincronizados con el resto de usuarios de la sala. De esta manera se puede aumentar la lista de posibilidades que tiene el usuario de hacer uso de la aplicación.

5.5 CLAVE PÚBLICA Y PRIVADA

Al iniciar la aplicación desde el navegador, se creará un usuario en la base de datos del servidor que se estará ejecutando. Para poder identificar a cada usuario que se encuentra haciendo uso de la misma, cada usuario consta de una clave pública y privada.

-La clave pública: es la clave por la que se identificará el nombre de un usuario.

-La clave privada: es la clave que autorizará al usuario realizar las acciones cuando se encuentre internamente en una sala.



Esto resolverá grandes problemas de seguridad que pueden encontrarse, cómo poder identificarse como otro usuario de la misma sala que tiene permisos superiores.

Otros de los puntos fuertes es que cada nombre de usuario será único, es decir, que no se repetirán dentro del servidor.

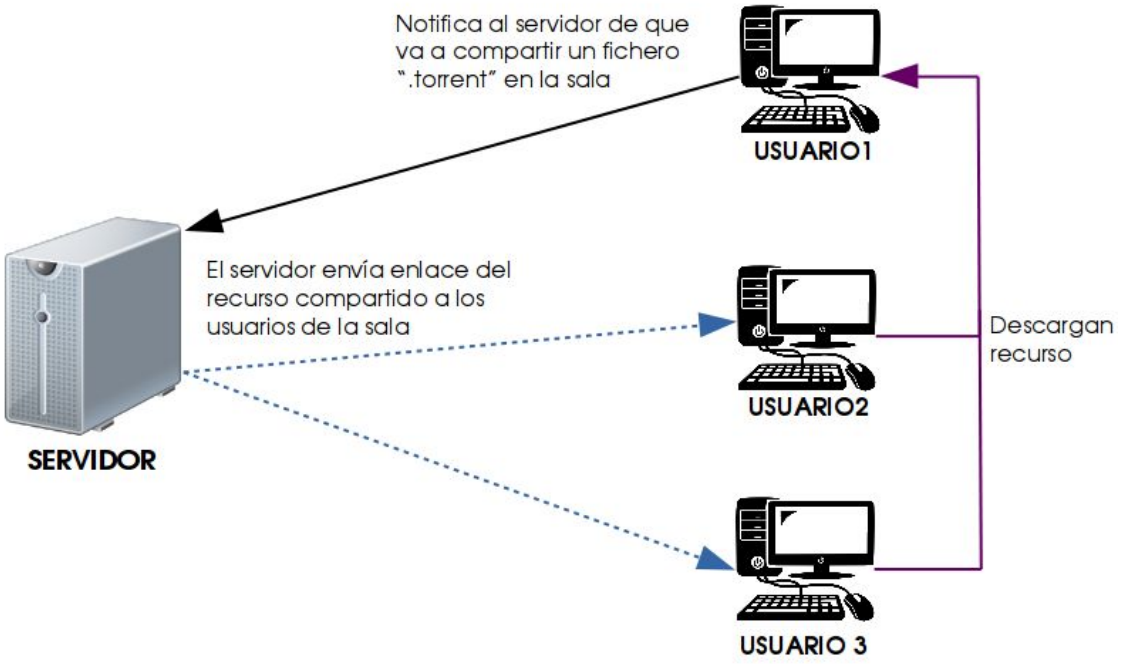
5.6 COMUNICACIÓN CLIENTE - SERVIDOR

Para emplear una comunicación entre los diferentes clientes que se conectan a la aplicación, y el servidor, hemos empleado uso de la librería "**socket.io**". Esta librería permite una comunicación en tiempo real entre los clientes y el servidor, esto permite una mayor flexibilidad y facilidad a la hora de comunicarse los diferentes clientes. La aplicación funciona de la siguiente manera:

-Los clientes de una misma sala, están comunicados entre sí.

-Los clientes que pertenecen a salas diferentes están incomunicados entre éstos.

La característica del servidor sería, ir agrupando a los diferentes usuarios por salas y controlar el contenido que se va compartiendo en la sala, para así, enviar un enlace del recurso compartido a los usuarios de la misma.

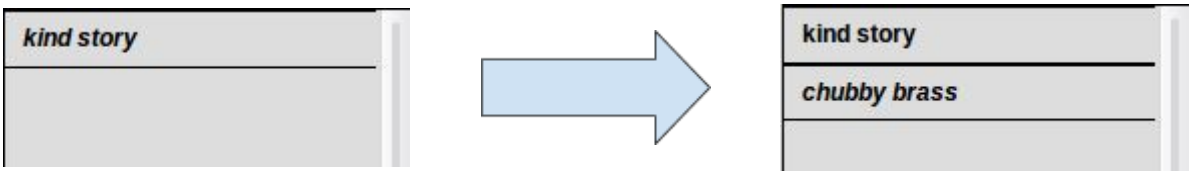


En el diagrama se puede ver que función tiene el servidor de la aplicación y los diversos clientes conectados en una misma sala.

Una vez se ha visto cómo funciona la interacción entre los clientes y el servidor, vamos a ver todas las comunicaciones que se encuentran en la aplicación. Esto se refiere todos los mensajes que emite el cliente al servidor y viceversa.

5.6.1 ACTUALIZAR LA LISTA DE USUARIOS

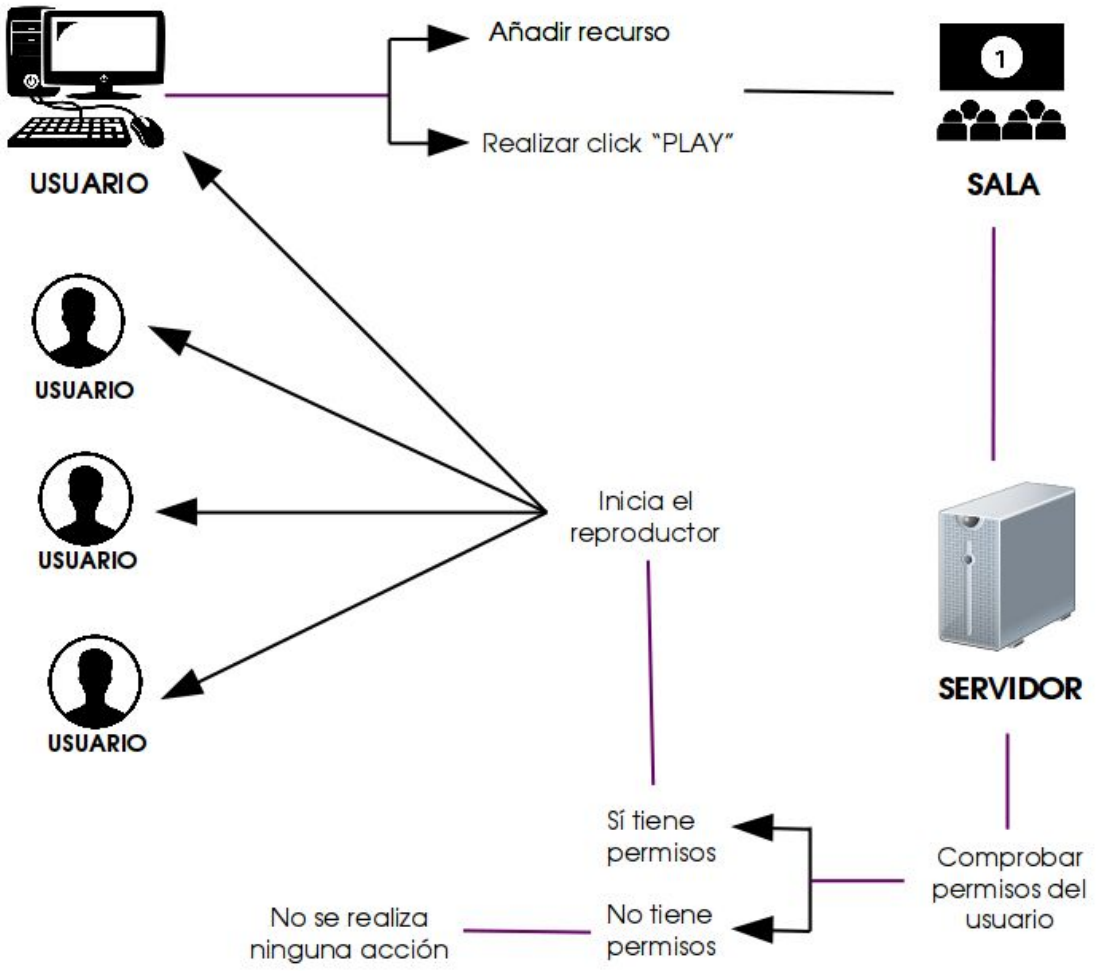
En el momento que al servidor perciba que la lista de usuarios ha sido modificada de una sala concreta, emitirá un mensaje a todos los clientes de esa sala, para que actualicen su lista de usuarios en la aplicación



En este contexto muestra que un usuario crea una sala, con lo cual solo hay un usuario en ésta. En el momento que otro usuario entra en nuestra sala, la lista de usuarios se actualiza para ambos, ya que ahora hay dos. Si hubiera más usuarios en dicha sala, la lista sería actualizada para todos ellos.

5.6.2 INICIAR EL REPRODUCTOR DE LA SALA

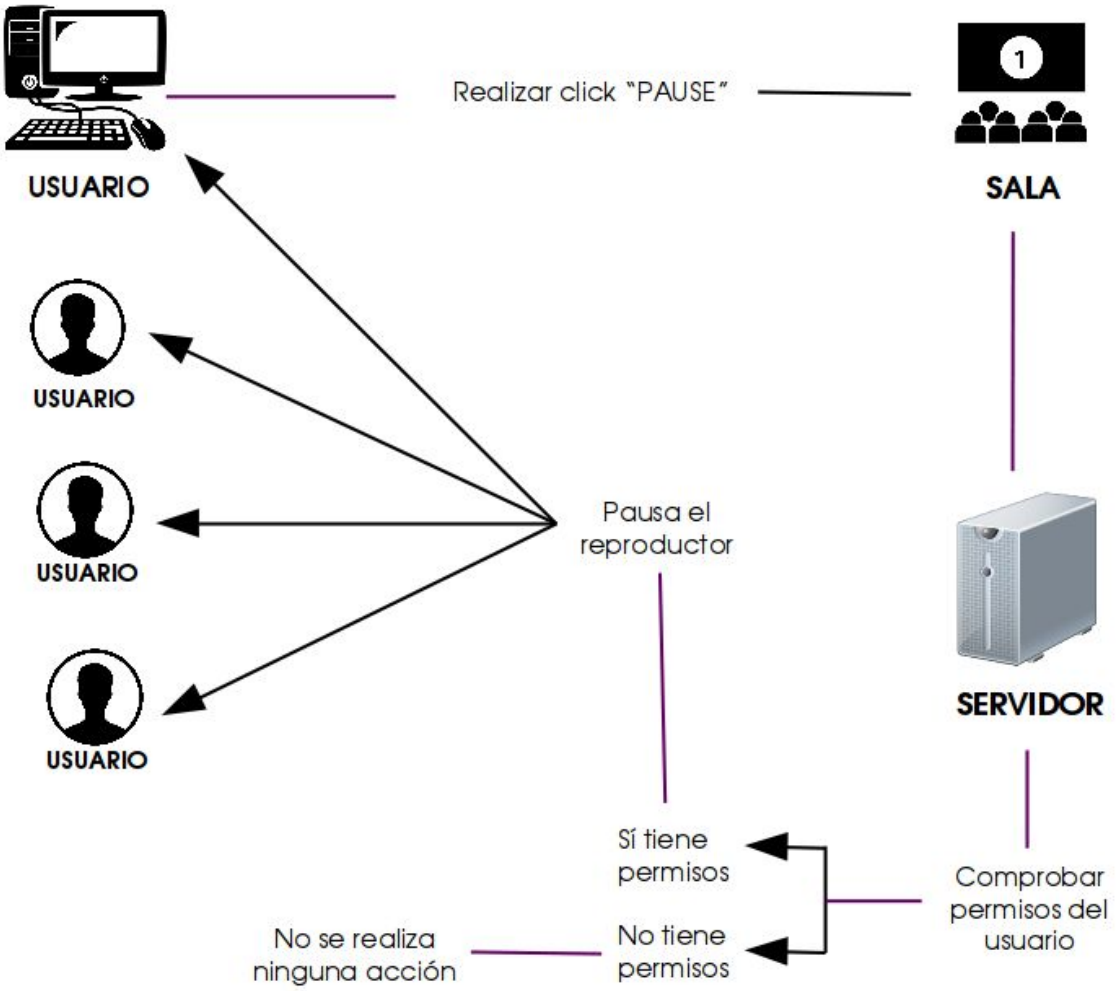
En el instante en el que un usuario de una sala concreta introduce un primer vídeo en la playlist o realiza la acción de hacer click en el botón “Play”, el reproductor se inicia. Cuando ese evento ocurra, el servidor enviará al resto de usuarios de la sala el mismo evento para que a todos se les inicie el reproductor. El usuario debe tener los permisos adecuados para poder llevar a cabo la acción, de no ser así no ocurrirá ningún tipo de acción.



5.6.3 PAUSAR EL REPRODUCTOR DE LA SALA

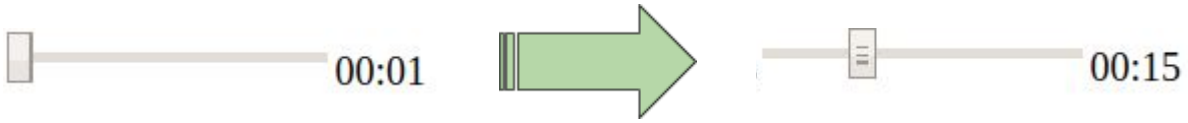
Este momento es el que un usuario de la sala pulsa el botón “Pause” y el recurso que se está reproduciendo se detiene en el minuto exacto. Por otra parte, el servidor envía al resto de usuarios de la sala, el evento de pausar el vídeo para que a todo el

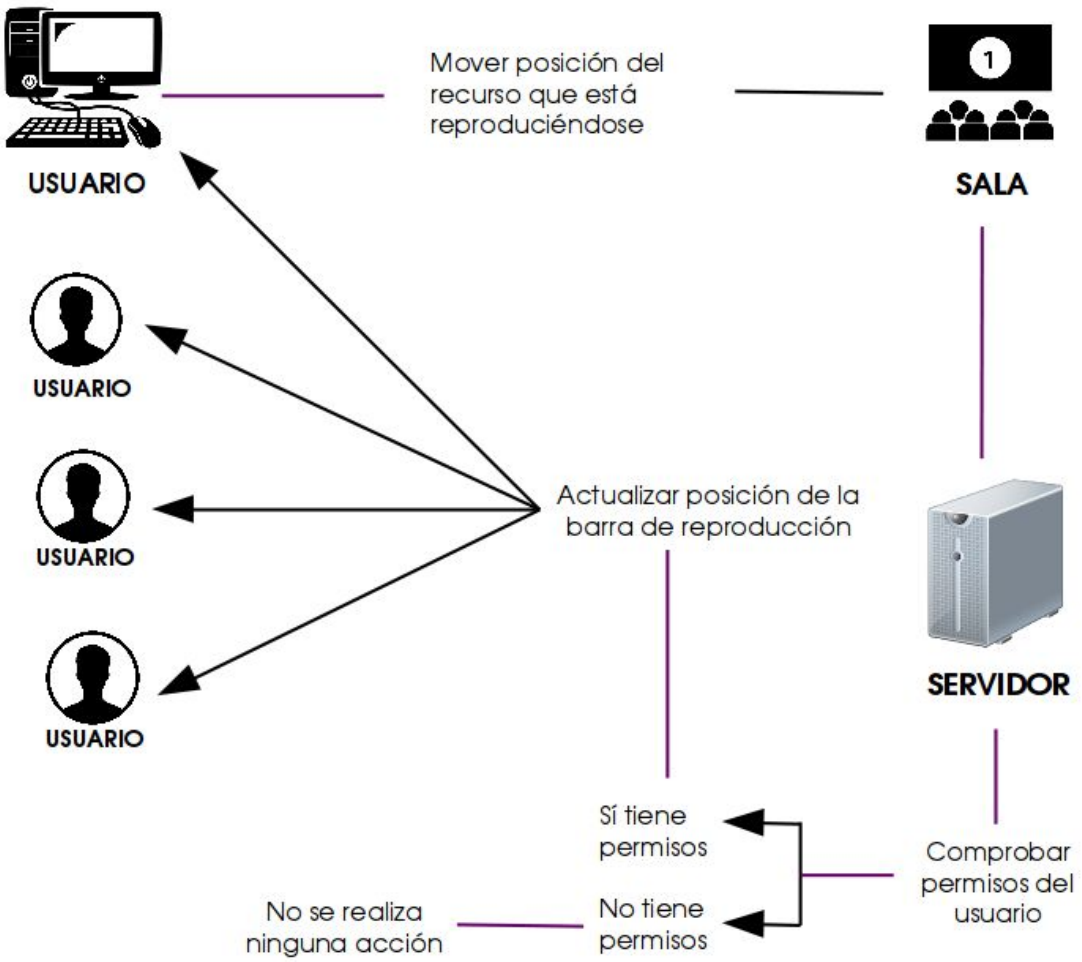
mundo se le detenga en el mismo momento y no perder la sincronización. El usuario debe tener los permisos adecuados para poder llevar a cabo la acción, de no ser así no ocurrirá ningún tipo de acción.



5.6.4 CAMBIAR POSICIÓN DEL REPRODUCTOR DE LA SALA

Cuando un usuario mueve con el cursor la posición del recurso que se está reproduciendo en el reproductor de la sala, el servidor envía al resto de usuarios que están en la sala la posición final del cambio. De esta manera si un usuario adelanta una canción o un vídeo, al resto también se les actualizará a esa posición sin perder la sincronización entre usuarios. El usuario debe tener los permisos adecuados para poder llevar a cabo la acción, de no ser así no ocurrirá ningún tipo de acción.





5.6.5 BLOQUEAR EL SERVIDOR

El servidor es bloqueado mientras todos los usuarios de la sala no tengan el recurso cargado listo para reproducir, ya que eso crearía una desincronización debido a que dependiendo del usuario puede cargarse antes o después. Entonces el objetivo es "Pausar" a los usuarios que tengan el recurso preparado, mientras quede alguno que no lo tenga. Una vez todos los usuarios tienen el recurso, el servidor deja de estar bloqueado, permite reproducir el recurso sin problema y hacer eso de los diferentes botones del reproductor para mantener a los usuarios de manera uniforme.

5.6.6 REPRODUCIR RECURSO AL USUARIO

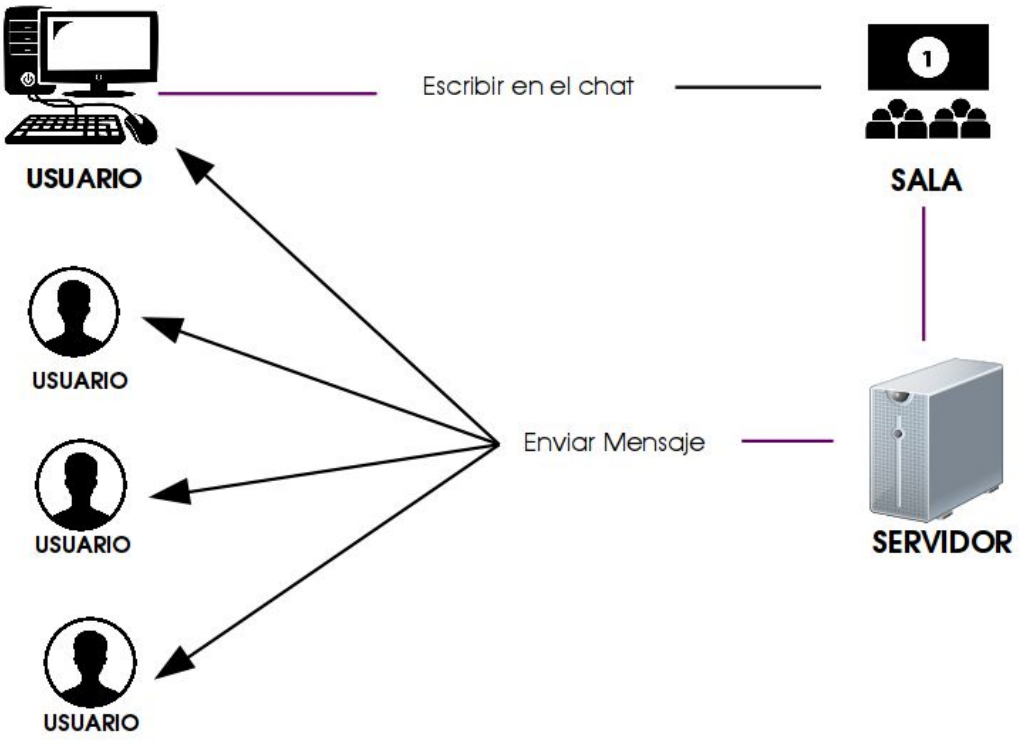
Cuando se dropea un fichero o en el momento que se añade la URI del recurso a compartir en una sala, se realiza una comprobación del tipo de fichero, es decir, si forma parte de un "Magnet Link", es un fichero ".torrent" o forma parte de otro servicio como puede ser "Youtube".

De esta manera el servidor enviará un enlace del fichero y el cliente se encargará de comprobar que tipo de fichero es y comenzará a reproducir el recurso.

5.6.7 INTERCAMBIAR MENSAJES EN EL CHAT DE LA SALA

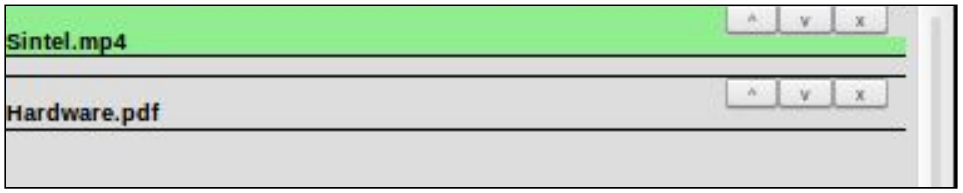
En una sala, los usuarios tienen la posibilidad de poder comunicarse con el resto de personas que están en la sala, haciendo uso de un chat de texto. El usuario se encarga de escribir el mensaje que desea enviar, y el servidor envía ese mensaje a todos los participantes que están en la sala, incluyendo al emisor.
Esto hace que el chat funcione correctamente, ya que sino los mensajes no serían mostrados a ninguno de los usuarios.

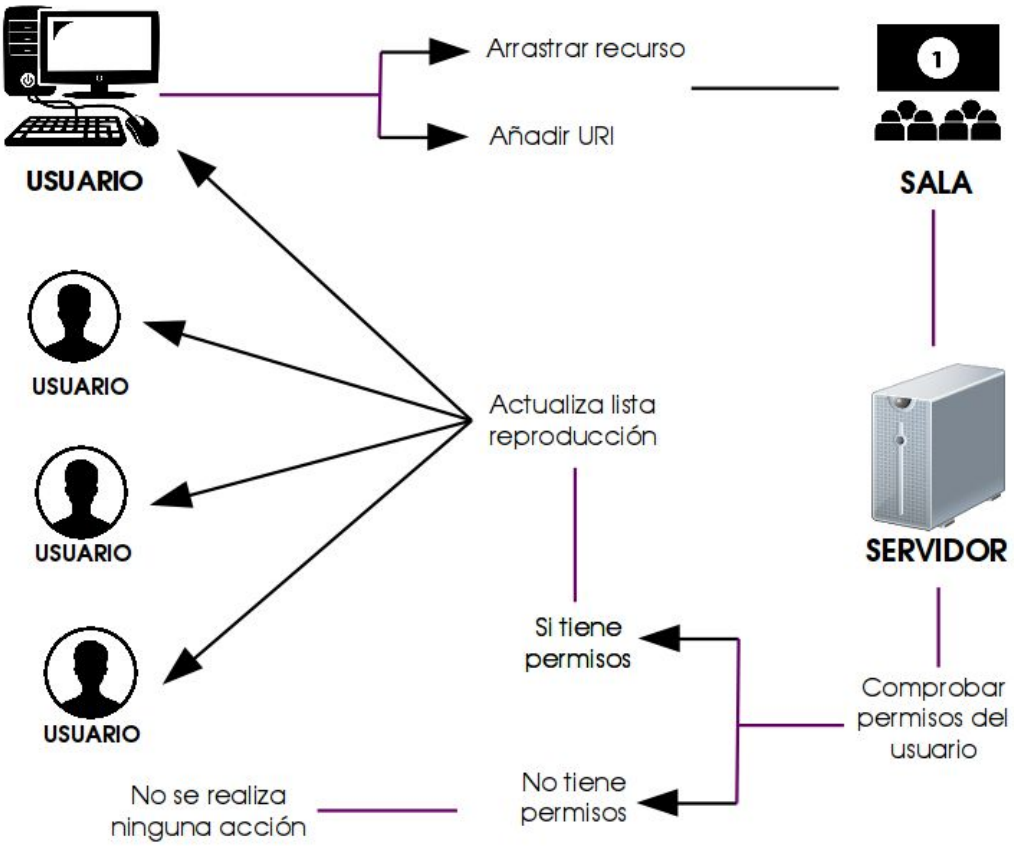
En este caso, se puede ver que hay dos usuarios que se intercambian un saludo y ambas frases son visualizadas en el cliente de cada usuario. Cuando un usuario nuevo se une a la sala y hay mensajes anteriores, a este usuario no se le muestran.



5.6.8 ACTUALIZAR LA LISTA DE REPRODUCCIÓN

Cuando se añade un fichero a la lista de reproducción, ya sea arrastrando desde nuestro directorio o agregando la URI, la lista de reproducción se ha de ir actualizando con los diferentes ficheros. Un usuario añade el recurso y en ese mismo momento, el servidor se encarga de ir actualizando la lista para todos los usuarios que están en la sala. De esta forma todos los usuarios pueden ver todos los ficheros que siguen compartidos.

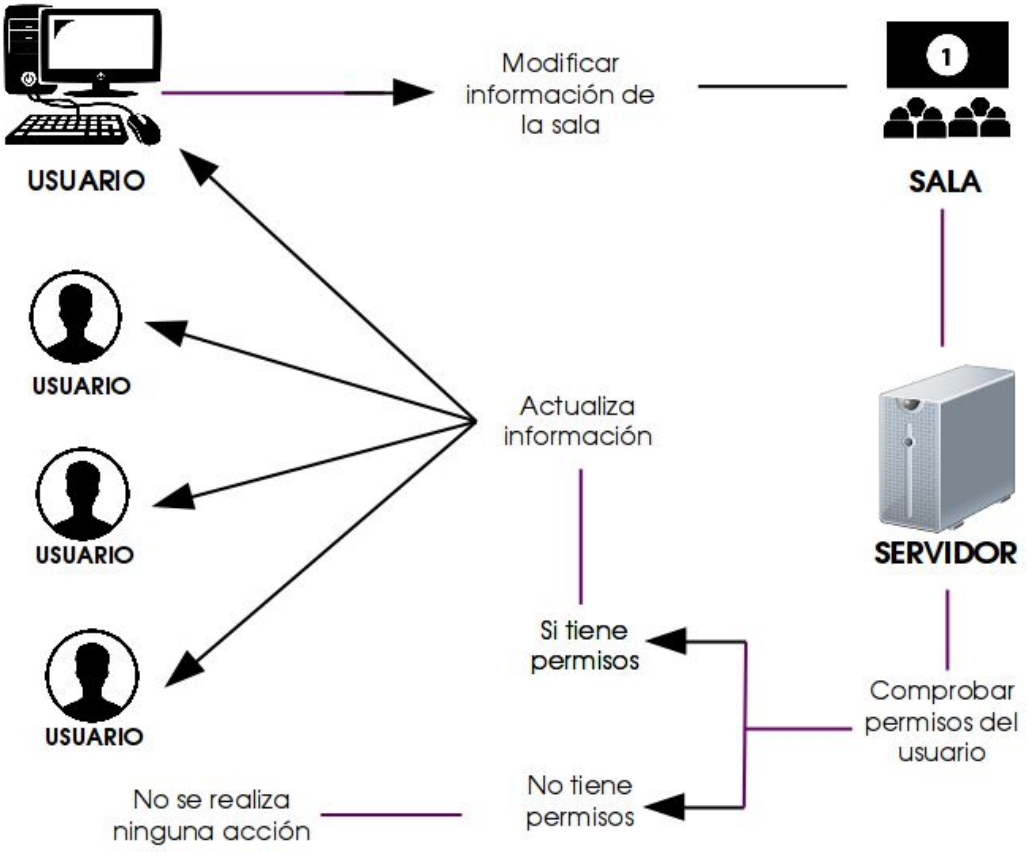




5.6.9 ACTUALIZAR INFORMACIÓN DE LA SALA

Cuando el usuario está dentro de una sala y se modifica información sobre la sala o permisos de esa sala concreta, se ha de enviar un evento al resto de usuarios para actualizar la información respecto a la sala o sobre los permisos que hay ahora

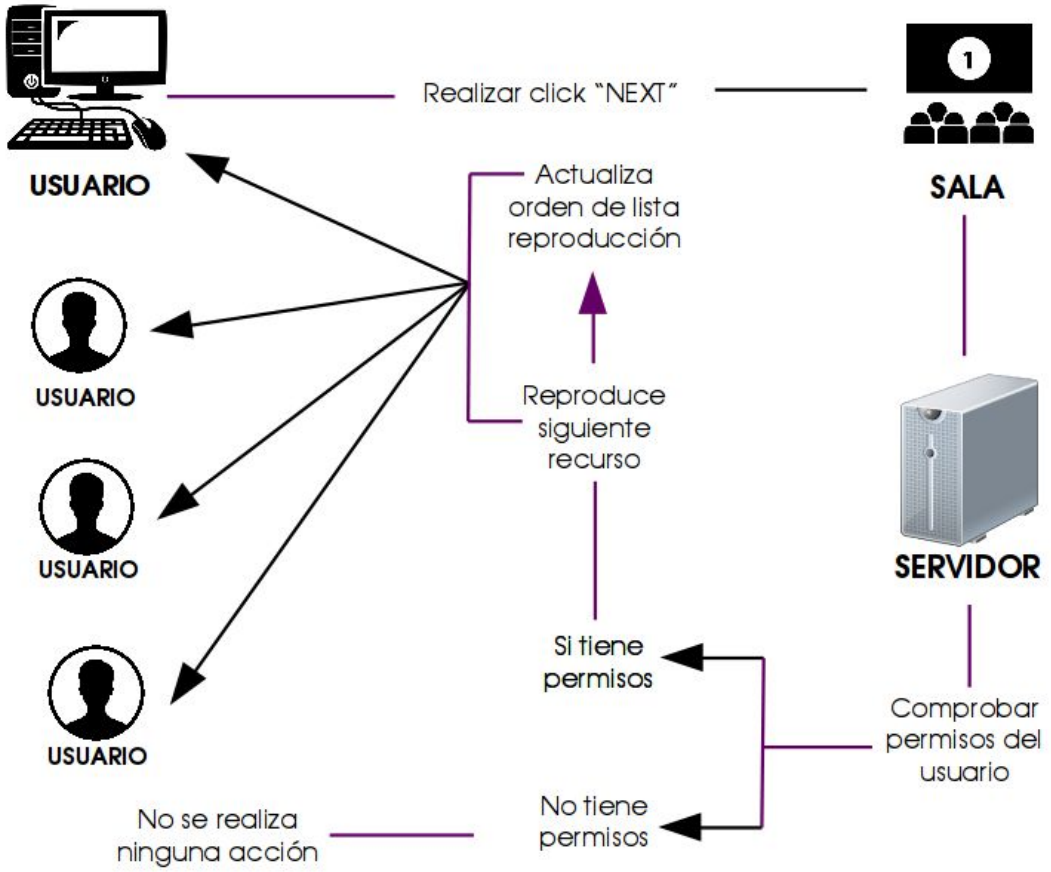
mismo por defecto. El cliente se encarga de modificar la información respecto a la sala y el servidor se encarga de emitir al resto de usuarios de la sala esa actualización.



5.6.10 ACTUALIZAR SIGUIENTE RECURSO A REPRODUCIR

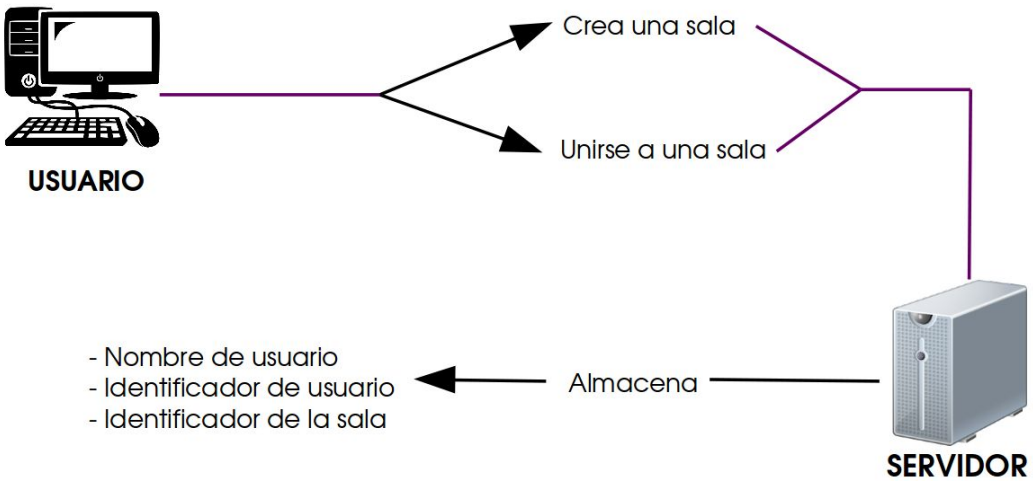
En el momento que el usuario añade el recurso a la lista de reproducción, al finalizar el actual, debe de proceder a comenzar a reproducir el siguiente. Básicamente, se está definiendo el significado de lista de reproducción en la que hay más de un recurso y se van reproduciendo uno detrás de otro. El cliente comprueba que el recurso actual haya finalizado, para reproducir el siguiente.

También se comprueba que forme parte de un fichero ".torrent" o sea de otro servicio y empezará a reproducir. El último paso que realiza el servidor es actualizar el orden de la lista de reproducción.



5.6.11 ALMACENAR DATOS DE LOS USUARIOS EN UNA SALA

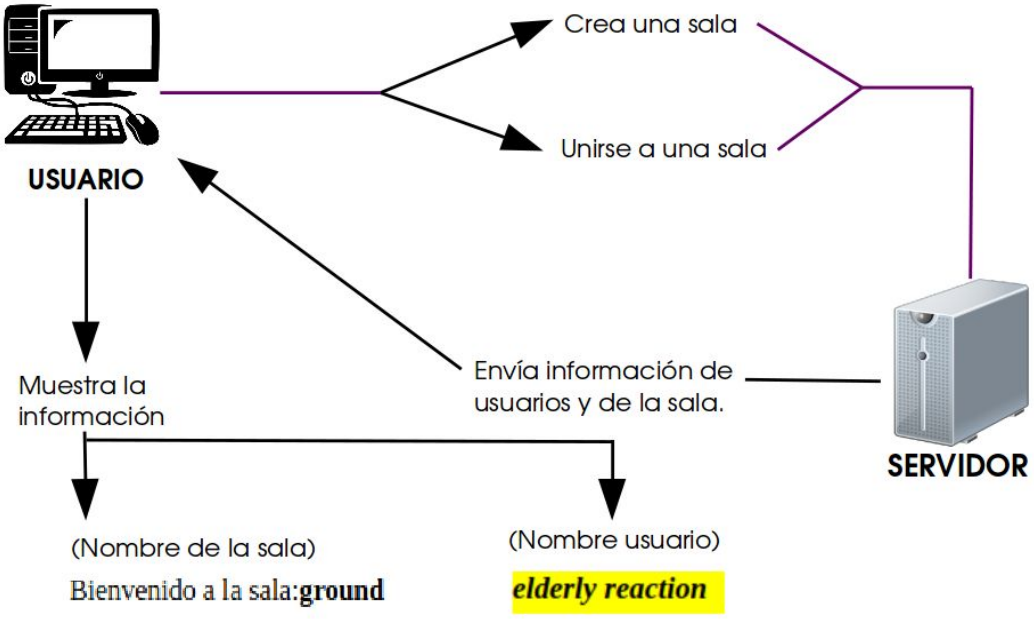
Al crear o unirse a una sala, se ha de almacenar la información del usuario juntamente con la de la sala en la que está. El cliente envía al servidor un mensaje de bienvenida donde se almacena el nombre del usuario, el identificador del usuario y la sala en la que está. De esta manera el servidor podrá devolver esa información al usuario al estar en la sala para mostrar esa información por pantalla y tener constancia de los usuarios que hay en ella y de la sala en la que están.



5.6.12 MOSTRAR INFORMACIÓN DE LA SALA

En el instante que un usuario crea una sala o un usuario se une a ésta, el usuario ha recuperar los datos sobre toda la información de la sala. El servidor envía los datos correspondientes a los clientes. Al crear una sala, el usuario entra en ella y el servidor envía la información para mostrar en el navegador.

En el caso de unirse es el mismo caso que el anterior. De esta manera al unirse una persona se le ha de actualizar los datos tanto de la sala, de la configuración de la sala, configuración de los usuarios, la lista de reproducción y el recurso que está siendo reproducido en ese momento.



Todos estos procesos es a lo que respecta entre la comunicación que hay entre los clientes que se encuentran en una misma sala y el servidor de la aplicación.

5.7 PERMISOS DE LA SALA

En esta aplicación existen diferentes permisos que se pueden configurar o personalizar con respecto a la sala que se encuentra el usuario.

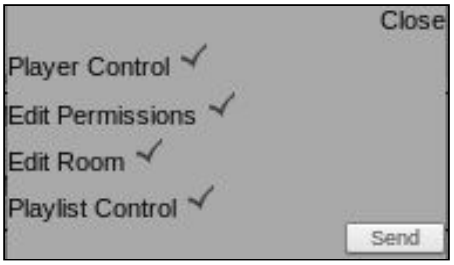
Es posible modificar el nombre de la sala, si al usuario no le agrada el nombre actual. También se puede definir si por defecto los usuarios que se encuentren en ella tendrán ciertos permisos o no, es decir, si tendrán control del reproductor, de la lista de reproducción o si pueden editar el nombre de la sala. Esto ayuda a no ir usuario por usuario definiendo los permisos, si van a tener los mismos.

En la imagen se visualiza las opciones de personalización que hay sobre una sala.

5.8 PERMISOS DE LOS USUARIOS

Los usuarios que se encuentran dentro de una sala, tienen definidos unos permisos sobre ciertas acciones que pueden realizar o no. Las acciones son las siguientes:

- Player Control: este permiso es el que gestiona si un usuario puede o no, repercutir en las acciones del reproductor, es decir, si tiene acceso a hacer click en los diferentes botones como “Play”, “Pause”, “Next”....
- Edit Permissions: este permiso es el que permite que un usuario tenga el poder de modificar los permisos asignados a otros usuarios de la sala.
- Edit Room: este permiso es el que decide si un usuario puede o no modificar la configuración de la sala en la que se encuentra.
- Playlist control: este permiso es el que permite poder modificar o no la lista de reproducción con todos los recursos que se han ido compartiendo entre los usuarios de dicha sala.



Los permisos que se muestran en la imagen sobre los usuarios son los mismos que podemos gestionar en menú de configuración de la sala, con la diferencia que en ese caso se dirige a todos los usuarios que se encuentren en ella.

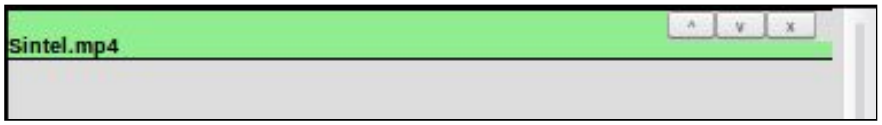
5.9 PERSONALIZACIÓN DEL USUARIO

El usuario tiene opciones de modificar su nombre en la sala y el color que usará en el chat como identificativo. Estas son opciones que ofrece la aplicación para una personalización sencilla, cómoda y básica.

5.10 FUNCIONAMIENTO DE LA LISTA DE REPRODUCCIÓN

La lista de reproducción es la parte más importante, ya que es dónde se almacenan todos los recursos que los usuarios comparten en una sala. En el momento de añadir un recurso vemos las acciones que puede ejecutarse:

- Priorizar Recurso: permite adelantar la reproducción del recurso, respecto a la posición en la que se encontraba.
- Posponer Recurso: permite que la reproducción del recurso se atrase a una posición inferior, dejando a otros recursos por encima en nivel de prioridad.
- Eliminar Recurso: permite borrar el recurso de la lista de reproducción.



5.11 LIMITACIONES DE LA APLICACIÓN

La aplicación cuenta con ciertas limitaciones a la hora de compartir ficheros, que no se han podido controlar debido a que las tecnologías que se han acabado usando no han implementado correctamente el permitir compartir ficheros un tamaño superior a 900MB por el momento. Esta tecnología, “Webtorrent”, trabaja de la manera que el recurso que se está reproduciendo se almacenan en la RAM de cada uno de los usuarios. Esto quiere decir, que dependiendo de la memória RAM que tenga la máquina que se esté usando, puede ser un problema grave.

Existe otro problema el cual, al intentar compartir ficheros a partir de 900MB, el recurso se rompe al hacer el proceso de hasheo del torrent. Se trata de un problema de la tecnología, que no ha tenido en cuenta este problema.

En la siguiente captura se muestra el “Monitor del Sistema” de la máquina cuando se desea compartir un vídeo de 1.5GB en una sala:

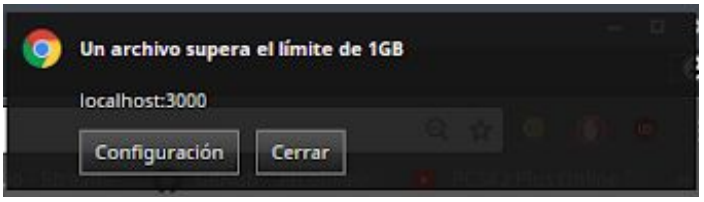
En la imagen se visualiza la máquina mientras realiza el hasheo del fichero compartido y cómo almacena en RAM poco a poco el archivo. Finalmente, este fichero no llega a cargarlo en la lista de reproducción de la sala.

Este es el único problema de uso con el que se encuentra la aplicación, después de investigar durante una gran cantidad de tiempo. Se ha mitigado indicando al usuario, haciendo uso de notificaciones del navegador, si el fichero que comparte es correcto o no.

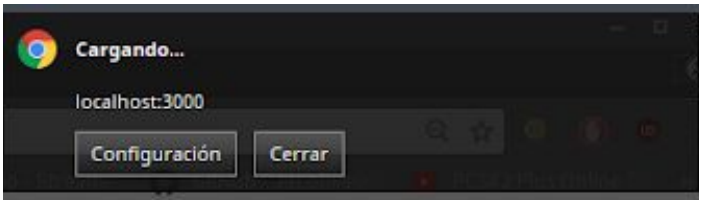
5.12 USO DE NOTIFICACIONES

La aplicación cuenta con notificaciones para poder informar al usuario en todo momento del estado de la sala. El navegador es el encargado de notificar al usuario y cuenta con la siguientes alertas:

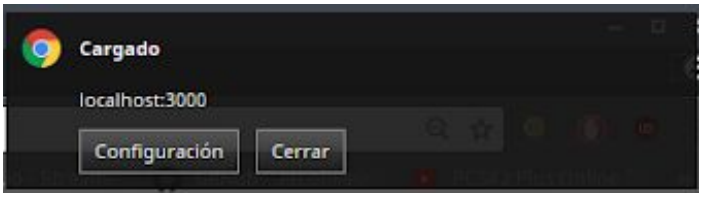
- Notificación de tamaño: se avisará al usuario en el momento de compartir un fichero que supere 1GB en una sala. Esto se notifica debido a la limitación existente que existe en la tecnología “Webtorrent” en este momento.



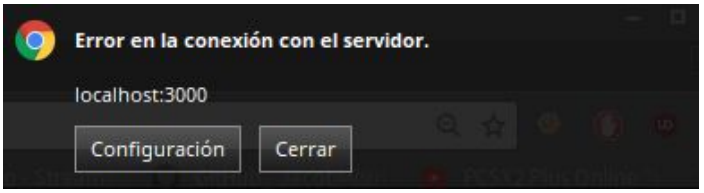
- Notificación del recurso cargándose: se notificará al usuario cuando el recurso añadido no haya sido cargado del todo en la sala, es decir, mientras el fichero está siendo hasheado.



- Notificación del recurso cargado: se avisará al usuario de que el recurso que ha añadido a la sala ha sido cargado con éxito y está listo para que el resto de usuarios puedan empezar a descargarlo y reproducirlo de forma sincronizada.



- Notificación de desconexión con el servidor: se avisará al usuario si existe algún tipo de error con el servidor.



6. CONCLUSIÓN

La conclusión del proyecto ha sido positiva ya que cumple con los requisitos más importantes y básicos del propósito que se tenía al inicio: compartir ficheros sin necesidad de realizar el paso de “subida” a un servidor y poder visualizarlos de manera sincronizada entre diferentes usuarios en una misma sala. Toda las comunicaciones entre los diferentes clientes con el servidor ha sido de los más complejo de realizar, pero al mismo tiempo lo más satisfactorio. Ha sido frustrante no poder arreglar el problema de la limitación a la hora de compartir ficheros, lo positivo es que en el caso de que en un futuro los desarrolladores de la tecnología “Webtorrent” arreglen ese problema, quedará solucionado en la aplicación.

7. FUTURAS MEJORAS

En este apartado se proponen mejoras de cara a un futuro, que no han sido posible llevar a cabo por falta de tiempo:

- Desarrollar una Aplicación Android.
- Añadir Chat de Voz y Vídeo.
- Integrar más servicios como SoundCloud, Openload, Vimeo….
- Sincronización entre ficheros de tipo PDFs.
- Permiso de Baneos de usuarios en una sala.
- Implementación de salas públicas organizadas por categorías.
- Integrar editor de documentos de texto sincronizado entre usuarios.
- Limitar conexiones por IPs.

8. GLOSARIO

8.1 QUÉ ES SOCKET.IO?

Socket.IO es un framework que se usa para poder realizar la comunicación entre los clientes de la aplicación (Navegador) y el servidor (Node.js), de manera bidireccional y en tiempo real. Permite una conexión de larga duración y con posibilidad de reconexión automática. Por otra parte es soportado por los navegadores más importantes como Firefox, Google Chrome o Safari.

La característica más importante es que permite crear el sistema de salas, donde permite unir a diferentes usuarios y comunicarse entre ellos, es decir, con un grupo de usuarios.

8.2 QUÉ ES NODEJS?

NodeJS es una librería o entorno de JavaScript que se usa en el lado del servidor de una aplicación web para poder realizar uso de eventos, es decir, una programación asíncrona. Ésta fue creado por Google para poder desarrollar aplicaciones web sencillamente, con gran estabilidad y velocidad. Destaca en las aplicaciones que manejan un gran conjunto de conexiones simultáneas en tiempo real y que requiere un gran rendimiento.

8.3 QUÉ ES WEBTORRENT?

Webtorrent es un cliente P2P que funciona a través de la web. Nos permitirá en nuestras aplicaciones web poder compartir contenido con otros usuarios de manera inmediata. Este cliente está escrito en JavaScript y usa el protocolo WebRTC para garantizar el transporte de los datos P2P (Peer To Peer). Una de sus características principales es que no hará falta descargar el contenido completo del torrent para poder visualizarlo, sino que a la vez que vayamos descargando de los pares, podremos ir mostrándolo en nuestro navegador. Los trackers han de ser compatibles con los protocolos Webtorrent o WebRTC.

8.4 QUÉ ES PLAYEMJS?

PlayemJS es una librería o componente en lenguaje Javascript con el cual permite reproducir pistas de audio o vídeo de diferentes plataformas importantes como:

- Youtube
- Soundcloud
- Deezer
- Bandcamp
- Vimeo
- Dailymotion
- Jamendo

Cada una de las plataformas, al tener reproductores distintos, se usarán de diferente manera a la hora de usarlos en la aplicación. Para la mayoría de las plataformas es necesario hacer uso de APIs para que se pueden integrar en la aplicación.

8.5 QUÉ ES MONGODB?

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Ha sido escogido porque es el tipo de base de datos que mejor encajaba con la función que le toca realizar, por otra parte se quería innovar usando una base de datos que no fuera relacional.

Durante el proyecto se hdeace uso este tipo de base de datos para poder almacenar ciertos datos sobre las salas y los usuarios. Estos datos sirven para tener un pequeño sistema de monitorización de las distintas salas existentes en el servidor.

9. BIBLIOGRAFÍA

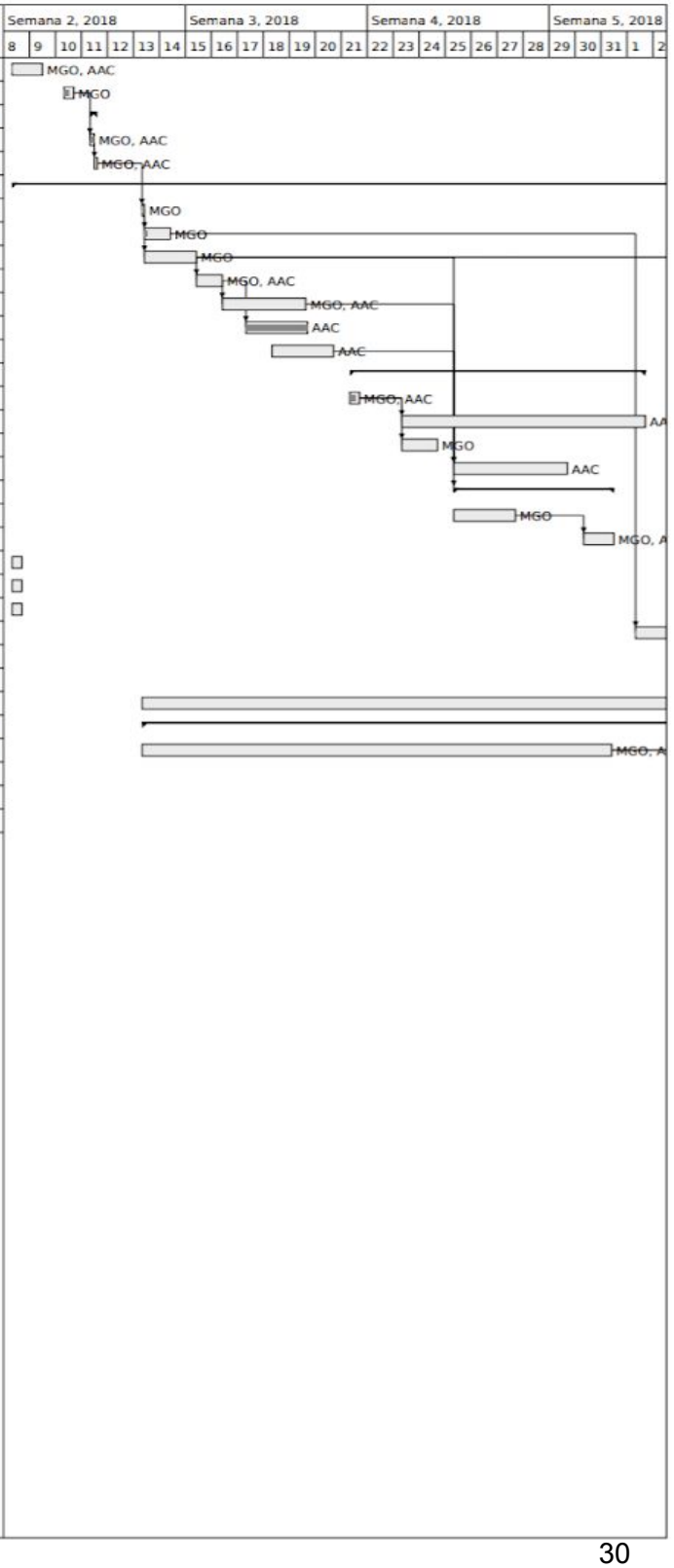
En este apartado se encuentran todos los enlaces que hemos ido visitando para adquirir conocimientos y documentación:

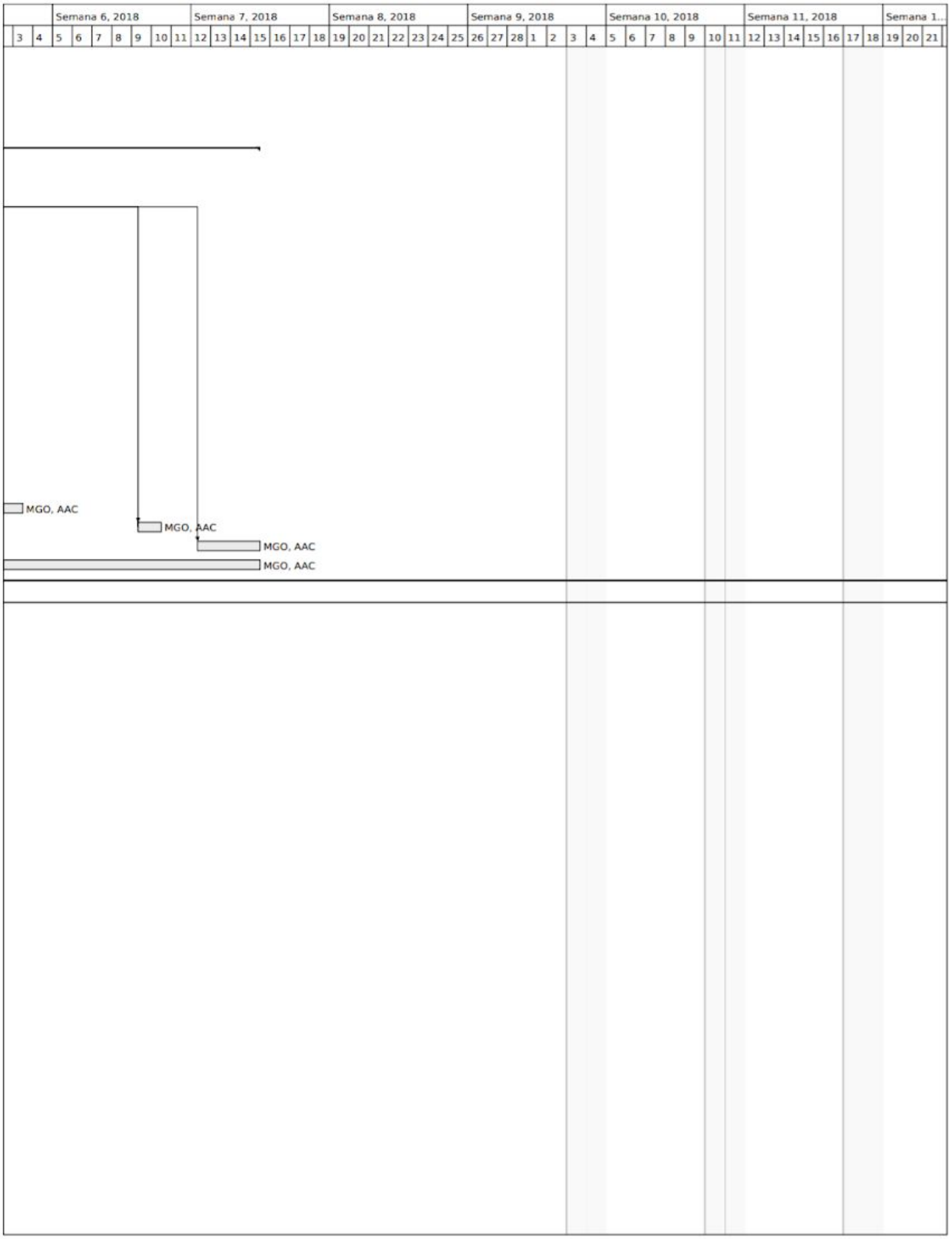
-https://socket.io/
-https://www.npmjs.com/
-https://github.com/
-https://webtorrent.io/
-https://stackoverflow.com
-https://instant.io/
-https://nodejs.org/en/
-https://cdn.rawgit.com/adrienjoly/playemjs/master/test/sample.html

10. ANEXOS

10.1 DIAGRAMA DE GANTT

Nombre	Trab...
Investigar tecnologías a usar	3d
Realizar pruebas antes de comenzar	1d
Diseñar aplicación	**1d 3h**
Diseñar aplicación web	1d
Diseñar servidor	3h 30...
Implementar aplicación	**61d 2h**
Crear un servidor web	2h 15...
Implementar salas	1d
Implementar comunicación entre cliente y servidor	2d
Implementar streaming P2P	2d
Implementar sincronización entre clientes del contenido multimedia	7d
Implementar Drag and Drop	3d
Generar torrents a partir de ficheros (Magnet Links)	3d
Definir diseño de la aplicación	**12d**
Estructurar Diseño	2d
Diseñar Aplicación	10d
Implementar chat de texto dentro de las salas	2d
Implementar Playlists	5d
Implementar usuarios	**6d**
Guardar estado del usuario en la sala	3d
Añadir permisos	3d
Añadir soporte para más tipos de ficheros (.pdf, .txt, .png, .jpg....)	1d
Implementar base de datos MongoDB para monitorizar información	1d
Implementar uso de enlaces de otros servicios (Youtube)	1d
Definir salas públicas y privadas	5d
Gestionar tracker	3d
Añadir seguridad a la aplicación	7d
Comprobar aplicación	67d
Memoria del Proyecto	**41d 4h**
Documentar la memoria	36d 4h
Revisar la versión final de la memoria	3d
Entregar la memoria	2d
Preparación de la presentación	5d




Semana 6, 2018
Semana 7, 2018
Semana 8, 2018
Semana 9, 2018
Semana 10, 2018
Semana 11, 2018
Semana 1...
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
MGO, AAC
MGO, AAC
MGO, AAC
MGO, AAC

10.2 TECNOLOGÍAS DESECHADAS

Aquí tenemos las tecnologías o librerías que se han dejado de usar por problemas de optimización de código o por otros motivos:

- Websockets nativos: La razón es porque encontramos la librería de sockets.io. Ésta librería nos permite manejar de manera más eficiente el concepto de las salas ya que permite un multicast sencillo.
- Eventos de DOM (Document Object Model) y JQuery: El motivo es que no se podía determinar si un usuario realiza un acción o si la acción se realiza automáticamente.
- Uso de semáforos: La razón es debido a que causaban desincronización y problemas entre los diferentes clientes de la aplicación.
- Crear aplicación de Escritorio con Electron: El problema es que seguí almacenando en RAM todos los archivos que se compartían en una sala, en vez de guardarlo en el disco local.

10.3 MANUAL DE INSTALACIÓN

Para instalar la aplicación correctamente en cualquier plataforma, se deberá tener instalado “MongoDB” y “NodeJS” (Versión 7.5 o superior). Una vez funcionan se ha instalado las aplicaciones y se tiene el proyecto en la máquina dónde hará el rol de servidor, se abrirá una shell dentro de el directorio del proyecto y se ejecutará el siguiente comando:

-“npm install” → Descargar todas las librerías del proyecto.

Esto descargará todas las librerías que se hacen uso en el proyecto, de esta manera, se permite que el proyecto ocupe un tamaño inferior.

Finalmente, para ejecutar el proyecto:

-”node app.js”. → Ejecutar la aplicación haciendo uso de Node.js

Al ejecutar la aplicación aparecerá lo siguiente:

```
/usr/bin/node /home/aalhama/WebstormProjects/peerSync/app.js
BitTorrent Tracker running on 8000
Server listening at port 3000
Conectado correctamente a la base de datos
```

10.4 MANUAL DE USUARIO

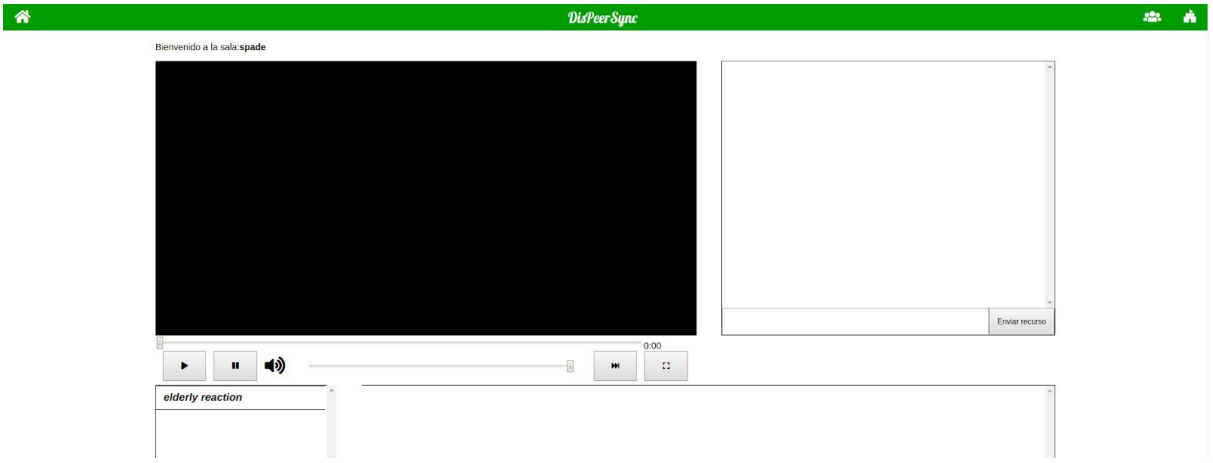
Inicialmente se ha de comprobar que el servidor está iniciado y funcionando correctamente para poder acceder a la aplicación mediante el navegador sin ningún tipo de problema. Para hacer el uso adecuado se deberá abrir el puerto “3000” del router, para que los usuarios que no se encuentran en la misma red que el servidor puedan usar la aplicación.

10.4.1 CREAR UNA SALA

En este paso se procederá a crear una sala para que otras personas se puedan unir en un futuro.

Primero se ha de acceder a la aplicación haciendo uso del siguiente enlace:
-https://IPdelServidor:3000

Una vez el usuario ha accedido a la interfaz inicial, se hará click al botón de “Crear Sala”. De esta manera automáticamente el usuario entrará en una sala nueva, después de crearla.

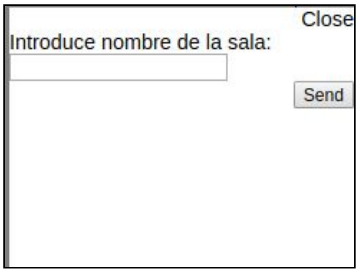


10.4.2 UNIRSE UNA SALA

Este paso permitirá a un usuario unirse a una sala de la aplicación, usando el nombre de la sala a la que desea conectarse.

Primero se ha acceder a la aplicación usando el siguiente enlace:
-https://IPdelServidor:3000

Una vez el usuario se encuentra en el menú principal de la aplicación, es decir, lo que sería la “HOME”, se hará click sobre el botón de “Unirse a una Sala”:

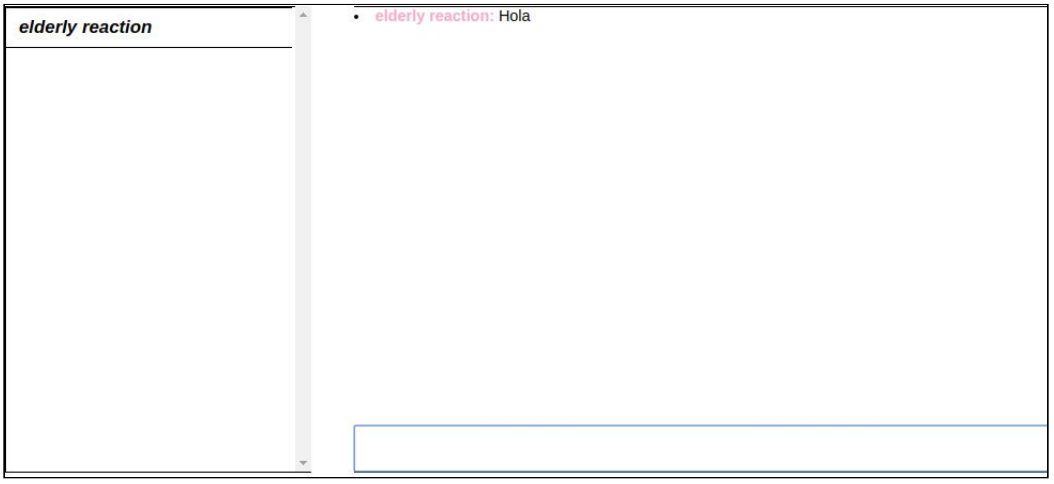


El usuario tendrá que introducir en el recuadro el nombre de la sala y hacer click en el botón de “Unirse”.

Después de esto, el usuario será redirigido a la sala con el resto de usuarios que la forman.

10.4.3 USO DEL CHAT

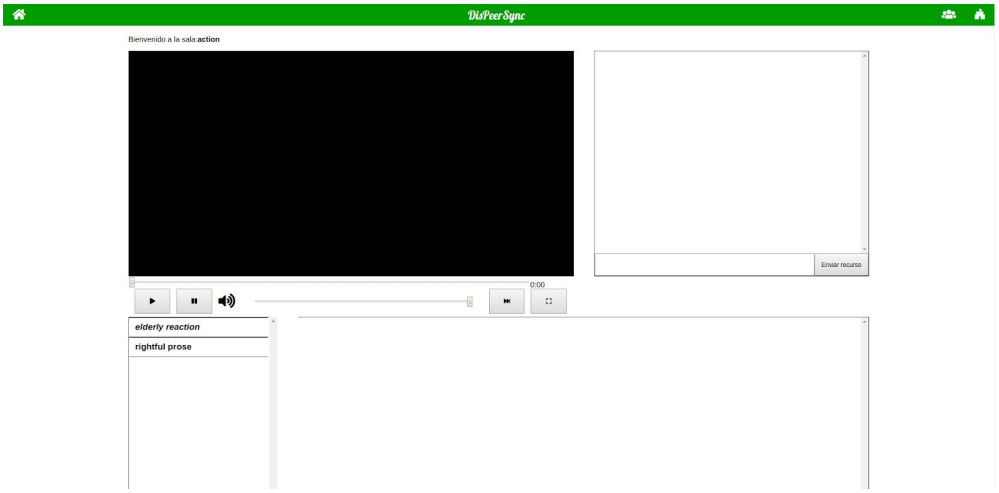
En este paso se mostrará la opción adicional que tienen los usuarios dentro d una sala de poder conversar mediante el uso de un chat con el resto de usuarios.



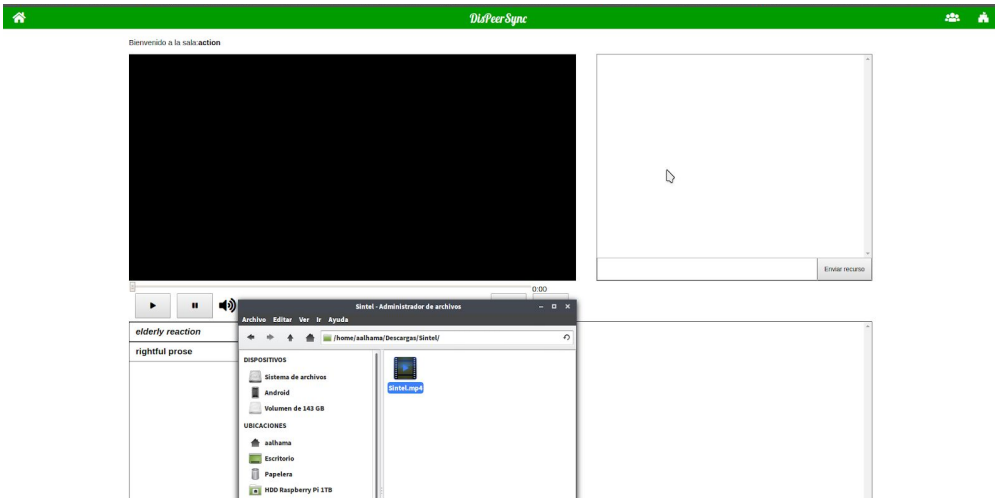
10.4.4 COMPARTIR FICHERO TORRENT (LOCAL)

Este paso define el funcionamiento de la aplicación. Cuando el usuario se encuentra dentro de una sala y tiene el deseo de compartir un fichero de su máquina con otros usuarios o desea directamente compartir un enlace con el resto, como puede ser un “Magnet Link”. En este caso, se usará mediante un fichero local de la máquina del usuario.

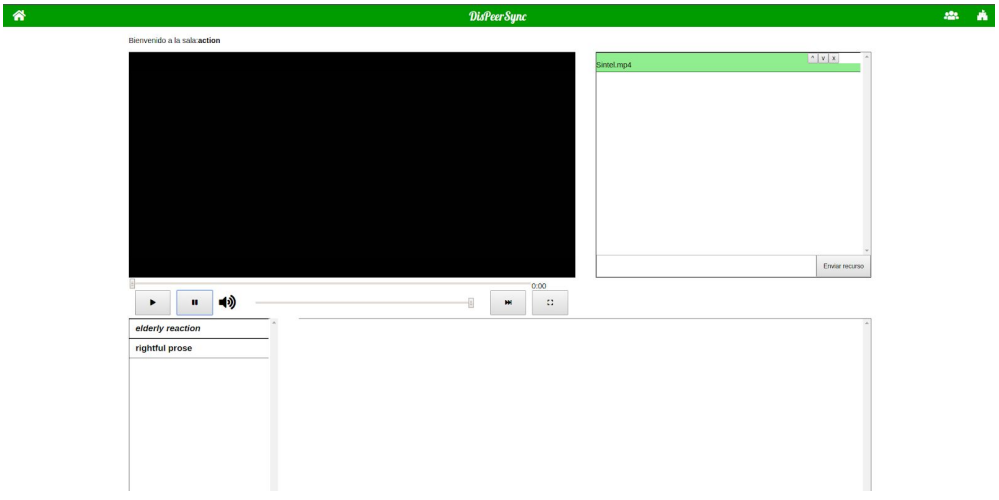
Primero el usuario tiene que estar en una sala y tener preparado el fichero que desea compartir con el conjunto de usuarios.



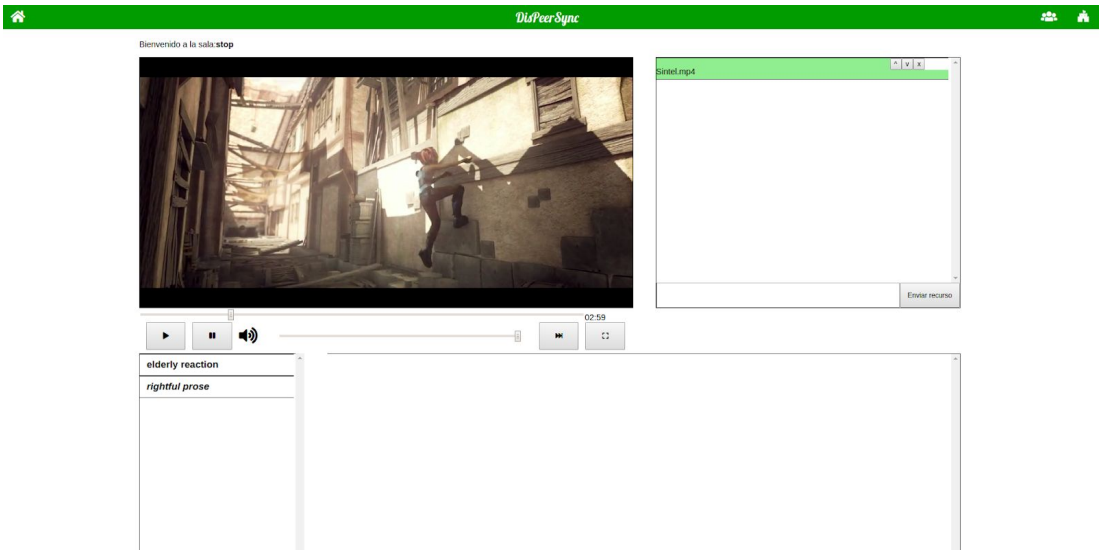
El siguiente paso es arrastrar el archivo preparado al navegador, para que se añada a la lista de reproducción de esa misma sala:



Cuando el recurso ha sido cargado en la lista de reproducción, el resto de usuarios empezarán a descargarse el elemento:

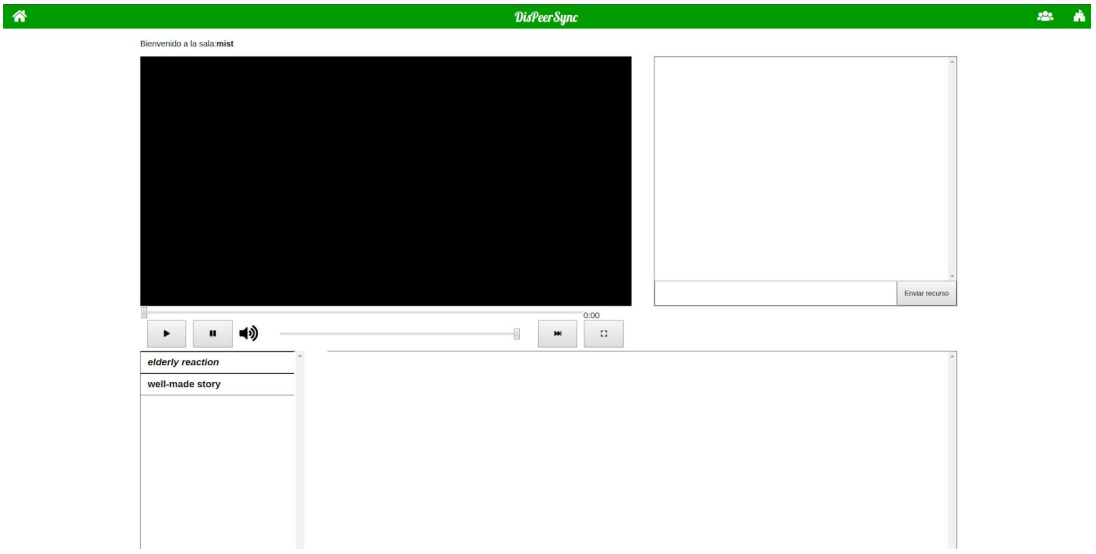


Para terminar una vez todos los usuarios de la sala tengan el recurso de la lista de reproducción descargado mínimamente, el fichero comenzará a reproducirse de manera sincronizada:



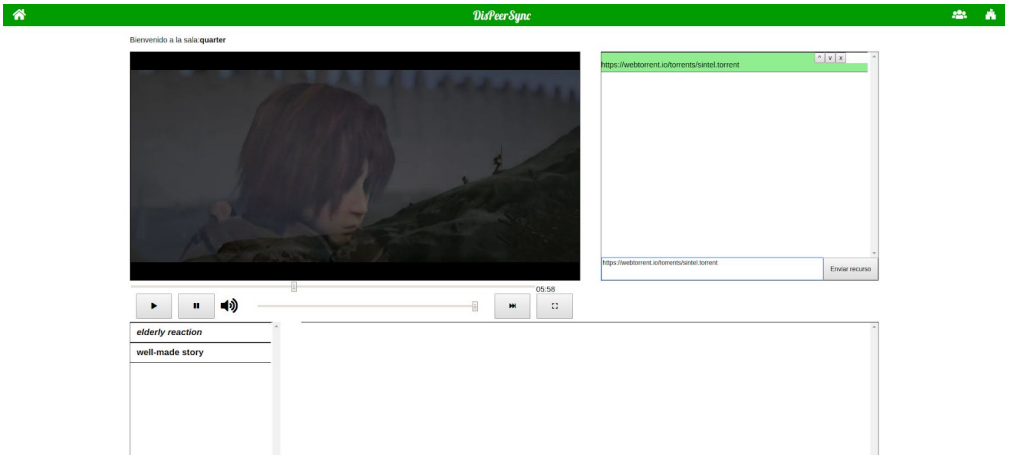
10.4.5 COMPARTIR FICHERO TORRENT (URI)

También el usuario puede compartir un fichero ".torrent" que no se encuentre en su ordenador, es decir, desde un enlace como se comentó en el paso anterior. El usuario debe situarse en la sala donde desea compartir el recurso:



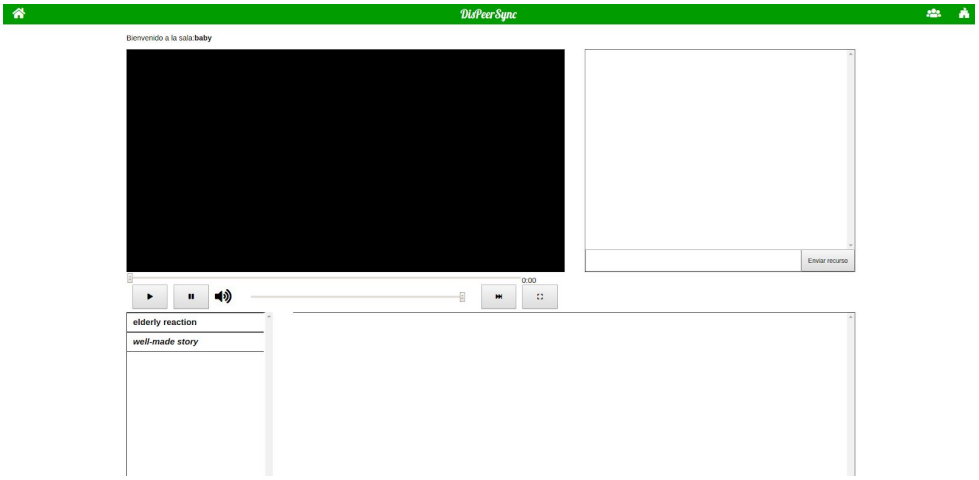
El siguiente paso es tener preparado el enlace que se quiere compartir y introducir la URI del mismo. Luego se hace click en "Enviar Recurso":

Ahora el recurso se encontrará en la lista de reproducción de la sala, esperando a que los usuarios comiencen a descargarlo. Después el recurso comenzará a reproducirse de manera sincronizada:

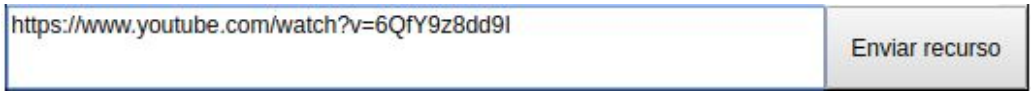


10.4.6 COMPARTIR VÍDEO DE YOUTUBE

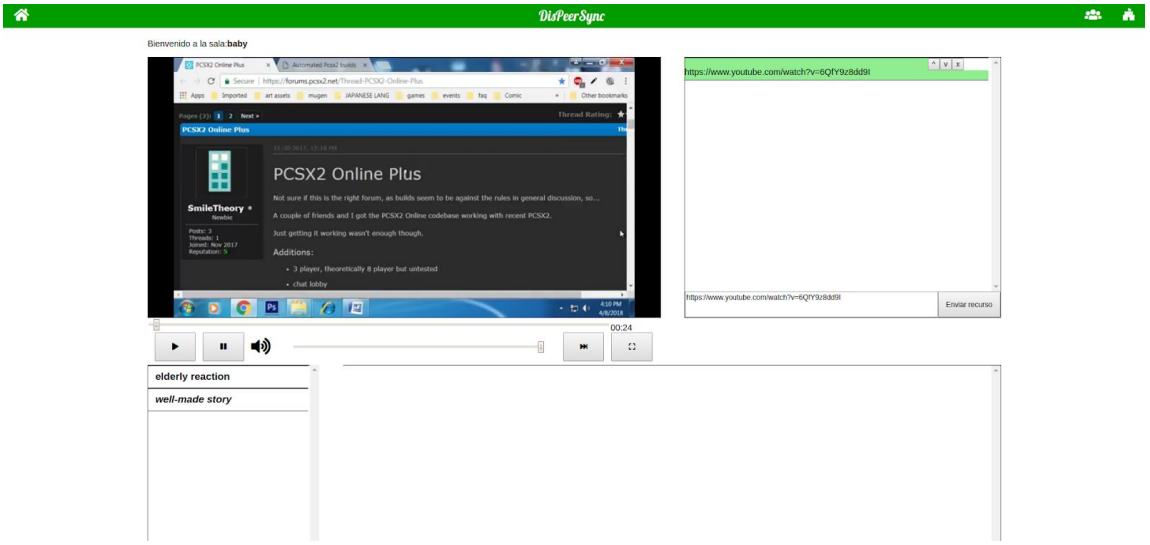
Como detalle adicional la aplicación permite compartir enlaces del servicio “Youtube” en una sala. En este paso se mostrará cómo realizarlo. El primer paso, esencial, es situarse en la sala:



En este instante el usuario deberá seleccionar el enlace del vídeo de Youtube que desee distribuir con el resto de usuarios. Una vez está seleccionado, deberá usarse la misma técnica que con los enlaces “.torrent”. Se introduce el enlace y se hace click en “Enviar recurso”:

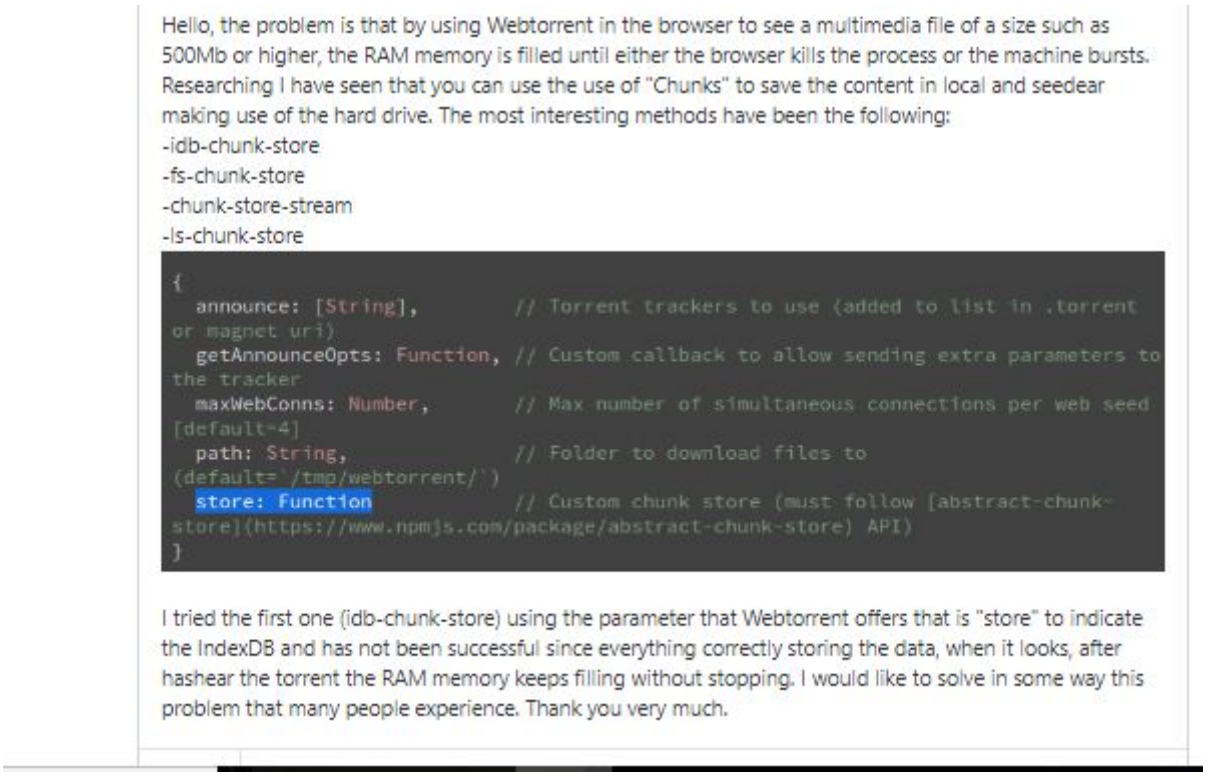


Ahora el vídeo de Youtube estará añadido en la lista de reproducción en la sala. Para terminar, los usuarios cargarán el vídeo compartido y en ese momento comenzará a reproducirse el vídeo de manera sincronizada:



10.5 CONVERSACIÓN CON DESARROLLADORES DE TECNOLOGÍA “WEBTORRENT”

Cuando surgió el problema de poder compartir y visualizar ficheros mayores a 1GB usando el protocolo de Webtorrent, se intentó comunicar con los desarrolladores por si había algún tipo de solución respecto al problema. En el github, en el apartado de “Issues” se puede escribir y comunicar a los desarrolladores o otros usuarios ciertas dudas, mejoras o problemas que surgen. El post creado fue el siguiente:



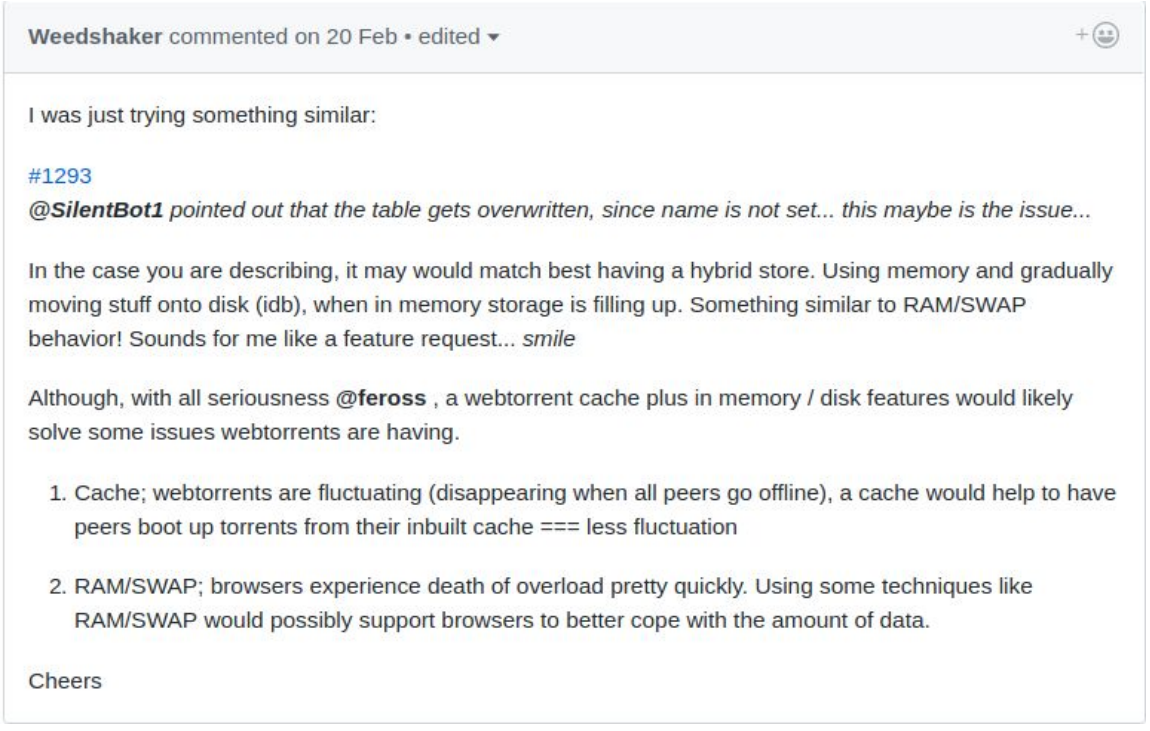
Hello, the problem is that by using Webtorrent in the browser to see a multimedia file of a size such as 500Mb or higher, the RAM memory is filled until either the browser kills the process or the machine bursts. Researching I have seen that you can use the use of "Chunks" to save the content in local and seedear making use of the hard drive. The most interesting methods have been the following:
-idb-chunk-store
-fs-chunk-store
-chunk-store-stream
-ls-chunk-store

```
{
  announce: [String],       // Torrent trackers to use (added to list in .torrent
or magnet uri)
  getAnnounceOpts: Function, // Custom callback to allow sending extra parameters to
the tracker
  maxWebConns: Number,      // Max number of simultaneous connections per web seed
[default=4]
  path: String,             // Folder to download files to
(default=`/tmp/webtorrent/`)
  store: Function           // Custom chunk store (must follow [abstract-chunk-
store](https://www.npmjs.com/package/abstract-chunk-store) API)
}
```

I tried the first one (idb-chunk-store) using the parameter that Webtorrent offers that is "store" to indicate the IndexDB and has not been successful since everything correctly storing the data, when it looks, after hashear the torrent the RAM memory keeps filling without stopping. I would like to solve in some way this problem that many people experience. Thank you very much.

Las respuestas fueron negativas, ya que se tendría que implementar en la tecnología usada “Webtorrent” que en vez de guardar el fichero en la RAM, se almacene en un fichero temporal para así no genera ningún tipo de destrucción a la hora de compartir el fichero. La respuesta fue del desarrollador de “Webtorrent”.

RESPUESTAS



Weedshaker commented on 20 Feb • edited

I was just trying something similar:

#1293
***@SilentBot1** pointed out that the table gets overwritten, since name is not set... this maybe is the issue...*

In the case you are describing, it may would match best having a hybrid store. Using memory and gradually moving stuff onto disk (idb), when in memory storage is filling up. Something similar to RAM/SWAP behavior! Sounds for me like a feature request... *smile*

Although, with all seriousness **@feross** , a webtorrent cache plus in memory / disk features would likely solve some issues webtorrents are having.

1. Cache; webtorrents are fluctuating (disappearing when all peers go offline), a cache would help to have peers boot up torrents from their inbuilt cache === less fluctuation
2. RAM/SWAP; browsers experience death of overload pretty quickly. Using some techniques like RAM/SWAP would possibly support browsers to better cope with the amount of data.

Cheers

SilentBot1 commented on 25 Feb

Hey **@aalhama**,

After looking through a memory snapshot, the cause of the memory filling up is due to the usage of immediate-chunk-store in torrent.js@492, which stores the chunk in memory until it is written to the chunk store set in opts.store. In the current implementation of immediate-chunk-store in the webtorrent library, this seems unavoidable if the chosen chunk store is too slow to store the chunks.

This seems to only be an issue when seeding files as read speeds from disk are greater than write speeds to the chunk store. Would reading only chunks of the file at a time be possible instead of processing the whole file at once?

Hopefully this has been useful, but I would love to hear what peoples thoughts on how to deal with this are.

SilentBot1 commented on 3 Mar • edited

@aalhama @feross,

I have confirmed the issue to be caused by the immediate-chunk-store `store.mem` array filling up with pending writes when a slow stores, such as an IndexedDB store, is used. As the read stream is faster than the store, the pending chunks waiting to be written to store are then stored in `store.mem` until the `store.put` has finished. This eventually builds up on large files (4GB in my case) and causes the browser to crash (tested in both Chrome 64.0.3282.186 and Firefox 58.0.2)

I was able to avoid `store.mem` filling up by using a stream throttle, such as node-throttle, in the stream `pump` here, though setting a universal throttle speed isn't a good solution as user environments may vary, implementing some form of backpressure such as a variable bit rate throttle based on the immediate-chunk-store pending writes would be my two cents but I would love to hear other ideas on how to deal with the situation.

All the best.