shareplay

Carlos Mateo Esteban y Juan Pujalte

Trabajo final de grado - DAM

Contenido

[1. Introducción. 2](#_Toc198920072)

[2. Análisis. 3](#_Toc198920073)

[Introducción. 3](#_Toc198920074)

[Trabajo conjunto. 3](#_Toc198920075)

[Requerimientos de la aplicación. 3](#_Toc198920076)

[Requerimientos de la interfaz. 3](#_Toc198920077)

[3. Back-end. 4](#_Toc198920078)

[Introducción. 4](#_Toc198920079)

[Servidor. 4](#_Toc198920080)

[Cliente. 4](#_Toc198920081)

[Control de reproductor MPV. 4](#_Toc198920082)

[Errores, problemas encontrados. 4](#_Toc198920083)

[4. Front-end. 5](#_Toc198920084)

[Introducción. 5](#_Toc198920085)

[Vistas. 5](#_Toc198920086)

[Controladores. 5](#_Toc198920087)

[Persistencia y control de JSON. 5](#_Toc198920088)

[Errores, problemas encontrados. 5](#_Toc198920089)

[5. Conclusión. 6](#_Toc198920090)

# Introducción.

Se ha llevado a cabo el desarrollo de una aplicación de escritorio que los usuarios se instalarán de forma local con el fin de crear o unirse en salas, los usuarios se unirán a las salas o las crearán con el fin de iniciar un reproductor que está integrado en la aplicación y ya desarrollado llamado MPV, de forma local cada uno de los usuarios iniciara el reproductor con el video a reproducir que se seleccionara de la ruta que hemos elegido previamente para seleccionar el video de reproducción y al momento del inicio de los reproductores estos se sincronizaran entre sí para reproducirse de manera simultánea a través de un servidor donde se comunicaran.

Los videos estarán sincronizados entre sí de manera que, si uno de los usuarios de la sala para la reproducción esta se parara en todos, si un usuario entra posteriormente cuando inicie la reproducción de su reproductor esta se iniciara por el minuto en el que los demás integrantes están en dicha reproducción para que se haga de manera simultánea, de esta manera se consigue que de manera local ver un video con otra persona en la distancia y que se vean juntos ambas reproducciones en el minuto exacto y con los parones sincronizados.

El fin de la aplicación mediante una interfaz sencilla donde rellenar unos campos mínimos conectarte a una sala donde estarán otros integrantes de la sala, el fin principal de la aplicación es la de sincronizar los reproductores simultáneamente cada uno desde su ubicación local y poder ver la misma reproducción de forma local, para ello se ha hecho un análisis de requerimientos y tecnologías necesarias para poder llevar a cabo el desarrollo de dicha aplicación.

Este análisis se ha llevado a cabo teniendo en cuenta los requisitos de dicha aplicación, dicha aplicación tiene un funcionamiento sencillo, dándole a un botón de inicio se inicia una reproducción de un video y se sincroniza con los reproductores de los demás integrantes de dicha sala, pero esto lleva unos requerimientos internos importantes, es necesario hacer la lógica interna de manera de que las solicitudes entren cada una por un hilo y un servidor gestione todas las solicitudes, después, este servidor debe gestionar dicha sala con todos los integrantes.

El trabajo de dicha aplicación se ha llevado a cabo entre dos personas, para ello se debió tener un canal de trabajo en común para poder llevar a cabo un control de versiones, para ello se ha consensuado que la mejor forma de llevarlo a cabo es mediante un repositorio en la nube donde con el reparto de tareas se vayan subiendo mediante pull request a dicho repositorio.

# Análisis.

## Introducción.

Para llevar a cabo el desarrollo de esta aplicación se ha llevado a cabo un análisis de los requisitos necesarios para ello, también ha sido necesario llevar a cabo un consenso entre las partes implicadas para llevar un registro del trabajo llevado a cabo por cada una de las partes implicadas y que haya sido de una forma ordenada.

Desde una visión generalizada es necesario llevar a cabo un control de las versiones durante el desarrollo de la aplicación, para ello se pensó en GitHub, con ellos se espera tener un control de las versiones y el desarrollo de la aplicación de una manera ordenada y consensuada.

Para los requerimientos de la aplicación se ha tenido en cuenta la necesidad de gestionar de manera simultaneas varias peticiones y procesos durante el desarrollo de cada cliente, para ello tenemos que tener en cuenta que un cliente representa un hilo de para el servidor al que debe gestionar y organizarlo mientras los demás clientes también son gestionados, para ello se necesita concurrencia de hilos, estos deben ser gestionados tanto a la llegada al servidor de manera concurrente, es decir, que pueda gestionar varias peticiones al mismo tiempo así como una vez se gestione la entrada al servidor se gestione su petición hacia la sala solicitada, en dicha sala los clientes o hilos tienen que ser gestionados en conjunto de manera concurrente, al mismo tiempo que sigue gestionando las otras salas con los demás clientes en cada una de ellas, en dichas salas el servidor estará escuchando a los clientes en todo momento y enviando respuestas en consecuencia de lo recibido de manera broadcast.

De la misma manera que el servidor gestiona varios hilos, es necesario que el cliente gestione varios procesos al mismo tiempo, el cliente tiene como principal trabajo conectarse al servidor para que empiece a gestionarlo, enviarle eventos, gestionar las respuestas recibidas del servidor y controlar el reproductor de video MPV de manera que el cliente este escuchando los cambios en dicho reproductor como la pausa, la reanudación y el cambio o saltos de minuto de reproducción, el cliente tiene que ser capaz de gestionar todos estos procesos de manera concurrente y enviar al servidor los datos recabados para que informen a los demás clientes de la sala.

Para la interfaz se necesita una interfaz mínima y sencilla que gestione los datos necesarios que el cliente tiene que enviar hacia el servidor, para ello se ha visto necesario una ventana que no se puede agrandar con los campos necesarios para escribirlos y enviarlos a la conexión.

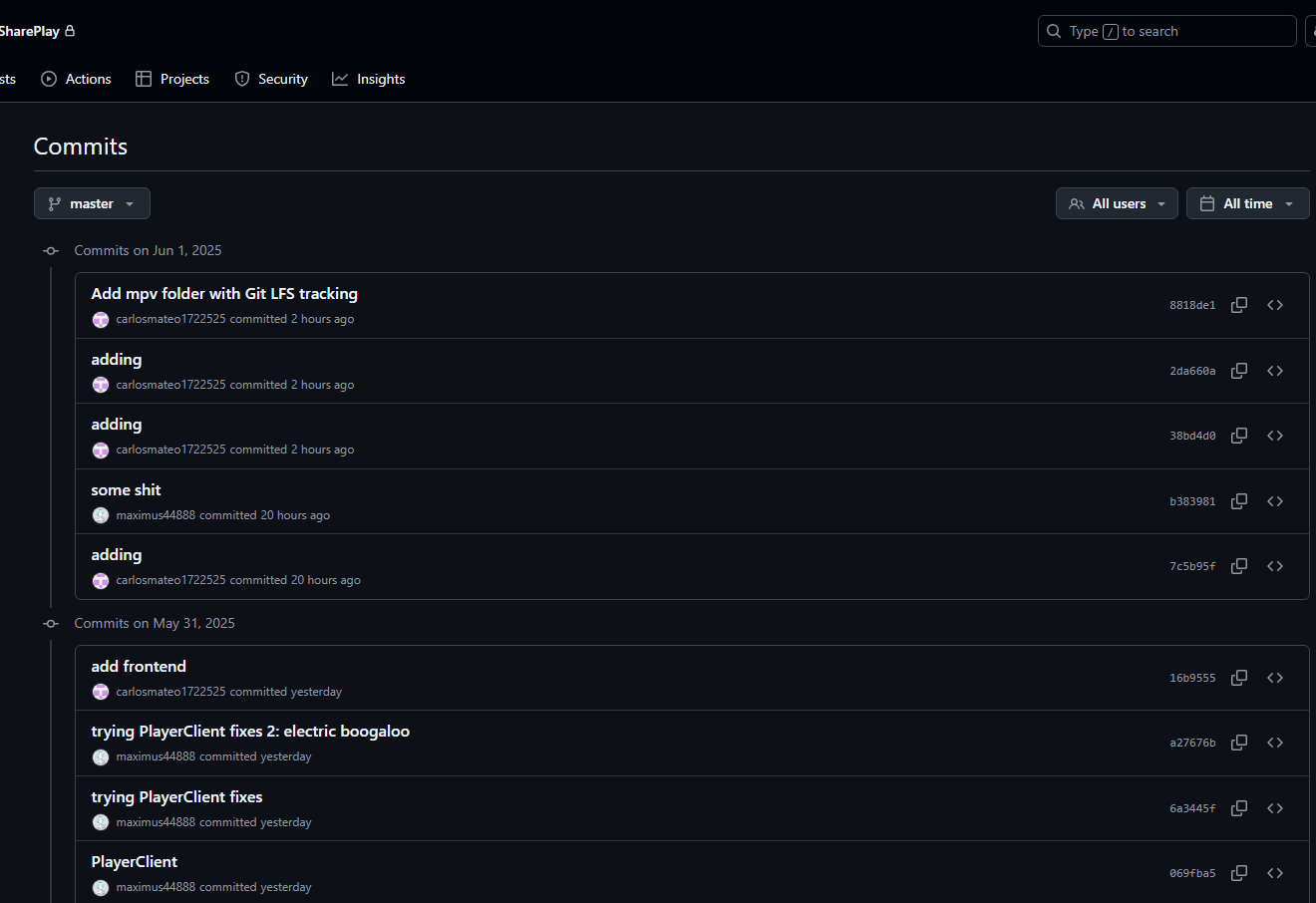
## Trabajo conjunto.

Según hemos visto anteriormente para el desarrollo de la aplicación se necesita una manera de poder llevar a cabo cada parte implicada de una manera ordenada, para ello se ha llegado a la conclusión de que la mejor opción para ello es el uso de GitHub.

Con esta herramienta lo que hemos conseguido es llevar un control de las versiones de dicha aplicación por ambas partes implicadas de una manera ordenada, el fin es la creación de un repositorio en la nube, tras crearlo con las dependencias necesarias, los dos integrantes del proyecto llevan a cabo su descarga de forma local en sus repositorios y mediante commits se actualiza el repositorio principal.

Las pull request que se llevan a cabo por cada usuario son estudiadas y discutidas mediante comentarios en el repositorio antes de mergearla a la rama principal, esto se hace para llevar un orden y consensuar los cambios llevados a cabo en el repositorio principal.

De esta manera se ha llevado a cabo un control de versiones implicando en la acción a todos los integrantes implicados en el desarrollo de la aplicación, a continuación, podemos ver un ejemplo de lo que se ha llevado a cabo.



Como podemos ver, se trata de un control de la versión y el normal desarrollo de la aplicación.

## Requerimientos de la aplicación.

Como hemos comentado anteriormente sabemos a dónde queremos llegar, pero tenemos que analizar cómo vamos a llegar a ello, para ello se consensuo que el código se llevaría a cabo con el lenguaje Kotlin, debido a su cercanía a Java y las funcionalidades extras que tienen que facilitan mucho el código.

Una vez elegimos el lenguaje más conveniente para llevar a cabo toda la lógica toco analizar los requerimientos necesarios para hacer lo que queremos, para ello lo que analizamos fue que en el backend de la aplicación necesitamos la lógica principal de servidor cliente, el servidor deberá de estar desplegado en un dominio en la web y este deberá de recibir las peticiones de diferentes clientes.

Para hacernos algo más a la idea vamos a hacer se ha hecho un croquis general de los requisitos que debe tener el cliente y el servidor.

Diagrama

El contenido generado por IA puede ser incorrecto.

Como podemos observar, el servidor estará en ejecución escuchando las conexiones de todos los clientes, el servidor tiene que poder gestionar todas las peticiones que reciba de manera concurrente, es decir, puede gestionar todas las peticiones de manera simultánea sin dejar ninguna en cola, de esta manera varios clientes pueden solicitar de manera simultáneamente la entrada y llegada al servidor.

Una vez entre al servidor, este debe de ser capaz de gestionar los clientes, los clientes se conectarán y entraran en un pasillo concurrente donde el servidor esperara un nombre de una sala, cuando lo reciba el servidor añadirá al cliente a esa sala, si no está creada la creara y si existe lo añadirá.

Una vez en la sala el servidor estará en escucha de todos sus clientes esperando mensajes de eventos, cuando recibe un mensaje de uno de los clientes este lo mandara en mensaje de broadcast a los demás miembros de la sala de manera que informara del evento que ha sucedido a los demás clientes, excluyendo al cliente del que ha recibido el mensaje de evento, para hacernos una idea, el servidor es el gestor de entradas nuevas, creación de salas, añadido de clientes a estas y finalmente mensajeros de todos los mensajes que envíen los clientes a los demás integrantes de la sala.

Para llegar a esto se ha llegado a la conclusión de que se necesita hilos virtuales concurrentes, es decir, pueden funcionar con independencia de los demás, pero al mismo tiempo puede haber comunicación entre esos hilos, la mejor forma de llevar a cabo en kotlin es mediante couruthinas, son los hilos virtuales que Kotlin nos deja a nuestra disposición para poder lograrlo, se necesita que el servidor gestione todos estos hilos simultáneamente y entre ellos, de esta manera se podrá llevar a cabo.

Para el cliente es necesario requerimientos muy parecidos a los del servidor, los clientes deben poder enviar mensajes al servidor de manera concurrente y al mismo tiempo escucharlo, esto representa dos hilos, por otra parte, el cliente tendrá el control del reproductor, por ello también debe de poder llevar a cabo varios procesos, la de enviar mensajes y recibir, al mismo tiempo que gestionar el proceso de control del reproductor.

En cuanto al control de dicho reproductor se trata de un reproductor ya desarrollado, de código libre y que tiene una amplia documentación que podemos ver en <https://mpv.io/manual/stable/>, el cliente tendría que poder escuchar en su reproductor que se ha iniciado al entrar en la sala con el video que le hemos indicado, el fin es que el cliente este observando continuamente los cambios que se hacen en la reproducción, principalmente será el pause y reanudación de la reproducción y el cambio o salto en el tiempo del video, el fin de ello es el de que cuando pase el cliente informe de ese evento al servidor enviando por el hilo del canal que tiene con el servidor.

Para llevar a cabo esto y según la documentación de mpv, se tiene que iniciar un pipe o documento temporal en el que se va registrando y escribiendo en él, cuando uno de estos eventos ocurre un hilo deberá de escribirlo en el pipe y otro leerlo para enviárselo al servidor y este informe a los demás clientes.

Teniendo en cuenta todos estos requisitos y análisis será necesario llevarlo a cabo con courothinas o hilos virtuales que nos permitan controlar independientemente cada uno de los procesos, canales de comunicación, salas, peticiones, reproductor y todos los clientes que tenemos.

Este cliente como requisito y toda deberá de ser un ejecutable java que se inicie con una interfaz gráfica y se conectará a través de internet hacia el servidor que deberá de desplegarse en la nube y en ejecución constante para las peticiones de los diferentes clientes.

## Requerimientos de la interfaz.

Para la parte grafica del cliente será la parte en el que el usuario final tendrá que interactuar con el servidor, se necesitará una interfaz sencilla en una pantalla que no será redimensionable, en esta primera vista deberíamos tener tres sencillos botones en los que dejamos las opciones por defecto, es decir, los campos vacíos para volver a rellenarlos, un botón que inicie la conexión al servidor e iniciar el mpv.

En esta primera pantalla deberemos tener unos campos obligatorios antes de poder darle a ejecutar SharePlay, se trataría de los campos de la dirección del servidor al que conectarse, el nombre de usuario que se verá en la sala y la sala a la que se tiene que conectar.

Como opcional, se le podrá pasar a la ejecución del mpv ya un fichero directamente para que empiece la reproducción y además a ello, otro field opcional en el que se le pueda pasar una ruta donde en la segunda vista ya en la sala pueda mediante esa ruta buscar todos los ficheros de reproducción dentro de ella y listarlo en una lista desplegable.

Una vez tenemos todos los campos obligatorios rellenos, podremos ejecutar con cualquiera de los dos botones que permite ejecutar shareplay, una vez hecho le mandara al cliente los datos requeridos y este ejecutara la conexión, si esta conexión es exitosa entonces deberemos dirigirnos a una nueva vista en la que se verá una listView en el que aparece la sala, los clientes, si están en reproducción o no y que archivo están reproduciendo.

En esta vista ya debería de iniciarse el reproductor, pero vacío si no le hemos pasado el fichero a reproducir o con el video si se lo hemos pasado, si no se lo hemos pasado deberemos tener un desplegable en el que nos lista los ficheros a reproducir en la ruta que le hemos pasado opcional, si no le hemos pasado ninguna ruta de ficheros entonces tendríamos que añadir nosotros mismo el fichero a reproducir en el reproductor, adicionalmente a esto como hemos hablado la vista de los integrantes de la sala y sus datos.

Como hemos comentado se ha indicado que en el caso de que queramos guardar la configuración podemos hacerlo mediante un botón, para guardar de forma local los datos de los fields, en el caso de que se guarden, al volver a iniciar la reproducción deberían de aparecer ya los datos previamente guardados, para llevar esto a cabo se ha determinado que se guardara un fichero json en el directorio del sistema del usuario en el que se guarde estos datos, se guardara la ruta de donde sacar los ficheros, el nickname, el host al que conectar y la sala donde hacerlo.

Estos datos serán guardados como hemos comentado en un JSON que se cargara a la hora de iniciar el programa en la pantalla inicial, con el botón de borrado serán borrados todos los fields para volver a poner la configuración.

Cuando ejecutemos SharePlay entonces este debería de comprobar los fields obligatorios e informar mediante una alerta de que falta información en campos obligatorios, también deberá informar de una alerta si no se ha podido llevar a cabo la conexión, indicando que es necesario que compruebe que el host name es correcto para conectarse al servidor, por ello y siguiendo con el lenguaje de Kotlin se ha decidido que la mejor forma de llevar a cabo la aplicación de escritorio es JavaFX por la versatilidad que tiene y los recursos que tiene.

# Back-end.

## Introducción.

## Servidor.

## Cliente.

## Control de reproductor MPV.

## Errores, problemas encontrados.

# Front-end.

## Introducción.

Para la interfaz de usuario se ha decidido como hemos analizado que sea lo mas sencilla posible para el uso de esta para el usuario, para ello se ha decidido hacer una primera vista en el que el usuario configurara la conexión al servidor y otros parámetros que podrá pasarle si se da el caso o no y una segunda vista donde se gestionara la sala en la que se ha unido y el reproductor MPV.

Estas vistas que son arvhivos fxml estarán controladas y gestionadas por su respectivo controlador, los controladores gestionarán la interacción con el backend mediante la inicialización del clientplayer que comienza el inicio de la sala, el reproductor y su control compartido en dicha sala junto con el resto de los integrantes de esta.

Las vistas estarán como hemos dicho compuestas de fxml, la principal se tratará de campos fields, combobox y botones de acciones seguido que gestionaran la conexión y posteriormente la carga de la otra vista, desde el controlador todo el flujo de estos datos será controlados en métodos de acción de dichos botones y comprobaciones en el que si fallan a algunas de las comprobaciones o no siguen los requisitos indicados indicaran la falta de estos.

Todo esto se ha llevado a cabo con JavaFX y su modelo vista controlador que permite llevar a cabo lo que queremos hacer que mediante el código de java controlamos las vistas del archivo FXML, también se le ha aplicado en una hoja CSS, la cual se ha añadido en la carpeta resources del proyecto donde se ha añadido a ambas vistas para gestionar la correcta visualización de los componentes y darle un aspecto mas moderno y fresco a dicha vista e interacción con el usuario final.

Los datos que se van a introducir y validar antes de llevar a cabo la conexión serán guardados en un archivo json dentro de la carpeta del usuario del sistema para guardar la configuración de una conexión y los campos requerido que al volver a iniciar la aplicación ya ocuparán los campos para llevar a cabo dicha conexión.

Finalmente se podrá volver de dicha segunda vista mediante un botón a la primera vista que lo que hará será cerrar esta segunda vista junto con el cierre de la conexión al servidor y el reproductor mpv para volver a iniciar una reproducción u otra conexión a un servidor u otro.

## Vistas.

Para las vistas vamos a explicar dichos archivos FXML en el que el usuario interactuara con la aplicación, para ello debemos explicar que tenemos dos vistas principalmente, la primera vista gestionara la conexión mediante el rellenado de dichos campos y posterior envió de dichos datos de los campos a la conexión que gestiona el backend, vamos a ver la vista a continuación.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Como podemos ver tenemos una interfaz visualmente cómoda en el necesitamos rellenar los tres primeros campos que vemos de manera obligatoria para que se pueda ejecutar el botón que ejecuta shareplay, seguidamente como vemos tenemos los campos sura al video que queremos reproducir, si no elegimos la ruta del video el shareplay se ejecutara con mpv sin ningún video, para buscar la ruta tendremos que hacerlo mediante el botón visualizar, este nos abre un explorador de archivos en el que tendremos que elegir un video para reproducir.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

También vemos un campo en el que podemos poner un texto y validar, se trata de un campo en el que tendremos que introducir un texto codificado en el cual una vez pegado al darle el botón validar lo descodificara y rellenara el campo de servidor y el de la sala por defecto, se trata de un código que un amigo u otro usuario ha copiado de su conexión para pasárselo a otra persona y haga la misma conexión y se incorpore a la misma sala donde está el otro usuario.

Como podemos ver los para llevar a cabo la conexión tenemos que rellenar los tres primeros campos antes de darle al botón, podemos ver que el primer field es un combobox.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Aquí podremos poner el uno de los hosts que nos dan por defecto o por el contrario el host donde hemos desplegado nosotros el servidor de shareplay para gestionar a los usuarios y las salas donde estos van a estar, después tenemos el campo nombre usuario y sala por defecto, debemos saber que como nombre podremos poner lo que queramos, pero en la sala si ponemos una que no existe creara una sala nueva.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Una vez rellenado todos los campos y si el servidor existe y esta escuchando entonces al ejecutar el botón de shareplay nos llevara a la siguiente vista, si no se puede llevar a cabo la conexión al servidor saldrá una alerta ERROR en el que indicara que no se ha podido llegar al servidor.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Una vez en la sala como hemos indicado se iniciará también el reproductor MPV con el video si le hemos pasado el campo del video para que se inicie junto con la conexión o en el caso de que no lo haya hecho se iniciara sin ningún video a la espera de que le pasemos un video para iniciar la reproducción.

Como hemos indicado aquí estarás en la sala y podemos esperar a los demás integrantes de la sala o cerrar la conexión con el servidor dándole al botón volver, también como hemos indicado podemos copiar la conexión actual para que otro usuario se conecte a las salas y sincronizar sus dos reproductores.

Cuando se conecten los demás usuarios veremos que de forma local cada uno de ellos vera a los demás integrantes de la sala.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Vemos a lo que vería cada usuario en su equipo local, ahora vamos a ver que lógica llevan los dos controladores de cada vista para llevar a cabo lo que estamos haciendo.

## Controladores.

Los controladores lo que hará será controlar los datos que le pasemos por dichas vistas y controlar los datos, debemos tener en cuenta que, en la primera vista, por cada botón tenemos un método que dentro gestiona dicha lógica para llevar a cabo lo que se indica y queremos llevar a cabo.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Como vemos tenemos 4 botones, lo demás son fields que gestionamos a través de estos 4 botones y un método al inicio que comprueba si existe el archivo de configuración existe y si es así carga los tres primeros campos con lo ya guardado, en el botón de ejecutar shareplay al clickar se ejecutara la conexión y se valida que los 3 campos estén rellenos, si no lo están saldrá el siguiente error de validación.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Como vemos si no se ha rellenado entonces nos indicara un error de validación, adicionalmente a esto, en el caso de que en el campo de dirección del servidor haya una string, es decir, no este vacío, pero el texto de dentro no permita la conexión al servidor entonces saltara una ventana de error indicando que no se ha podido conectar al servidor.

Cuando se conecta entonces este método le pasa el socket de conexión del servidor a la siguiente vista junto con la string del campo de la ruta al fichero a reproducir, en la siguiente vista se gestionará en este primer método la conexión al servidor, pasándole al método de comienzo del programa el socket, el path mpv, el nickname y la sala donde quieres incorporarte.

Siguiendo con esta vista, el controlador de visualizar al ejecutarlo se abre un explorador que requiere seleccionar un archivo de reproducción de video, al seleccionarlo entonces se cerrará el explorador y se pondrá el path junto con el archivo en el field para pasárselo al siguiente controlador.

El botón validar descodificara el código introducido en el field y después de descodificarlo pondrá en cada campo el parámetro descodificado, se rellenaran los campos de host y de sala a la espera de que se rellene el campo de nombre o ya esta relleno guardado de la ultima configuración, al iniciar la conexión recordemos que cada vez que se valide y se lleve a cabo la conexión entonces se guardara la ultima configuración, siempre lo hará y al iniciar la aplicación se cargara la última conexión establecida.

El restablecer configuración lo que hará será borrar dicho archivo que guarda en JSON la última conexión establecida, este lo borra y limpia todos los campos del formulario para volver a empezar a rellenar.

Como se ha indicado, después de validar los campos del formulario y validar la conexión con el socket y su formato, entonces se guardará la configuración actual, mediante una clase llamada config y salvará dicha configuración en el JSON para cargarla en la próxima configuración, también se le pasará el socket a la siguiente vista junto con los datos del campo de filepath, después de hacer esto lo que hará será destruir la vista e iniciar la siguiente vista con los datos que le hemos pasado.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

En este controlador de la vista lo que tenemos que indicar es que al iniciarse hay un primer método que lo que hace es recibir el socket e iniciar la conexión con el servidor pasándole todos los parámetros e iniciando el reproductor mpv con o sin un fichero y le indica al servidor quien es y la sala donde conectarse, seguidamente mediante un método en el que se lleva a cabo la conexión se recupera un array con los integrantes de la sala.

En dicho array se le aplica un observable que observa cualquier cambio en dicha lista y los imprime en la listview, de esta manera se imprime de continuamente los integrantes de la sala y se actualiza constantemente.

En esta vista estamos en la sala con los integrantes y con el reproductor mpv iniciado y con el video que o bien se ha iniciado con la ruta que le hemos indicado en la vista anterior o bien lo hemos puesto nosotros después, dicho mpv está en el ejecutable que tenemos como cliente, este la primera vez que se inicia la reproducción lo saca a una ruta del sistema y guarda la ruta, si esta no existe entonces lo volverá a hacer, de manera que se ejecuta siempre de ahí cada vez que se inicia la reproducción del programa.

Aquí empieza el control sobre el reproductor y lo que se esta viendo de manera simultanea en todos los integrantes de la sala, es decir, que se sincroniza la reproducción, la pausa y los saltos de video.

Al indicar volver entonces destruirá toda la vista, cerrará la conexión para que el servidor para que quite el cliente de la sala y cierre la conexión con él, se volverá a la vista principal de conexión donde cargara esta última conexión que es la que hay guardada en el archivo JSON de configuración.

En el botón de copiar lo que hace es gestionar el código, genera un código codificado en el que se copia en el portapapeles, esto lo hace con el fin de pasárselo a otra persona para que el otro usuario lo ponga en el campo de copiar otra configuración y entonces validarlo y que se ponga esa configuración, esto se hace con el fin de que sea fácil invitar a otros usuarios a la misma sala.

Con esto tenemos el control de toda la sala, vamos ahora a ver y especificar como funciona el control de los archivos de configuración que guardamos en el sistema para cargar y guardar la última configuración realizada.

## Persistencia y control de JSON.

Como se ha indicado, cada vez que se inicie el programa este cargara una configuración en el caso de que exista rellenando los campos obligatorios de la última conexión establecida.

Para ello la gestión se ha llevado a cabo mediante una clase de gadgets que gestiona archivos JSON, se ha creado una clase data kotlin que tiene estos 3 campos y cuando se llama a la clase savedata se guardan los datos que se le pasan de los 3 fields y se salvan en el archivo.

Esto se guarda en un archivo .configuracion.json en los archivos del usuario del sistema, si este archivo no existe lo crea con la estructura y los datos y los carga cuando se le llama al config para recuperar los campos que sean necesarios.

De esta manera hemos guardado las ultimas conexiones establecidas con el servidor para que sea más cómodo conectar con dicho servidor en las próximas conexiones.

Cuando se borra la configuración y se deja dicha configuración por defecto de fabrica de inicio por primera vez la aplicación lo que hace es borrar dicho archivo para librar recursos y este no sea redundante, de igual manera se lleva a cabo de una manera parecida con el reproductor MPV y sus archivos de ejecución que están dentro del ejecutable, cuando se inicia este saca los archivos necesarios para iniciar el reproductor en una ruta del sistema del usuario para usarla para las próximas veces y usar dicha ruta para las siguientes conexiones, de esta manera no usa un archivo temporal cada vez que quiere usar dicho reproductor para iniciar un video y no redunda los datos.

Esto se lleva a cabo con una dependencia que controla JSON y nos permite mapear un archivo de este tipo a los datos que buscamos abstrayéndonos de la complejidad de tener que buscar en el documento el campo o key para recuperar el valor.

## Errores, problemas encontrados.

Durante el desarrollo del frontal se tuvo que decidir qué datos tenían que ser obligatorios o no y como validarlos, para hacer más sencilla la aplicación se tuvo que pensar que la aplicación solo controla un reproductor y tiene su control por código para ello, esto generaba un problema en el que hay que pasarle una ruta de ejecución de un reproductor, pero la aplicación solo gestiona un tipo, que es el MPV.

Para solventar este problema se acabo decidiendo meter el ejecutable de MPV y todos los archivos necesarios dentro del ejecutable y a la hora de ejecutar dicho MPV y pasarle la ejecución al playerclient para iniciar el control del reproductor y comenzar a ver el video, para ello lo que se ha hecho es como hemos indicado guardar el MPV en el ejecutable.

Después de esto lo que hace el programa al iniciar la reproducción es guardar dichos archivos en una ruta del sistema del usuario y después usa esta ruta para reproducir el video e iniciar el control del reproductor.

También surgió el problema de saber cómo teníamos que tratar los campos y los datos para hacerlos persistentes, ya que no tenemos una gestión de usuarios mediante una BBDD que gestione datos persistentes, pero para hacer una gestión mas intuitiva de la aplicación para el usuario final teníamos que guardar al menos los datos de la última configuración.

Para llevar esto a cabo se tomó la decisión de guardar los datos en un archivo JSON de manera que se guarda las ultimas configuraciones de conexión de manera persistente en el mismo usuario del sistema que lleva a cabo la ejecución del programa, de esta manera su configuración se guardara en el mismo archivo de manera persistente y en el caso de borrar y dejar por defecto se borrara el archivo liberando recursos.

Otro problema que se observo es que solo controlamos un reproductor, pero a la instancia del objeto que controla todo el reproductor y la conexión al servidor, de esta manera hay que pasarle al objeto la ruta donde se encuentra la ejecución del reproductor, se considero pasarle esta ruta por un campo en la primera pantalla de carga, pero puede inducir a problemas porque puede que el usuario le pase la ruta de otro reproductor que no controla el programa.

Por esto se determino que en el ejecutable se guardaría el reproductor dentro de las carpetas del proyecto y se llevo a cabo una lógica en la que en el momento de la ejecución del programa y la conexión el reproductor se ejecutase en un archivo temporal, esto hizo otro problema que es que por cada vez que se ejecutase la reproducción o la conexión se creaba un archivo temporal lo que hacia que se llenase de archivos temporales en el sistema del usuario.

Tras esto, se llevó una solución en el que el programa a la hora de llevar a cabo la ejecución del programa y se ejecutase el reproductor se comprobaría que en los archivos del sistema del usuario existiese el programa en cierta ruta dentro de los archivos del sistema, si no existe este guardaría de los archivos del programa el reproductor en esta ruta y siempre usaría esta ruta, de esta manera se guarda en un sitio único una sola vez y se usa esa ruta y ese reproductor con cada ejecución resolviendo el problema de llenado de memoria a la hora de usar el programa.

Otro problema que se encontró es la validación de los campos, para ello se determino que los tres primeros campos deben ser requeridos, ya que son los campos obligatorios y necesarios para llevar a cabo la conexión con el servidor, para validar que dichos campos estén llenos el programa no deja ejecutar la conexión si estos están en estado empty, de esta manera se obliga rellenar los campos.

Adicionalmente a esto el primer campo que es el campo del host es necesario que sea un campo correcto y que se pueda llevar a la conexión a dicho host, para eso se determino que en esta vista se tenia que validar el socket, por ello se instancia y prueba conexión al socket en el que se intenta conectar, en caso de que no sea una ruta valida o sea valida pero no se pueda conectar con el saltara una alerta en el que indica que es incorrecta la conexión o el formato de servidor.

Una vez hecha esta validación si es correcta junto con el campo de path para el file de reproducción entonces se enviaría al controlador de la segunda vista para tratar estos campos como se ha indicado anteriormente.

# Conclusión.

En conclusión, se ha llevado a cabo una aplicación funcional en el que el propósito es que un grupo de personas que quieren ver una película o video de forma local pero sincronizados entre ellos se ha llevado a cabo y funciona a la perfección.

Esta aplicación es útil para poder sincronizar los reproductores locales y su reproducción junto con los demás integrantes de la sala, es útil, de sencillo uso e intuitiva que invita a su uso, pese a esto, la aplicación tiene margen de mejora y evolución, que es lo que se ha pensado y la intención que se tiene también.

Si bien esta primera versión de la aplicación es totalmente funcional y cumple con los requisitos necesarios para llegar a lo que se pretende, se ha pensado en las siguientes versiones, por ejemplo, como se ha indicado, esta aplicación controla un reproductor el reproductor MPV, pero se pretende que dicha aplicación controle de igual manera que hace con el MPV cualquier reproductor, lo que se ha pensado para mejorar dicha aplicación es que en el futuro se le pase la ruta de ejecución de cualquier reproductor y el resultado final sea el mismo que hace actualmente pero con cualquier reproductor que se pueda pasar.

Otro margen de mejora en el que se trabajara en la siguiente versión es en que actualmente la aplicación controla el reproductor pero este puede está reproduciendo cualquier video en el reproductor de cada usuario de forma local, para mejorar esto, si bien se puede añadir al reproductor cualquier video que se quiera reproducir, se ha pensado en una mejora en el que al la hora de incorporarte en la sala, si no eres el primero se le pase al nuevo integrante a su aplicación el video a reproducir y el programa busque el video con el mismo nombre en una ruta que se le pasa al programa para que busque ahí todos los videos a la hora de hacer esto de esta manera iniciara la reproducción con el mismo video de los integrantes si existe en el directorio que gestiona la aplicación, si no es así entonces se deberá añadir el video a mano como hasta ahora en la aplicación.

También se ha pensado que si bien ves a todos los integrantes de la sala en la segunda vista y puedes saber quien esta contigo en dicha sala, no se puede hablar con ellos, se pacta que ver con los integrantes de la sala de manera externa, por lo que se ha pensado en añadir un chat en la segunda vista para hablar con los integrantes y pactar todo esto en la misma aplicación abarcando así más control e interacción entre los usuarios en la misma aplicación sin la necesidad de necesitar herramientas externas.

Actualmente la aplicación se ejecuta mediante un ejecutable .jar que necesita la maquina java para funcionar, también se ha pensado en versiones posteriores para crear ejecutables que funcionen directamente en el sistema operativo anfitrión, se ha pensado en Windows y Linux.

Por lo tanto, actualmente funciona tal y como se pretende, pero estamos ante la fase temprana de la aplicación, tiene margen de mejoras o actualizaciones que lleven a otro nivel esta aplicación que si bien funciona y hace lo que se le requiere, puede suavizar su uso y versatilidad con versiones posteriores.

Como se ha indicado, la aplicación cumple sus requisitos, pero como debería de pasar con cualquier aplicación durante su desarrollo se ha pensado en mejoras y versiones 2.0 posteriores a la actual, para esto también se pensó en el momento de desarrollo de la aplicación y por ello se llevo a cabo un desarrollo generalizado en el que el código esta modularizado y abstraído de tal manera que invita a llevar a cambios en el mismo de una manera mas sencilla y escalable, precisamente pensando en esto.

Tanto el cliente como el servidor se ha llevado a nivel de código atendiendo a la versatilidad del código de manera que invite a facilidad de cambios y nuevos aplicativos de funcionalidad en la aplicación a la hora de necesitarlo, el servidor, que esta desplegado de manera sencilla en una url en internet o bien en el pc de algún usuario en el que se puede comenzar la ejecución del servidor y conectarte a ese servidor por internet hace un despliegue versátil en el que cualquier persona puede desplegar este de manera sencilla, haciendo posible que los usuarios o bien utilicen servidores comunes desplegados ya o bien estos mismo desplieguen el servidor mediante el ejecutable y el cambio de unas cuantas configuraciones del router en su red para poder usarlo.

Se ha creado un repositorio en la nube mediante GitHub para llevar a cabo un control de versiones escalonado y estudiado, esto facilita la escalabilidad de dicha aplicación, con ellos se ha llevado un correcto estudio de los requisitos de la aplicación y se ha atendido en paralelo en el desarrollo a las limitaciones actuales de dicha aplicación a la par que su crecimiento, atendiendo a esto y pensando en versiones posteriores.

Todo esto ha desembocado en una aplicación que cumple con todo lo anterior descrito y necesario como requisitos de aplicación, es una buena aplicación que invita a juntar a las personas en la distancia para poder ver una película o cualquier video separados, pero al mismo tiempo a la vez de manera local en sus casas.