Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Desarrollo de una Aplicación Móvil Capaz de Retransmitir en Directo Contenido Multimedia y en Vivo

Autor: Jorge Cordobés Delgado

Tutor: Raúl Alonso Calvo

Madrid, junio 2022

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Desarrollo de una Aplicación Móvil Capaz de Retransmitir en Directo Contenido Multimedia y en Vivo
Junio 2022

Autor: Jorge Cordobés Delgado

Tutor: Raúl Alonso Calvo
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este proyecto recoge el trabajo de investigación y desarrollo que hay detrás de Panóptico, una aplicación móvil para Android que permite compartir contenido multimedia y en vivo con aquellos dispositivos que se encuentren conectados a una misma red local. Esta aplicación nos ofrece la posibilidad tanto de compartir archivos multimedia que se encuentran almacenados en el dispositivo, como la de retransmitir en vivo la cámara o la pantalla del teléfono, pudiendo incluso interactuar con esta última desde otros dispositivos.

A diferencia de otras opciones que podemos encontrar en el mercado tecnológico, en este caso no resulta indispensable la posesión o no de un dispositivo Android que cuente con la aplicación instalada para la visualización del contenido. También permite reproducir contenido en aquellos dispositivos compatibles con la tecnología UPnP, como es el caso de una televisión con DLNA; un dispositivo con la tecnología Miracast, como es el caso de los dispositivos Chromecast; o incluso desde un navegador si lo que se quiere visualizar es una retransmisión en vivo compartida a través de la tecnología WebRTC, con la que queda implementada esta última funcionalidad.

Abstract

This project collects the research and development work behind Panóptico, a mobile application for Android that allows you to share multimedia and live content with those devices that are connected to the same local network. This application offers us the possibility of sharing multimedia files that are stored on the device, as well as broadcasting the camera or the screen of the phone, being able to even interact with the latter from other devices.

Unlike other options that we can find in the technological market, in this case the possession or not of an Android device that has the application installed for viewing the content is not essential. It also allows you to play content on those devices that are compatible with UPnP technology, such as a television with DLNA; a device with Miracast technology, such as Chromecast devices; or even from a browser if what you want to watch is a live broadcast shared through WebRTC technology, with which this last functionality is implemented.

Índice

1 Introducción

1.1 Motivación y necesidad del proyecto

Debido al auge que hay en el uso de plataformas para consumir contenido multimedia o hacer transmisiones en vivo, como son el caso de Netflix y Twitch, nace la idea de desarrollar Panóptico, una aplicación móvil capaz de esto mismo. Esto es, que te permita visualizar o retransmitir contenido que abarque desde los archivos multimedia almacenados en el dispositivo, hasta la posibilidad de compartir el contenido en vivo emitido por la cámara o pantalla del teléfono (*Screen Mirroring* [1]).

Sin embargo, a diferencia de otros servicios, el uso de esta aplicación está enfocado en la retransmisión del contenido mencionado a los dispositivos conectados a la misma red, por medio de un servidor multimedia que ofrezca el contenido, o bien haciendo uso del protocolo UPnP (*Universal Plug and Plugin*) [2] a través de la tecnología DLNA (*Digital Living Network Alliance*) [3] en dicha red. Esto permite que no sea necesario enviar el contenido multimedia a aquellas personas con las que se desea compartir, evitando además que ocupe espacio en sus respectivos dispositivos, si no que únicamente pueda ser accesible en un momento determinado, a través de la aplicación u otros dispositivos compatibles.

Algunos de estos dispositivos para la reproducción del contenido serían: un smartphone con la aplicación desarrollada, una televisión inteligente, un ordenador, un dispositivo Chromecast, entre otros.

Por tanto, esta aplicación está destinada tanto para uso personal como para acontecimientos en vivo con el fin de facilitar la comunicación frente a posibles factores que a veces impiden disfrutar por completo de estos. Algunos de los ejemplos en los que pueden ser necesario el uso de esta aplicación, serán desarrollados a continuación:

- Aunque existen plataformas para retransmitir las clases, como Zoom, Teams o Blackboard, cuyo uso ha tenido un crecimiento exponencial durante los meses de pandemia por la COVID-19, en el caso de la Universidad Politécnica de Madrid dejaron de usarse tras la vuelta a la actividad presencial completa desde el 4 de octubre de 2021 [4]. Esto es debido a que es una universidad que imparte sus clases completamente presencial. Sin embargo, durante el periodo en el que se impartían las clases de forma semipresencial, aquellos que asistían de forma presencial hacían uso de las bondades que tiene el hecho de que la clase fuera retransmitida para los alumnos que asistían de forma online. A todos aquellos que se sitúan en la parte de atrás del aula o tienen dificultades para seguir la clase, como puede ser visualizar el contenido de la pizarra o escuchar al profesor, la posibilidad de poder unirse a la retransmisión hacía que fuera más fácil seguir las explicaciones del profesor.
  Por ello, la aplicación podría ser una herramienta válida para que el profesor que imparte la clase, pueda retransmitir el contenido de la pizarra o algún archivo que pueda ser útil para los alumnos durante la sesión. Todo esto, sin necesidad de hacer uso de plataformas externas y únicamente a aquellos que se encuentren en el aula ya que sus dispositivos deben estar conectados a la misma red.

- Del mismo modo, se puede aplicar el caso anterior a una conferencia, a las cuales suelen asistir un gran número de asistentes y por factores como el ruido del ambiente o ubicación del asiento, a veces puede ser difícil disfrutar de esta. Haciendo uso de la aplicación, se podría emitir la conferencia o mandar la presentación usada de forma privada únicamente a los asistentes, de manera que estos puedan en todo momento ver y escuchar sin perder ningún detalle.
- Otro de los posibles casos, sería por ejemplo en un evento con aforo limitado, como puede ser una graduación, en la que puede haber algún caso que algún familiar o conocido de los graduados tenga que quedarse fuera. Al poder hacer uso de la aplicación, cualquier asistente que haya conseguido entrar, podrá retransmitir el evento en vivo a aquellos que no hayan tenido la oportunidad, y sin que se salga del alcance al hacerse dentro de una red local.
- Una de las principales motivaciones de llevar a cabo este proyecto, era facilitar la visualización en eventos tales como conciertos o competiciones deportivas. Este tipo de eventos normalmente requieren pagar una entrada para acceder y no siempre te ubica en el lugar deseado, pues suelen ser en pista o desde una grada con asiento numerado. Si no dispones de un buen asiento o hay alguien delante que no te permite visualizar bien el evento, hace que la experiencia no sea tan satisfactoria como esperas. Y no en todos los lugares donde se celebran estos eventos disponen de monitores con los que seguir del evento. Por esta razón, el disponer de una aplicación que facilite esta retransmisión a todos aquellos que se vean afectados por los inconvenientes mencionados, hace que puedas disfrutar del evento que has pagado sin sufrir los inconvenientes mencionados.
- Ver las fotos y vídeos de un viaje que has capturado con el móvil en una televisión sin hacer uso de cables o dispositivos externos, usar un monitor junto con la aplicación para hacer un *video wall* [5] casero en un establecimiento, retransmitir una película desde el móvil sin necesidad de tener instalada la aplicación de la plataforma a usar en el dispositivo que la quieres reproducir como ocurre en algunos casos con la tecnología Chromecast, hacer uso de un televisor inteligente para mejorar la visualización de un evento que se celebra al aire libre, son otros de los muchos casos en los que se puede hacer uso de la aplicación.

Como se puede apreciar en la gran mayoría de los ejemplos descritos anteriormente, lo que se trata es de preservar principalmente la privacidad al no tener que hacer uso de plataformas externas, ya que el contenido permanece siempre dentro de una red local. Del mismo modo, otra de las principales ventajas es lo fácil y rápido que es transmitir dicho contenido, sin necesidad de pasar por servidores externos o que ocupe memoria en los dispositivos cuando se recibe, ya que este contenido se emite y se visualiza en vivo.

1.2 Objetivos

El objetivo principal del trabajo será facilitar la retransmisión o visualización de contenido multimedia dentro de una misma red. Este contenido multimedia abarca desde los archivos multimedia almacenados en el dispositivo, hasta la posibilidad de compartir el contenido en vivo emitido por la cámara o pantalla del teléfono (*Screen Mirroring*).

Para ello, usaremos un dispositivo Android, para el que se desarrollará la aplicación, como servidor de estos contenidos a aquellos dispositivos que se encuentren, al estar conectados a la misma red, mediante el protocolo UPnP (*Universal Plug and Play*). Algunos de los dispositivos compatibles serían: un smartphone con la aplicación desarrollada, una televisión inteligente, un ordenador, un dispositivo Chromecast, entre otros.

Además, se trata de usar protocolos como el DLNA (*Digital Living Network Alliance*), con el que por ejemplo ya es compatible con una televisión inteligente. De esta forma, se evita se requiera, en caso de no ser necesario, de dispositivos adicionales como un Chromecast que hace uso del protocolo de Google Cast, una mezcla de conceptos de DLNA y Miracast.

Por consiguiente, los objetivos que se desean alcanzar con este proyecto, son los siguientes:

- Objetivo principal:
  - Crear una aplicación que ofrezca las funcionalidades necesarias para retransmitir contenidos multimedia y en vivo, además de poder reproducirlos en aquellos dispositivos que esté instalada o bien sean compatibles para ello.
- Objetivos secundarios:
  - Creación de un servicio web capaz retransmitir por el protocolo UPnP, el cual incluye la tecnología DLNA, a otros dispositivos de la red y otro capaz de retransmitir por web *streaming*.
  - Desarrollar las funcionalidades de descubrimiento y publicación de dispositivos UPnP, así como los métodos y servicios necesarios para reproducir los distintos contenidos del móvil desde otros dispositivos compatibles que se encuentren en la misma red.
  - Implementar la funcionalidad para poder retransmitir el contenido mencionado anteriormente, de manera que pueda ser codificado de forma correcta para su posterior decodificación y visualización.
  - Despliegue y pruebas del sistema y la aplicación.

1.3 Tareas y planificación

1.3.1 Definición de tareas

En base a los objetivos que se desean alcanzar, las tareas a realizar no han cambiado y siguen siendo las siguientes:

1. Análisis del estado del arte y posibles soluciones: 50 horas.
2. Diseño de la arquitectura del sistema mediante el uso de diagramas que lo representen: 20 horas.
3. Diseño del modelo de datos empleando un modelo acorde al sistema: 24 horas.
4. Implementación de las funcionalidades de la API en el servidor: 130 horas.
5. Definición de las pruebas del sistema y realización de estas, documentando los errores y las mejoras encontrados: 5 horas.
6. Corrección de errores e implementación de mejoras encontrados en el sistema: 2 horas y media.
7. Diseño de la aplicación y la interfaz de las posibles pantallas que incluya: 3 horas.
8. Desarrollo de la aplicación y su interfaz: 7 horas.
9. Integración de los métodos de la API en la aplicación: 5 horas.
10. Definición de las pruebas de la aplicación con la integración del sistema y realización de estas, documentando los errores y las mejoras encontrados: 5 horas.
11. Corrección de errores e implementación de mejoras encontrados en la aplicación con la integración del sistema, para garantizar así, la calidad del proyecto: 2 horas y media.
12. Creación de la memoria y análisis de las conclusiones del proyecto y futuras líneas de desarrollo: 60 horas.
13. Preparación de la presentación del proyecto: 10 horas.

1.3.2 Diagrama de Gantt

A continuación, podemos observar cómo se han distribuido las tareas definidas mediante un diagrama de Gantt:

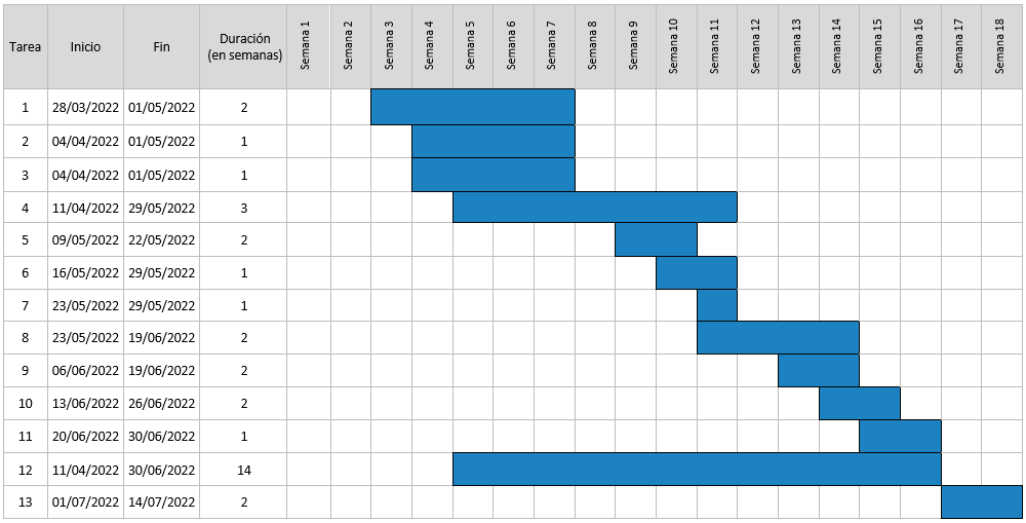

Ilustración 1. Diagrama de Gantt.

1.4 Estructura de la memoria

El resto de la memoria se estructura de la siguiente forma:

- El capítulo 2, "Estado del arte y tecnologías empleadas", describe brevemente las alternativas y tecnologías más adecuadas que se han seleccionado para el desarrollo del proyecto.
- En el capítulo 3, "Análisis de requisitos", se enumeran y detallan los requisitos necesarios para el desarrollo de la aplicación.
- En el capítulo 4, "Desarrollo de la aplicación", por un lado, se habla de las líneas de diseño que se han seguido tanto a nivel de software, como visuales. También se explica en detalle la implementación llevada a cabo de las distintas funcionalidades de la aplicación.
- En el capítulo 5, "Pruebas y evaluación", se incluyen las pruebas y evaluación de estas, además de cómo es la usabilidad de la aplicación.
- En el capítulo 6, "Resultados, conclusiones y líneas de futuro", se hace una valoración de los resultados obtenidos y se llegan a unas conclusiones.
- En el capítulo 7, "Análisis de impacto", se hace un análisis de impacto tanto a nivel personal como nivel social en sus distintos campos.

2 Estado del arte y tecnologías empleadas

2.1 Estudio de aplicaciones similares en el mercado

Dentro del mercado, se pueden encontrar un gran número de programas y aplicaciones compatibles con distintos sistemas operativos, que ofrecen soluciones similares a las que se plantea con la aplicación que se desea desarrollar. Sin embargo, muchas de ellas se centran únicamente en compartir únicamente contenido multimedia o hacer retransmisión en vivo. También carecen de funcionalidades, como poder hacer un *streaming* en la que puedan conectarse varios dispositivos a la vez, o poder elegir entre si se quiere compartir la pantalla del dispositivo o las distintas cámaras que este posea. En el caso de los programas de escritorio, requieren hacer uso de un ordenador y pierde la portabilidad y accesibilidad, como sí ofrece un smartphone en la actualidad. Otro detalle diferencial es la compatibilidad de estas soluciones para que pueda usarse en dispositivos y sistemas operativos diferentes al que se ha usado para implementarlas.

En definitiva, no hay una solución que permita compartir contenido multimedia y retransmitir contenido en vivo, y que puedan recibirlos distintos dispositivos con distinto sistema operativo al empleado para desarrollar esta aplicación, sin hacer uso de la misma aplicación que se emplea para generar estos servicios. No obstante, en las siguientes secciones de este apartado, se recogerán algunas alternativas en las que se analizará sus ventajas e inconvenientes ante la solución que se plantea desarrollar que unifique todo en uno.

2.1.1 Plain UPnP

PlainUPnP [6] es una aplicación que ofrece tanto la reproducción de contenido multimedia recibido, como fotos, vídeos y música.

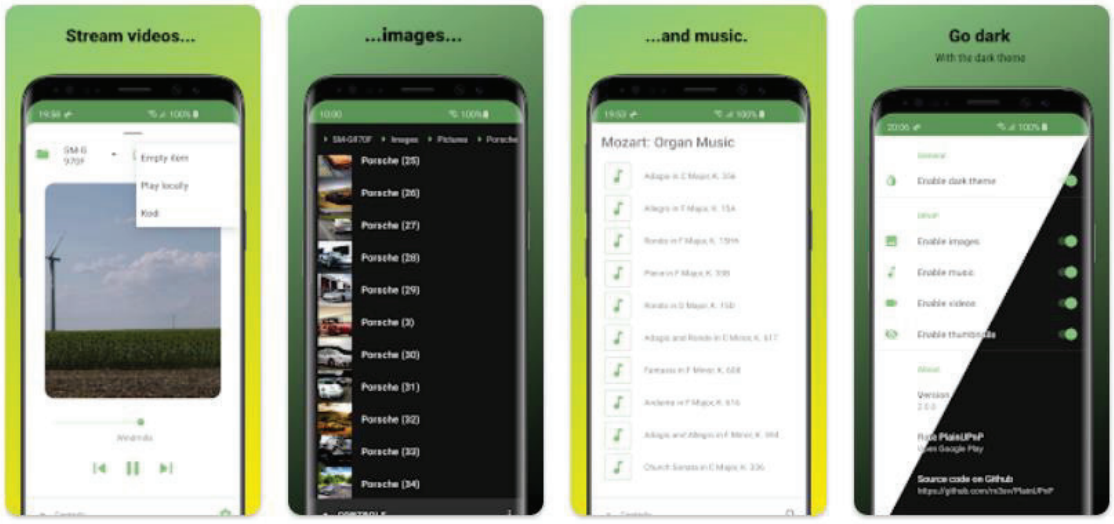

Ilustración 2. Capturas de pantalla de la aplicación Plain UPnP. Fuente: Google Play Sotre.

También da la posibilidad de enviar este contenido, actuando como servidor UPnP. Esta aplicación se basa en un proyecto mencionado en el siguiente apartado de proyectos similares. Sin embargo, no es capaz de retransmitir el contenido capturado en vivo, por lo que, a pesar de ser una buena opción, no cumple con todos los requisitos que quiere abarcar este proyecto.

2.1.2 Screen Mirroring

Screen Mirroring [7] es una aplicación que se centra en la transmisión del contenido en vivo de la pantalla del teléfono a cualquier dispositivo reproductor de este contenido como puede ser televisores inteligentes, un dispositivo Chromecast o receptores de DLNA.

Ilustración 3. Capturas de pantalla de la aplicación Screen Mirroring. Fuente: Google Play Sotre.

Aunque sirve como servidor de contenidos, no es capaz de reproducirlos desde otro móvil o de mandar contenidos multimedia. Por esta razón, se descarta como posible alternativa a la aplicación que se desarrolla en este proyecto.

2.1.3 Duplicar Pantalla

Duplicar Pantalla [8] es una aplicación para iOS, la cual permite duplicar la pantalla del teléfono en dispositivos Chromecast, televisores inteligentes e incluso navegadores web.

Ilustración 4. Capturas de pantalla de la aplicación Duplicar Pantalla. Fuente: Apple App Sotre.

Pese a que es una buena opción, para retransmitir contenido en vivo de la pantalla, no ofrece la posibilidad de usar la cámara directamente como fuente de vídeo en la retransmisión. Al igual que la anterior, tampoco permite enviar contenido multimedia, pues su función únicamente es retransmitir la pantalla del dispositivo en vivo.

2.2 Análisis de tecnologías empleadas

En el siguiente apartado, se realizará un análisis de las tecnologías empleadas en el proyecto, así como aquellas que finalmente se han descartado por diversas razones.

2.2.1 Por qué una aplicación para Android

Ilustración 5. Logotipo de Android. Fuente: Wikipedia.

Durante los últimos años, aunque la cuota de mercado de iOS frente a Android en EEUU es de casi un 20% superior, la cuota de mercado de usuarios que usan Android es de casi un 50% superior según los datos recogidos en Statcounter [9].

Esto se debe a lo accesible que es la gama de entrada en Android al contar con diversos fabricantes que usan este sistema operativo. Si nos vamos por ejemplo a la tienda de Realme, una marca que está emergiendo por sus teléfonos que cuentan con buenas prestaciones a un precio asequible, encontramos un teléfono Android con un precio inferior a 100€ [10]. Si, por el contrario, quieres comprarte el último móvil de gama de entrada de Apple, el precio en España alcanza un precio de 529€ desde la tienda oficial [11].

Muchos de los usuarios Android, no solo por el precio de los dispositivos, lo prefieren frente a un dispositivo de Apple debido a su gran y amplía variedad de aplicaciones gratuitas en el Google Play Store.

También se ha elegido desarrollar una solución para móvil frente a un programa de escritorio, debido la tendencia que ha habido en los últimos años de usar un smartphone como ordenador para las tareas diarias. Esto se debe principalmente a la portabilidad, pero también a la comodidad de disponer de todo lo esencial, en un único dispositivo.

Si nos centramos en el desarrollo de aplicaciones de forma nativa, en Android pueden ser desarrolladas en Java, un lenguaje de programación muy extendido y que la gran mayoría de programadores conoce.

Por todas estas razones, se ha decidido partir el desarrollo de esta aplicación en Android, abriendo una posibilidad a que, en un futuro, también sea desarrollada para iOS.

2.2.2 Entorno de programación y lenguajes

Dentro de esta sección se analizará y argumentará sobre la elección del IDE (Integrated Development Environment) de desarrollo, así como la elección del lenguaje de Java frente a Kotlin.

2.2.2.1 Android Studio

Ilustración 6. Logotipo del IDE de Android Studio. Fuente: Wikipedia.

El hecho de desarrollar una aplicación para Android de forma nativa, hace indispensable hacer uso del entorno de programación para Android Studio [12], el cual ofrece todas las herramientas necesarias para ello.

Este IDE, basado en el software IntelliJ IDEA de JetBrains, está disponible para todos los sistemas operativos para ordenadores, entre los que se incluye Windows 10, el sistema operativo que se ha utilizado para realizar este proyecto.

Se trata un potente editor de código con edición inteligente y refactorización de código. También cuenta con la posibilidad de instalar diversos emuladores con la versión del sistema operativo Android que se prefiera para ejecutar la aplicación desarrollada. Tiene un soporte de compilación basado en Gradle, plantillas para crear diseños comunes en Android y un editor de diseño enriquecido para arrastrar y soltar componentes de la interfaz, entre otras funciones. Admite lenguajes de programación como Java y Kotlin, así como otros lenguajes de programación de IntelliJ.

Por tanto, el usar este IDE es lo más apropiado si lo que se quiere es desarrollar aplicaciones para Android de forma nativa.

2.2.2.2 Java frente a Kotlin

Ilustración 7. Java vs. Kotlin. Fuente: Analytics India Magazine.

Dentro de los lenguajes de programación usados para desarrollar aplicaciones para Android, se encuentran Java y Kotlin. Debido a mi carencia de conocimientos con Kotlin al aparecer en 2016, frente al amplio uso durante la carrera y conocimiento del lenguaje de Java, me he decantado por este último. También se ha elegido este lenguaje debido a la gran cantidad de recursos y soluciones a problemas detectados, en cuanto al desarrollo Android se refiere.

2.2.3 Protocolos empleados para la retransmisión

Para llevar a cabo la retransmisión y recepción de contenido, se tendrán que hacer uso de diversos protocolos que serán descritos en las siguientes subsecciones de esta sección.

2.2.3.1 UPnP

Ilustración 8. Logotipo UPnP. Fuente: Open Connectivity Foundation.

Universal Plug and Play, conocido por su abreviatura UPnP, es un protocolo de red que define una arquitectura para la conectividad de red *peer-to-peer*. Este protocolo permite que los dispositivos compatibles con este protocolo sean descubiertos y puedan detectar de manera transparente la presencia de otros dispositivos en la red. Son *plug-and-play*, ya que pueden conectarse a la red y una vez conectados puede establecer comunicaciones entre los distintos dispositivos sin necesidad de una configuración manual o intervención del usuario [13].

Podría decirse que este protocolo se encarga de automatizar los pasos necesarios para agregar y conectar dispositivos entre los que se encuentran los dispositivos inteligentes, dispositivos inalámbricos y las computadoras personales.

Al tratarse de un protocolo con una arquitectura abierta y distribuida que se basa en estándares reconocidos como la Familia de protocolos de Internet (TCP/IP), HTTP, XML, y SOAP. Por tanto, no requiere de ningún controlador o tecnología adicional para funcionar. Sin embargo, el proceso en el que un dispositivo se une a la red para que el resto de los dispositivos pueda comunicarse con este, pasar por seis fases [13] [14]:

- **Direccionamiento:**
  El nuevo dispositivo debe tener una dirección IP única para formar parte de la red, esto es, cada dispositivo debe implementar un cliente DHCP. Por tanto, este buscará un servidor DHCP (Dynamic Host Configuration Protocol, conocido en español como Protocolo de Configuración Dinámica de Host) cuando se conecte por primera vez a la red y solicitará una dirección IP. En caso de no haber ninguno disponible, el dispositivo se asigna a sí mismo una dirección, conocido este proceso como AutoIP. Si el dispositivo obtiene un nombre de dominio durante la transacción DHCP (por ejemplo, a través de un servidor DNS o mediante el reenvío de DNS), el dispositivo utilizará ese nombre en las operaciones de red. Aquellos dispositivos IoT (Internet of Things, conocido en español como Internet de las cosas) como cafeteras, termostatos o bombillas conectados a Internet no siempre requiere una dirección IP, ya que se comunican a través de tecnologías como Bluetooth o RFID (Identificación por Radio Frecuencia).

- **Descubrimiento:**
  El dispositivo utiliza un protocolo llamado Simple Service Discovery Protocol (SSDP) para presentar sus detalles a los puntos de control de red. Además, este protocolo le permite buscar los dispositivos que le interese controlar. El intercambio fundamental en ambos casos es un mensaje de descubrimiento que contiene datos básicos del dispositivo o uno de sus servicios, como, por ejemplo: su tipo, su identificador y un enlace una URL en la que obtener información más detallada.
  Dependiendo de la configuración de UPnP, el punto de control de red puede buscar activamente dispositivos de interés o escuchar pasivamente mensajes SSDP
- **Descripción:**
  En esta fase, aún se dispone de poca información, por lo que el punto de control debe obtener la descripción del dispositivo desde la URL proporcionada por el dispositivo en el mensaje de descubrimiento para conocer mejor sus capacidades y poder interactuar con él. Esta información detallada, codificada en XML incluye: información sobre el proveedor, nombre del modelo, número de serie, una lista de servicios y de forma opcional, URLs a sitios web específicos del fabricante.
  Tras recibir esta información, la red crea un documento de descripción del dispositivo que enumera las direcciones URL para el control, los eventos y la descripción del servicio. Cada descripción del servicio también incluye los comandos a los que el servicio puede responder y los parámetros para cada acción.
- **Control:**
  Antes de que el punto de control pueda comenzar a interactuar con el dispositivo descubierto, la red envía mensajes para invocar acciones en los servicios. Estos mensajes de control también están en formato XML y utilizan el Protocolo simple de acceso a objetos (SOAP). El servicio responderá con un mensaje de control con los resultados de la acción de forma similar a una llamada a una función. Los efectos de la acción, en caso de existir, se modelarán mediante cambios en las variables que describen el estado del servicio.
- **Notificación de eventos:**
  UPnP tiene un protocolo llamado General Event Notification Architecture (GENA) que permite que un punto de control se registre como un servicio para recibir notificaciones de cambios de estado del dispositivo. Cuando un punto de control se suscribe por primera vez se le envía un mensaje especial de eventos; que contiene el nombre y los valores de todas las variables que generan eventos y permite al suscriptor conocer el estado actual del servicio.
  El servicio envía una notificación de evento a todos los puntos de control registrados cada vez que cambia una variable de estado. Estos mensajes de eventos simples están en formato XML y contienen solo las variables de estado y su valor actual.
- **Presentación:**
  Si el nuevo dispositivo proporciona una dirección URL de presentación en la fase de descripción, el punto de control puede obtener una página dicha dirección y cargarla en un navegador donde, en algunos casos, el usuario puede controlar el dispositivo o ver su estado.
  El grado en que un usuario puede interactuar con un dispositivo depende de interfaz de presentación y el dispositivo.

Las fases descritas anteriormente, podrían resumirse en los siguientes puntos:

- Configuración de la dirección IP del dispositivo.
- Transmitir el nombre y las capacidades del dispositivo al resto de la red.
- Informar a la nueva pieza de hardware sobre las capacidades de otros dispositivos conectados.
- Permitir que los dispositivos de red se comuniquen y funcionen en conjunto.
- Deja una red automáticamente, sin problemas y sin dejar ninguna información de estado. Los servidores DHCP y DNS son opcionales y solo se utilizan si están disponibles en la red.

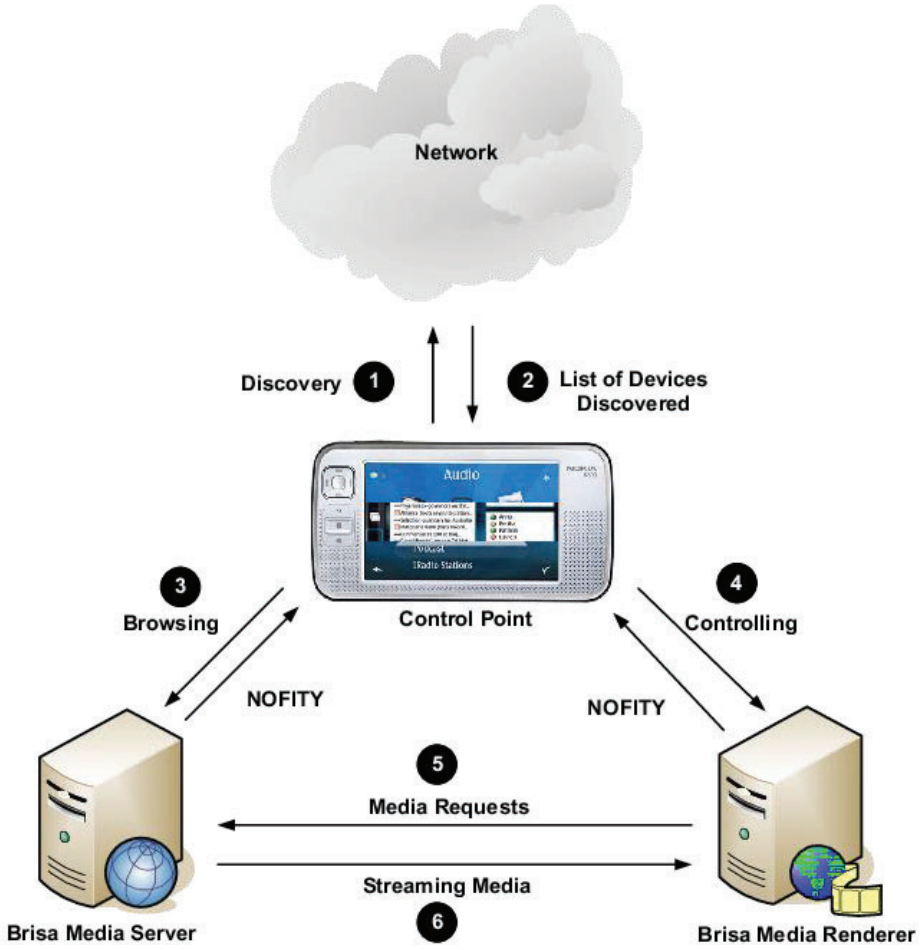

Ilustración 9. Diagrama funcionamiento UPnP. Fuente: ResearchGate.

Algunos de los casos de uso más comunes para UPnP incluyen: conexión inalámbrica con una impresora, conexión de una consola a un servidor de videojuegos, transmisión de contenido multimedia desde un smartphone o un ordenador a una televisión inteligente, vincular unos altavoces inalámbricos al teléfono, conectarse a un sistema de videovigilancia del hogar, conexión y control remoto de un sistema IoT desde el teléfono (iluminación y cerraduras inteligentes, termostatos controlados por Wi-Fi, entre otros).

Beneficios y características destacadas [14][15]:

- Independencia de formato y dispositivos.
  La tecnología UPnP puede ejecutarse en muchos medios que soportan IP incluyendo Ethernet, FireWire, IR (IrDA) y RF (Bluetooth, Wi-Fi). No son necesarios controladores especiales, en su lugar se utilizan protocolos de red comunes.
- Independencia de la plataforma.
  Los proveedores pueden usar cualquier sistema operativo y cualquier lenguaje de programación para crear productos UPnP.
  No especifica o restringe el diseño de un API de las aplicaciones que se ejecutan en los puntos de control; los desarrolladores de sistemas operativos pueden crear APIs que se ajusten a las necesidades de sus usuarios.
- Control de la Interfaz de usuario (IU).
  La arquitectura UPnP permite que los dispositivos presenten una interfaz de usuario mediante un navegador web, así como su interacción.
- Extensibilidad.
  Los productos UPnP pueden tener servicios específicos para dicho dispositivo en capas superiores a la arquitectura básica UPnP. Además de combinar servicios definidos por el Foro UPnP de múltiples formas, los fabricantes pueden definir tipos de servicios y dispositivos propios, y pueden extender dispositivos y servicios ya definidos en el estándar para soportar acciones, variables de estado, elementos de estructuras de datos y variables definidas por ellos mismos.

Si bien es conveniente, el protocolo UPnP no está libre de riesgos. No autentica los dispositivos, asumiendo por defecto que todos los dispositivos son confiables. Permite que cualquier persona experta mediante un programa encargado de hacerse pasar por dispositivo LAN, pueden enviar una solicitud UPnP al router y una vez que se conectan a la red, ejecutan cualquier malware en el sistema llegando incluso a poder robar datos confidenciales.

Este riesgo de seguridad es la razón por la que UPnP normalmente se habilita en redes domésticas y no en un ámbito empresarial.

Otra opción más segura es utilizar la solución no estándar llamada UPnP-UP (Universal Plug and Play - User Profile). Esta versión tiene una extensión para la autenticación de usuarios de la que carece el protocolo original, pero no todos los dispositivos son compatibles con esta opción.

2.2.3.2 DLNA y Miracast

Ilustración 10. Logotipo DLNA. Fuente: Wikipedia.

Digital Living Network Alliance, comúnmente conocido como DLNA, es un estándar de comunicación entre dispositivos conectados a la misma red. Utiliza UPnP, protocolo definido en la subsección anterior, por lo que no es necesaria una configuración manual para hacer uso de esta tecnología.

Su principal función es comunicar diversos dispositivos de forma sencilla y que pueda haber dentro de una misma red, por lo que puede funcionar tanto en redes Wifi como Ethernet.

Las clases de dispositivos con certificación DLNA [16], se dividen en tres: los dispositivos de red doméstica, los dispositivos portátiles móviles y los dispositivos de infraestructura doméstica.

Intel, junto con Sony y Microsoft en junio 2003 [17], establecieron el DLNA. Fue en 2006 cuando se agregaron las categorías de productos las impresoras y dispositivos móviles.

Este suele estar presentes en dispositivos tales como televisores, discos duros de red, consolas de sobremesa, teléfono móvil, cámaras, impresoras, entre otros.

En este estándar, los dispositivos pueden adquirir uno o varios de los siguientes roles:

- **Digital Media Servers:** son los que disponen de todo el contenido.
- **Digital Media Controllers:** son los que controlan la reproducción en la red DLNA. Definen el origen y el destino de los flujos de datos (vídeo, audio o fotos) así como permiten controlar la forma de reproducción.
- **Digital Media Renderers:** son aquellos capaces de negociar y recibir un flujo de datos compatible e interpretarlo para visualizarlo.

Los dispositivos que comparten contenidos por DLNA trabajan a nivel TCP, con una interfaz HTTP+SOAP basada como se menciona anteriormente, en UPnP. Un dispositivo no necesita estar conectado de manera inalámbrica a otro para que funcione.

Ilustración 11. Logotipo Miracast. Fuente: Wikipedia.

El incluir este protocolo en la aplicación es relevante, a diferencia de Miracast que utiliza Wifi para realizar una conexión directa entre dos dispositivos, DLNA puede involucrar múltiples dispositivos que se encuentren dentro de una red doméstica. Un dispositivo Chromecast, usa el protocolo Google Cast, un protocolo que mezcla tanto DLNA como Miracast.

La ventaja de un dispositivo con DLNA frente a un Chromecast es que no es necesario tener en ciertas ocasiones tener instalada la aplicación necesaria para reproducir el contenido como sí hace el Chromecast, pues simplemente reproduce el contenido que recibe y no manda la orden de que reproduzca el contenido elegido en la aplicación que corresponda al reproducirse desde Internet.

Por ello, este protocolo tiene cierta ventaja para el uso que se desea dar en el proyecto a desarrollar, aunque se hará uso de Miracast para usar un dispositivo con esta tecnología como reproductor multimedia.

2.2.3.3 WebRTC

Ilustración 12. Logotipo WebRTC. Fuente: Wikipedia.

WebRTC (Web Real-Time Communication) es una tecnología basada en un proyecto de código abierto, esto es, de software libre, que permite a las aplicaciones y sitios web comunicación en tiempo real (RTC). Permite retransmitir tanto audio como vídeo, así como datos genéricos que incluyen los archivos multimedia, dando la posibilidad de hacer videoconferencias de igual-a-igual, conocido comúnmente como comunicaciones peer-to-peer. Una de sus bondades es que no requiere de instalación de plug-ins o software adicional para que la comunicación sea efectiva. Además, destaca por su calidad de video de alta definición y su baja latencia, siendo así la comunicación en tiempo real.

WebRTC incluye interfaces de programación de aplicaciones (API) y protocolos comunes interrelacionados necesarios para que pueda realizarse la comunicación. [18]

Fue en enero de 2011, cuando Google lanzó tras comprar una compañía de software de VoIP y videoconferencia conocida como Global IP Solutions, cuando lanzó este proyecto de código abierto. Desde entonces WebRTC, a parte del apoyo de Google y la integración de esta tecnología en Google Hangouts, cuanta además con la participación de empresas como Apple, Microsoft, Mozilla y Opera, entre otras, que dan soporte en sus navegadores a esta tecnología. Esto ha dado lugar a que finalmente se está estandarizando los protocolos relevantes en el Internet Engineering Task force (IETF) y la API del navegador en el World Wide Web Consortium (W3C).

Para iniciar una interacción peer-to-peer, son necesarios tres componentes principales, cuyos roles se detallan a continuación[19]:

- Media Stream o transmisión de medios, cuya API se encarga de acceder a la cámara y el micrófono del dispositivo. Este componente se encarga de la captura y reproducción de los medios.
- Peer Connection o conexión entre pares, responsable de establecer la conexión peer-to-peer y así crear una comunicación directa sin la intervención de ningún intermediario. De esta manera, se puede adquirir o consumir los medios además de producirlos, y establecer así una llamada de audio y vídeo.
- Data Channels o los canales de datos, son los que ayudan y permiten crear una transferencia bidireccional entre dos navegadores y compartir datos a través de una conexión peer-to-peer- Esto funciona con el Stream Control Transmission Protocol (SCTP). Se crea estos canales de datos para reducir la congestión en las redes y garantizar la transmisión.

Para establecer una conexión a través de WebRTC, es necesario que se produzcan una serie de pasos.

Primero, es necesario hacer uso de un servidor de señalización. Este servidor es el intermediario encargado de dos dispositivos se encuentren y establezcan una conexión. Para ello, cuando se manda una señalización al servidor, se incluye una descripción de la sesión, en formato SDP.
Por otro lado, cada dispositivo cuenta con un subsistema ICE. Este se encarga de asegurar la mejor conexión posible entre dos pares, incluso si es difícil de conectar. Por ello, cuando un subsistema ICE le indica al servidor de señalización que envíe los datos que incluyen una descripción a otro par, el otro sabe cómo recibir y entregarla a su propio subsistema ICE. En estos datos, se trata de exponer el mínimo de información privada, entre los que se encuentran la dirección IP de los agentes que intervienen, las pistas de audio y vídeo con las que cuenta un agente y las que transfiere otro agente y los canales de datos que establecen el tipo de los medios que se intercambian.
Por tanto, se puede decir que el contenido no importa en absoluto al servidor de señalización, pues esta señalización lo que hace es ayudar al servidor a enviar y recibir datos, haciendo directa la comunicación por medio de servidores Standard Traversal Utilities for NAT (STUN) y Traversal Using Realys arround NAT(TURN). [20]. Con esto servidores, conseguimos que la comunicación bidireccional entre dos pares sea posible, aunque estos cuenten con diferentes protocoles de red o direcciones de transporte, habiendo así una conexión entre ellos.

Una vez se produce la señalización y la conexión, hace falta que haya un aseguramiento. Con el aseguramiento, se garantiza que la comunicación compartida entre dos agentes se encuentre encriptada y sea confidencial con cualquier tercero. Para ello, se usa dos protocolos que son: Datagram Transport Layer Security(DTLS) y Secure Real-Time Transport Protocol (SRTP).

Por último, se establece la comunicación para transferir datos, audio y vídeo a través de la web. Para ello, WebRTC hace uso de los siguientes protocolos centrales de comunicación:

- Real-Time Transport Protocol (RTC): es un protocolo de transporte en tiempo real y se emplea para la entrega de video a tiempo real. No garantiza la baja latencia o la fiabilidad, pero proporciona las herramientas para implementarla.
- Real-Time Transport Control Protocol(RTCP): es un protocolo de control en tiempo real. Se encarga de monitorizar la calidad de las llamadas a partir de metadatos recopilados. También se usa para la latencia, entre otros problemas de VoIP.

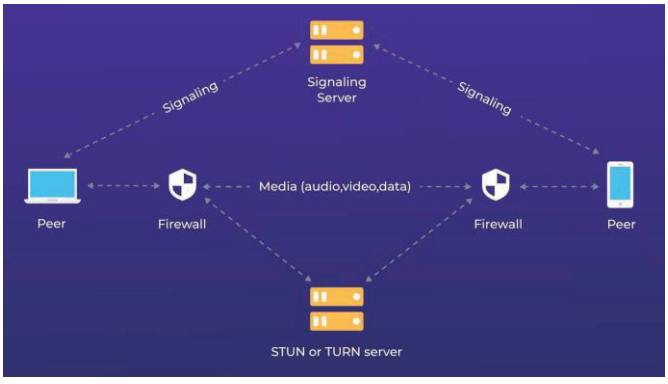

Ilustración 13. Diagrama del funcionamiento del protocolo WebRTC. Fuente: videosdk.live

El uso de esta tecnología ofrece ciertas ventajas: calidad por encima de latencia, autenticidad de los mensajes, ancho de banda reducido, seguridad end-to-end (E2E), coordinación con valores SDP. Es gratis y flexible, ofrece calidad y rapidez, así como combinación con VoIP. Tiene compatibilidad absoluta, pues es compatible con la mayoría de los navegadores y se puede utilizar en cualquier tipo de red y sistema operativo.

Por otro lado, también se han encontrado algunas desventajas: aunque se trata de exponer el mínimo de datos y se trata de garantizar la seguridad al máximo, en enero de 2015, TorrentFreak informó que los navegadores compatibles con WebRTC, podrían comprometer la seguridad de los túneles VPN, al exponer la verdadera dirección IP. Sin embargo, se ha ido corrigiendo esta falla o aportando soluciones como el uso de extensiones que eviten que ocurra esa vulnerabilidad.

2.2.3.4 Wireless Projection

En la actualidad, la gran mayoría de dispositivos Android cuentan con la tecnología Miracast.

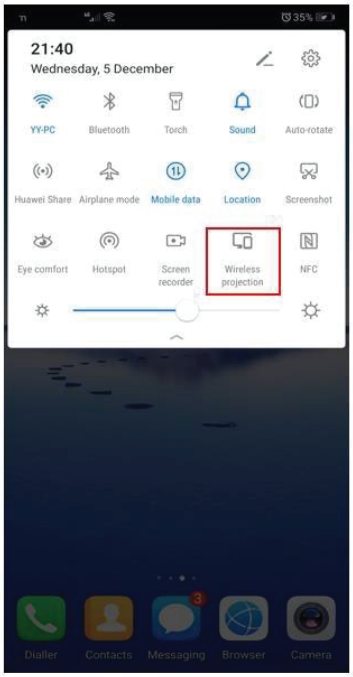

Ilustración 14. Captura de pantalla del acceso directo del panel de notificaciones a Wireless Projection. Fuente: Reddit.

Esto permite que puedas duplicar la pantalla del dispositivo con otros dispositivos compatibles con esta tecnología, como puede ser un ordenador con el sistema operativo de Windows [21], o un dispositivo Chromecast cuyo uso ha crecido exponencialmente a lo largo de los años debido a su facilidad de integración en cualquier dispositivo que cuente con un puerto con HDMI.

Esto permitirá que, aunque no se integre Google Cast para enviar el contenido a dispositivos compatibles con esta tecnología, podamos duplicar la pantalla del dispositivo, y del mismo modo, reproducir el contenido deseado en los dispositivos que poseen la tecnología Miracast.

2.2.4 Formatos de codificación

Las próximas subsecciones reflejan tecnologías y formatos para la codificación del contenido.

2.2.4.1 MediaProjection y MediaCodec

Son clases de Android. La primera de ellas [22] otorga la capacidad a las aplicaciones de capturar contenido de pantalla y/o grabar audio del sistema, mientras que la segunda [23] se utiliza para acceder a componentes de codificación y decodificación.

2.2.4.2 Códec VP8 y Códec H.264 High Profile

El códec VP8 [24] es un formato de compresión que se emplea con la tecnología WebRTC para la compresión de vídeo. Este, fue diseñado por On2 Technologies y es de formato abierto, por lo que cualquiera puede usarlo.

Por otro lado, el códec H.264 High Profile [25] es un estándar de alta compresión desarrollado por el ITU-T Video Coding Experts Group (VCEG) y el ISO/IEC Moving Picture Experts Group (MPEG). Este puede ser empleado en aplicaciones de transmisión y almacenamiento, y es adecuado cuando se trata de emisión de vídeo en tiempo real.

2.2.5 Dependencias empleadas

A continuación, en las siguientes subsecciones, se recogerá el uso de las librerías más relevantes que se han empleado en el desarrollo de la aplicación y se explicará brevemente el porqué de su uso.

2.2.5.1 Cling

Ilustración 15. Imagen del repositorio de cling. Fuente: GitHub.

Cling es una librería para Java y Android que implementa la arquitectura de dispositivos UPnP. Esta será usada para descubrir dispositivos UPnP y utilizar así sus servicios. Su licencia es LGPL, por lo que no hay restricciones para usarla sin modificarla. El código fuente y el manual de usuario se encuentra en la web de 4thline [26].

2.2.5.2 WebRTC para Android

Ilustración 16. Imagen logotipo WebRTC para Android. Fuente: Ekobit.

WebRTC para Android es una librería que adapta el proyecto WebRTC, facilitando así su implementación en el desarrollo de aplicaciones que integren esta tecnología, de este sistema operativo. Además, al tratarse un proyecto de código libre, cuenta con una gran comunidad de desarrolladores que se dedican a la mejora y resolución de problemas que se puedan encontrar al usarse para desarrollar aplicaciones que integren este sistema [27].

2.2.5.3 NanoHttpd

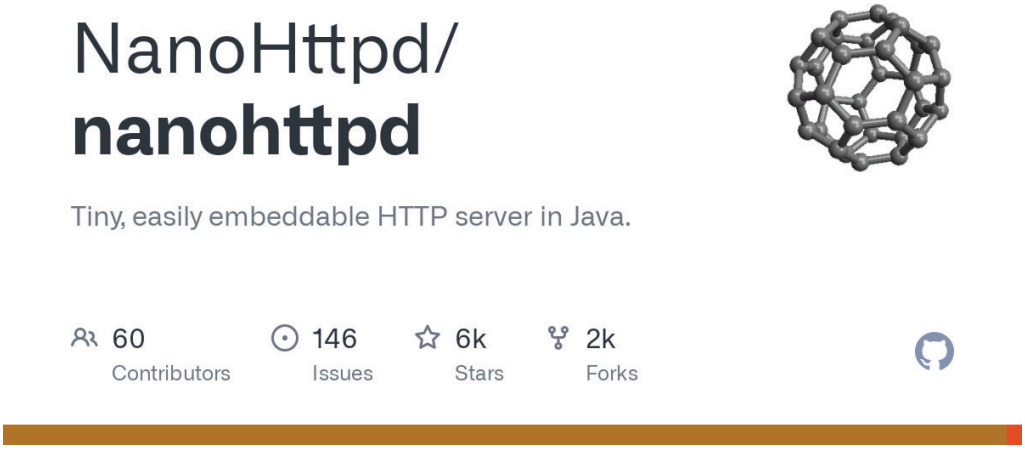

Ilustración 17. Imagen del repositorio de NanoHttpd. Fuente: GitHub.

NanoHttpd es un servidor HTTP gratuito y ligero que se puede integrar en la aplicación de Android al estar implementado en Java, para crear un servido de interfaz de API. Admite solicitudes GET, POST, PUT, HEAD y DELETE. Se hará uso de esta dependencia para crear la parte de servidor de la aplicación. Toda la información se puede encontrar en su página web [28].

2.2.5.4 Firebase Crashlytics y Firebase Analytics

Firebase es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que ofrece un conjunto de herramientas. De estas, se ha decidido integrar al proyecto tanto Crashlytics como Analytics.

Firebase Crashlytics [29] es una herramienta útil y necesaria para el proyecto ya que informa de las fallas de la aplicación en tiempo real, permitiendo hacer un seguimiento de los problemas de estabilidad, priorizarlos y corregirlos.

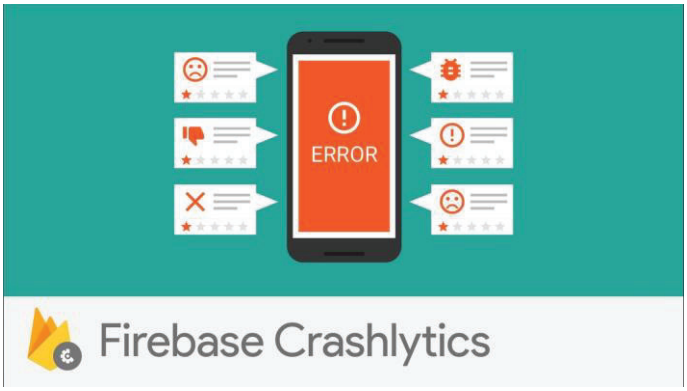

Ilustración 18. Logotipo Firebase Crashlytics. Fuente: Firebase.

Por otro lado, Firebase Analytics [30] proporciona estadísticas sobre el uso de las apps y la participación de los usuarios, útil para la futura comercialización de la app, así como para las optimizaciones del rendimiento.

Ilustración 19. Logotipo Firebase Analytics. Fuente: Firebase

2.2.5.5 ZXing

Ilustración 20. Logotipo de ZXing. Fuente: GitHub.

ZXing [31] es un proyecto de código abierto implementado en el lenguaje de Java, que permite la generación de códigos QR a partir de textos, y que se ha empleado en el desarrollo de la aplicación, para compartir la dirección de retransmisión de contenido en vivo, de una forma más rápida, práctica y más fácil de compartir.

2.2.6 Tecnologías descartadas

Algunas de las tecnologías que se han ido descartando, como se ha explicado en algunos de los puntos, son las siguientes:

- **El uso del lenguaje Java en vez de Kotlin.**
  Aunque el empleo de Kotlin empieza a ser más frecuente en el desarrollo de aplicaciones Android, he preferido optar por Java, un lenguaje que conozco a fondo y que cuenta con gran soporte de la comunidad de desarrollo.

- **El desarrollo de una aplicación nativa frente a otras soluciones.**
  La elección del desarrollo de una aplicación frente a otras soluciones como son las aplicaciones de escritorio o aplicaciones web se debe al uso destinado. Hoy en día, es más frecuente consumir y enviar contenido multimedia a través de dispositivos móviles que en un ordenador. También a la hora de generar ese contenido en vivo, es más fácil generarlo a través del móvil con la cámara integrada que si se requiere un ordenador con una cámara conectada y la limitación de estar cerca de este. Por otro lado, una aplicación web sería ideal para que dicha solución coexistiera en todas las plataformas, pero estaría limitado en la forma que se genera, codifica y decodifica el contenido en vivo.

- **Uso de WebRTC para la retransmisión de contenido en vivo en vez de otros protocolos.**
  Existen otros protocolos que se podrían haber empleado para las emisiones de contenido en directo.
  Por un lado, tenemos RTMP (Real Time Messaging Protocol), un protocolo desarrollado por Macromedia, una empresa que pertenece a Adobe, que emplea la mensajería en tiempo real de audio, vídeo y datos entre servidor y un reproductor Flash. Es un protocolo que se suele utilizar para transmisiones en vivo, sin embargo, Flash se encuentra en fase de desaparición debido a su latencia notable si hay pérdida de paquetes en la secuencia que retransmite [32].
  Por otro lado, se encuentra el protocolo HLS (HTTP Live Streaming) que destaca por soportar una transmisión con tasa de bits adaptativa, y, por tanto, la calidad se adaptar al ancho de banda del cliente Player. Soporta, además, el códec H.265 y su codificación es realizada por segmentos (chunks de vídeo), enviando así pequeñas descargas del vídeo cada cierta cantidad de segundos. Otras de las ventajas es su compatibilidad tanto con dispositivos móviles como los reproductores de vídeo HTML5, haciendo que sea el más adecuado para transmitir video en vivo a bases de espectadores masivos. Sin embargo, como el protocolo anterior es la elevada latencia que existe durante la transmisión. [32]
  Por tanto, debido a la latencia de estos protocolos y los pocos recursos que se han encontrado para la transmisión de pantalla y cámara para dispositivos Android, se ha decidido usar WebRTC, el cual destaca por su calidad de video de alta definición y su baja latencia, pudiéndose realizar así la transmisión en tiempo real, al ser un protocolo que está preparado y se emplea principalmente para videollamadas.

- **Uso de soluciones preparadas para la transmisión en vivo como GStreamer:**
  GStreamer es un *framework* multimedia libre multiplataforma escrito en el lenguaje C para la creación de aplicaciones audiovisuales. Entre sus posibilidades, existe la de poder transmitir contenido multimedia y en vivo. Aunque cuenta con un repositorio en el que ofrecen código de ejemplo para la implementación en Android, se ha considerado un framework bastante complejo que requiere muchos recursos y aprendizaje previo para su implementación. Por eso, se ha optado por una solución más sencilla, aunque se podría haber implementado junto con la tecnología WebRTC y no se descarta su uso para posibles líneas de futuro.

2.3 Proyectos similares

En las siguientes secciones detallaré alguno de los proyectos que he ido encontrando y pueden ser útiles para el desarrollo de la aplicación.

2.3.1 DroidDLNA

DroidDLNA [33] se trata de proyecto que implementa un reproductor de contenido multimedia recibido por un servidor UPnP. Hace uso de la librería de Cling y está desarrollado en el lenguaje de Java para Android. Este código se terminó de desarrollar en 2014 y pasó a ser de código abierto en 2015.

Este proyecto podría servir para implementar la parte de cliente, que recibe los contenidos multimedia y los reproduce.

2.3.2 SlickDLNA

SlickDLNA [34] es una aplicación similar a la anterior que sirve como cliente para retransmitir contenido multimedia alojados en servidores UPnP y DLNA. Utiliza la librería de Cling y Jetty para la conexión. Está desarrollado en el lenguaje de Java para Android, y la aplicación se encuentra disponible en el Play Store con una valoración de 4,4/5 estrellas.

Es otra alternativa para implementar la parte del cliente del proyecto e implementar cómo recibir y reproducir los contenidos multimedia.

2.3.3 ScreenCastSample

ScreenCastSample [35] es un repositorio de una aplicación encarga de retransmitir en vivo el contenido capturado en la pantalla del dispositivo, haciendo uso del MediaProjection y MediaCodec. Hay un enlace en japonés en el que explica paso a paso el proceso que ha seguido para implementar esta funcionalidad y que sin duda es útil para la funcionalidad de retransmitir contenido en vivo en este proyecto.

2.3.4 DroidUPnP

DroidUPnP [36] es la aplicación más completa que he encontrado para el desarrollo de este proyecto. Se trata de una aplicación capaz de descubrir los dispositivos dentro de la misma red haciendo uso del protocolo UPnP y reproducir así el contenido recibido, como el caso de los dos primeros proyectos mencionados. A diferencia de estos, además es capaz de ser proveedor de contenido, permitiendo ser detectado como servidor UPnP y de mandar únicamente contenido multimedia. Por tanto, a este proyecto le faltaría el poder retransmitir además contenido en vivo, pero es un buen proyecto base para el desarrollo de mi aplicación y con la que otras aplicaciones han basado su aplicación como es el caso de Plain UPnP [4].

2.3.5 WebScreen

WebScreen [36] es una aplicación enfocada en la retransmisión de contenido en vivo mediante la tecnología WebRTC, además de permitir el control remoto de pantalla. Esta, se encuentra publicada en el Play Store, sin embargo, la versión publicada no ofrece la posibilidad de poder retransmitir el video capturado de las distintas cámaras del dispositivo ni el audio del micrófono. No obstante, ha sido un proyecto fundamental para el resultado final que se ha implementado en este trabajo, pues tan solo ha sido necesario implementar la posibilidad de capturar el vídeo que recogen las cámaras y el canal de audio para cubrir así la funcionalidad requerida de la emisión de contenido en vivo.

3 Análisis de requisitos

En los capítulos anteriores, se ha hablado del propósito y la necesidad del desarrollo de esta aplicación, el alcance que podría tener y los posibles casos de usos que se podrían dar, se ha hecho un estudio del estado del arte donde se recogían las posibles alternativas que no cumplen con el propósito final de este proyecto y las tecnologías que emplear para realizar su implementación.

Por tanto, en este capítulo se ha definido aquellos requisitos que la aplicación debe cumplir, tanto para la funcionalidad de poder reproducir el contenido multimedia y en vivo de los dispositivos descubiertos en la red local, como la funcionalidad del servidor que ofrece estos contenidos, antes de pasar al desarrollo de la aplicación.

3.1.1 Requisitos para la reproducción de contenido

Requisito 1 (R1): Descubrir dispositivos en la red local.

Descubrir cualquier dispositivo compatible con el protocolo UPnP o Miracast, de la red local.

Requisito 2 (R2): Reproducir el contenido multimedia y en vivo de los dispositivos descubiertos en la red local desde la aplicación.

Reproducir desde el dispositivo donde se encuentre instalada la aplicación cualquier contenido multimedia y en vivo compartido en la red local: imagen, audio, video o retransmisión en vivo.

Requisito 3 (R3): Reproducir un contenido multimedia a través de un dispositivo descubierto en la red local.

Reproducir desde un dispositivo compatible con la tecnología UPnP o Miracast de la red local, cualquier contenido multimedia compartido en la misma red local: imagen, audio o video.

Requisito 4 (R4): Reproducir una retransmisión en vivo a través de un navegador o dispositivo compatible con la tecnología Miracast.

Reproducir desde un navegador o a través de un dispositivo compatible con la tecnología Miracast una retransmisión en vivo compartida en la misma red local.

3.1.2 Requisitos para el servidor de contenido

Requisito 5 (R5): Iniciar servidor de contenido multimedia y en vivo.

Iniciar un servidor de contenido multimedia y en vivo, que se publique como como dispositivo UPnP, permitiendo así ser descubierto por otros dispositivos de la misma red local y compartir el contenido con estos.

Requisito 6 (R6): Apagar el servidor de contenido multimedia y en vivo.

Apagar servidor para la retransmisión de contenido multimedia y en vivo, de manera que este dispositivo se deja de ser visible como dispositivo UPnP en la misma red local.

Requisito 7 (R7): Seleccionar contenido multimedia que se mostrará en la red local.

Seleccionar imágenes, vídeos o audios del dispositivo que se deseen compartir con otros dispositivos que se encuentren conectados a la misma red local.

Requisito 8 (R8): Borrar contenido multimedia que se esté mostrando en la red local.

Borrar imágenes, vídeos o audios que no se quieran compartir con otros dispositivos de la misma red local.

Requisito 9 (R9): Modificar el nombre del servidor.

Cambiar el nombre como dispositivo UPnP del servidor de contenido, con el que se muestra a otros dispositivos en la misma red local.

Requisito 10 (R10): Modificar puerto para el servidor de contenido multimedia.

Definir el puerto de enlace que usará el servidor de contenido en la misma red local, cuyo valor comprenda entre 1025 y 65535.

Requisito 11 (R11): Iniciar una retransmisión en directo.

Iniciar una retransmisión en vivo que se compartirá con aquellos dispositivos conectados a la misma red local.

Requisito 12 (R12): Detener una retransmisión en directo.

Detener una retransmisión en vivo que se está compartiendo con aquellos dispositivos conectados a la misma red local.

Requisito 13 (R13): Modificar puerto para el servidor de retransmisión de contenido en vivo.

Definir el puerto de enlace que usará el servidor de retransmisión de contenido en vivo en la red local, cuyo valor comprenda entre 1025 y 65535.

Requisito 14 (R14): Elegir qué medio de video retransmitir en vivo.

Durante la retransmisión en vivo, poder elegir entre compartir: cámara frontal o trasera del dispositivo, pantalla del dispositivo, u ocultar el vídeo.

Requisito 15 (R15): Elegir qué medio de audio retransmitir en vivo.

Durante la retransmisión en vivo, poder elegir entre compartir: únicamente el micrófono del dispositivo, micrófono más audio interno del dispositivo, o silenciar el audio.

Requisito 16 (R16): Habilitar opción de control remoto del dispositivo durante una retransmisión en vivo de pantalla del dispositivo.

Durante la retransmisión en vivo de pantalla, se podrá activar o desactivar la posibilidad de que otros dispositivos conectados a la retransmisión puedan controlar remotamente el dispositivo.

4 Desarrollo de la aplicación

En este capítulo, tras la definición de requisitos, se detallará exhaustivamente, la implementación de la aplicación. Por un lado, cómo se ha estructurado el proyecto, también se explicará cómo se ha implementado las distintas funcionalidades, cumpliendo así con los requisitos establecidos. Del mismo modo, también se explicará el diseño de la aplicación y las funciones de las que se dispone en cada pantalla.

4.1 Diseño

4.1.1 Patrón de arquitectura software

Para el desarrollo software, se ha seguido el patrón de arquitectura software conocido como MVC (Modelo-Vista-Controlador). Este patrón, es comúnmente utilizado para separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Por tanto, se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo en este caso de la aplicación, para su posterior mantenimiento.

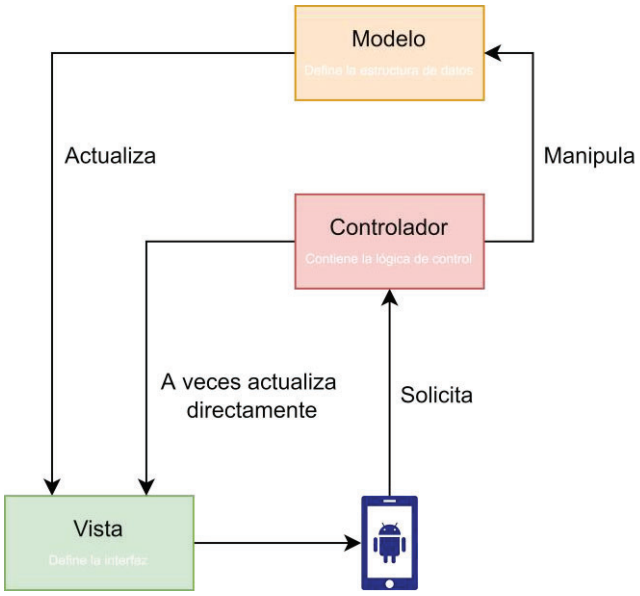

Ilustración 21. Diagrama de patrón de arquitectura software.

4.1.2 Casos de uso

Antes de definir los casos de uso de la aplicación, se hará una breve descripción de los posibles actores que pueden intervenir:

- **Usuario:** este actor puede adquirir dos roles, pues, por una parte, está el usuario que comparte y ofrece el contenido a través de la aplicación, y, por otro lado, aquél que reproduce y consume el contenido con la aplicación.
- **Aplicación:** al igual que el actor Usuario, puede adquirir dos roles, en base a la acción que se desea realizar. Si se desea compartir el contenido, será Servidor, y si, por el contrario, se desea reproducir el contenido, adquirirá el rol de Reproductor.

- **Dispositivo externo:** este actor engloba a los dispositivos compatibles con la tecnología UPnP o Miracast, y o bien ofrece contenido que reproducir, o bien actúa como reproductor de dicho contenido.

A continuación, se listará aquellos posibles casos de uso que se pueden encontrar en la aplicación, asociando cada caso con los requisitos que se han definido, el propósito, los actores que intervienen, si debe existir alguna precondición, y la descripción de este caso de uso.

CU1: Iniciar el servidor de contenido.

- **Requisitos:** R5
- **Propósito:** iniciar el servidor de contenido multimedia y en vivo, para que se muestre visible al resto de dispositivos conectados a la misma red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** -
- **Descripción:**
    - Desde la vista Servidor:
        - Usuario pulsa vista Servidor.
        - Usuario pulsa botón para activar servidor.
        - Aplicación inicia servidor.

    - Desde la vista Ajustes:
        - Usuario pulsa menú.
        - Usuario pulsa opción de Ajustes.
        - Usuario pulsa botón para activar servidor.
        - Aplicación inicia servidor.

CU2: Cambiar nombre del servidor de contenido.

- **Requisitos:** R9
- **Propósito:** cambiar el nombre del servidor con el que su publicará en la lista de dispositivos disponibles UPnP de la misma red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** -
- **Descripción:**
    - Desde la vista Servidor:
        - Usuario pulsa vista Servidor.
        - Usuario pulsa nombre del servidor.
        - Usuario escribe nuevo nombre del servidor.
        - Usuario le da a guardar.
        - Aplicación cambia nombre de servidor.
    - Desde la vista Ajustes:
        - Usuario pulsa menú.
        - Usuario pulsa opción de Ajustes.
        - Usuario pulsa opción Ajustes del servidor.
        - Usuario pulsa Cambiar nombre del servidor.
        - Usuario escribe nuevo nombre del servidor.
        - Usuario le da a guardar.
        - Aplicación cambia nombre de servidor.

CU3: Cambiar puerto del servidor de contenido.

- **Requisitos:** R10
- **Propósito:** cambiar el puerto del servidor de contenido de la red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** -
- **Descripción:**
    - Desde la vista Ajustes:
        - Usuario pulsa menú.
        - Usuario pulsa opción de Ajustes.
        - Usuario pulsa opción Ajustes del servidor.
        - Usuario pulsa Cambiar puerto del servidor de contenido.
        - Usuario escribe nuevo puerto.
        - Aplicación muestra advertencia de que se debe reiniciar la aplicación para aplicar cambio.
        - Aplicación se reinicia automáticamente para aplicar cambio.

CU4: Cambiar puerto del servidor de retransmisión de contenido en vivo.

- **Requisitos:** R13
- **Propósito:** cambiar el puerto del servidor de retransmisión de contenido en vivo de la red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** -
- **Descripción:**
    - Desde la vista Ajustes:
        - Usuario pulsa menú.
        - Usuario pulsa opción de Ajustes.
        - Usuario pulsa opción Ajustes del servidor.
        - Usuario pulsa Cambiar puerto del servidor de retransmisión de contenido en vivo.
        - Usuario escribe nuevo puerto.
        - Aplicación aplica el cambio.

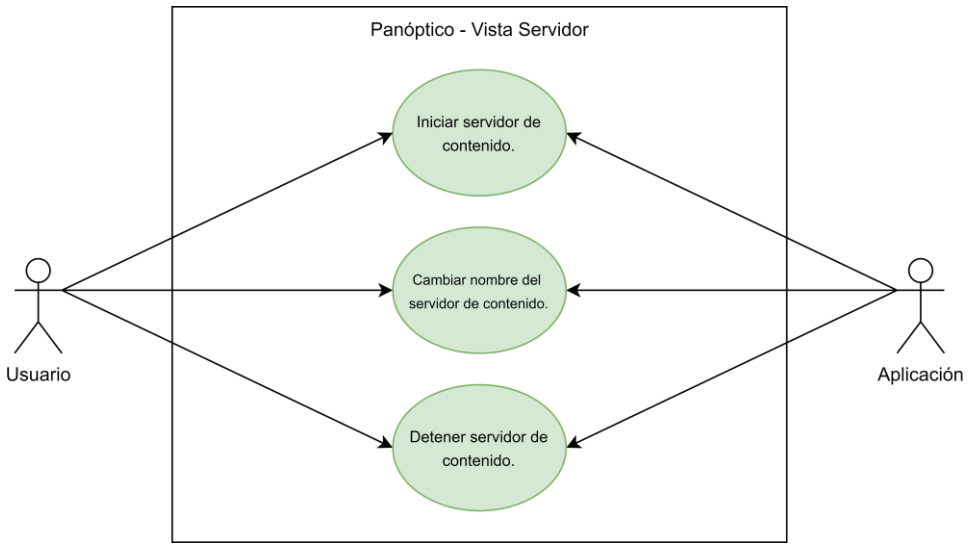

Ilustración 22. Diagrama para representar los siguientes casos de uso desde Vista Servidor: CU1 y CU2.

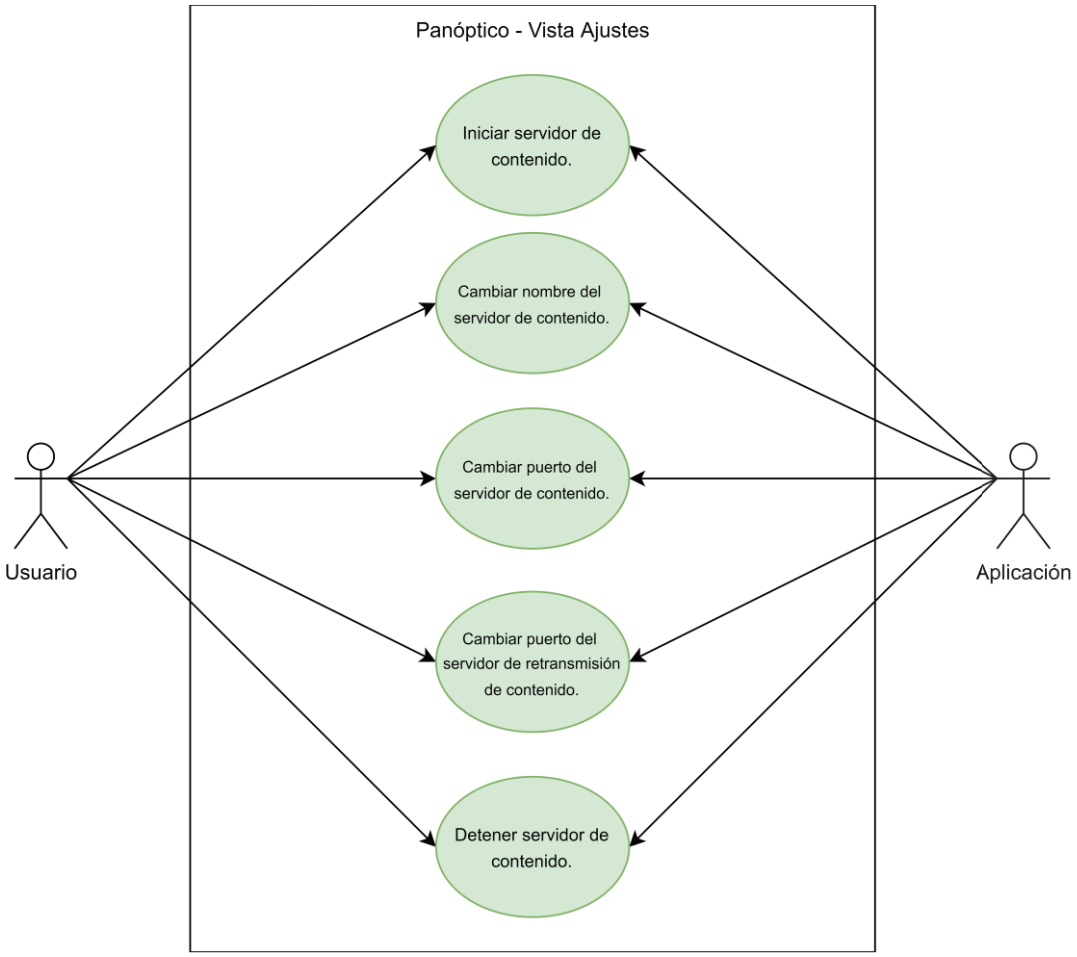

Ilustración 23. Diagrama para representar los siguientes casos de uso desde Vista Ajustes: CU1, CU2, CU3 y CU4.

CU5: Añadir contenido a compartir en el servidor.

- **Requisitos:** R5 y R7.
- **Propósito:** añadir archivos de imagen, vídeo o audio, para así compartirlos con los dispositivos que se encuentren en la misma red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** Servidor de contenido iniciado.
- **Descripción:**
    - Desde la vista Servidor:
        - Usuario pulsa vista Servidor.
        - Usuario pulsa botón flotante Añadir.
        - Usuario pulsa botón en base al tipo de archivos que quiera añadir: fotos, vídeos o audio.
        - Usuario selecciona los archivos que quiere añadir de ese tipo.
        - Usuario pulsa añadir.
        - Aplicación actualiza cambios en el servidor y a otros dispositivos que visualicen el contenido.

CU6: Borrar contenido a compartir en el servidor.

- **Requisitos:** R5 y R8.
- **Propósito:** borrar archivos de imagen, vídeo o audio.
- **Actores:** Usuario y Aplicación.
- **Precondición:** Servidor de contenido iniciado.
- **Descripción:**
    - Desde la vista Servidor, dentro de la carpeta de archivos:
        - Usuario pulsa un archivo que se desea eliminar.
        - Aplicación habilita vista de selección.
        - Opcionalmente el usuario selecciona más archivos.
        - Opcionalmente el usuario pulsa el botón de seleccionar o deseleccionar todo.
        - Usuario pulsa botón de borrar.
        - Aplicación borrar los archivos seleccionados del servidor.
    - Desde la vista Servidor, en la vista de carpetas del directorio.
        - Usuario pulsa una carpeta que se desea vaciar.
        - Aplicación habilita vista de selección.
        - Opcionalmente el usuario selecciona más carpetas.
        - Opcionalmente el usuario pulsa botón de seleccionar o deseleccionar todo.
        - Usuario pulsa botón de borrar.
        - Aplicación vacía las carpetas, y por tanto borra los archivos de esas carpetas del servidor.

CU7: Iniciar retransmisión de contenido en vivo.

- **Requisitos:** R5, R11, R14, R15 y R16.
- **Propósito:** iniciar la retransmisión de contenido en vivo para que pueda ser reproducido por dispositivos conectados a la misma red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** Servidor de contenido iniciado.
- **Descripción:**
    - Desde la vista Servidor:
        - Usuario pulsa botón flotante Añadir.
        - Usuario pulsa botón Streaming.
        - Aplicación muestra vista Streaming.
        - Opcionalmente el usuario cambia la opción de vídeo por defecto Cámara a Pantalla.
        - Opcionalmente el usuario cambia la opción de audio por defecto Micrófono a Micrófono+Interno.
        - Usuario pulsa botón flotante Iniciar retransmisión.
        - Opcionalmente el usuario durante la retransmisión puede pulsar distintas opciones en base al vídeo que retransmita, para cambiar entre cámaras del dispositivo, activar o desactivar control remoto, u ocultar vídeo a transmitir.
        - Opcionalmente el usuario durante la retransmisión puede pulsar la opción relacionada con el audio para activar o silenciar el micrófono.
        - Aplicación inicia la retransmisión en vivo y es compartida por el servidor de contenido al resto de dispositivos que se encuentren conectados a la misma red.

CU8: Detener retransmisión en vivo.

- **Requisitos:** R5, R11 y R12.
- **Propósito:** detener la retransmisión de contenido en vivo, dejando de ser compartida a dispositivos conectados a la misma red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** Servidor de contenido iniciado y retransmisión de contenido en vivo iniciada.
- **Descripción:**
    - Desde la vista Streaming:
        - Usuario pulsa botón flotante Detener retransmisión.
        - Aplicación detiene la retransmisión y deja de compartirlo en el servidor de contenido al resto de dispositivos que ese encuentre conectados a la misma red.

CU9: Detener servidor de contenido.

- **Requisitos:** R5 y R6.
- **Propósito:** detener el servidor de contenido multimedia y en vivo, para que deje de mostrarse al resto de dispositivos conectados a la misma red local.
- **Actores:** Usuario y Aplicación.
- **Precondición:** Servidor de contenido iniciado.
- **Descripción:**
    - Desde la vista Servidor:
        - Usuario pulsa botón Desactivar servidor.
        - Aplicación deja de compartir el servidor de contenido al resto de dispositivos que ese encuentre conectados a la misma red.

CU10: Búsqueda de dispositivos en la misma red local:

- **Requisitos:** R1.
- **Propósito:** descubrir los dispositivos UPnP que se encuentran conectados a la misma red.
- Actores: Aplicación o Usuario y Aplicación.
- **Precondición:** Al menos, servidor de contenido iniciado desde otra aplicación o algún dispositivo UPnP que ofrezca contenido.
- **Descripción:**
    - Desde la vista Reproductor, sin intervención del usuario:
        - Aplicación comprueba periódicamente los dispositivos UPnP que se encuentran en la misma red, descubriendo nuevos dispositivos o dejando de mostrar los que ya no están disponibles.
    - Desde la vista Reproductor, interviniendo el usuario:
        - Usuario refresca la lista de dispositivos descubiertos.
        - Aplicación comprueba los dispositivos UPnP que se encuentran en la misma red, descubriendo nuevos dispositivos o dejando de mostrar los que ya no están disponibles.

CU11: Reproducir contenido en el dispositivo:

- **Requisitos:** R1 y R2.
- **Propósito:** visualizar el contenido de los dispositivos UPnP descubiertos al encontrarse conectados a la misma red, y que ofrezcan contenido.
- Actores: Usuario y Aplicación.
- **Precondición:** Al menos, servidor de contenido iniciado desde otra aplicación o algún dispositivo UPnP que ofrezca contenido.
- **Descripción:**
    - Desde la vista Reproductor:
        - Usuario refresca la lista de dispositivos UPnP que ofrecen contenido y se encuentran conectados a la misma red local, o espera a que se actualice dicha lista automáticamente.
        - Usuario selecciona uno de los dispositivos.
        - Usuario navega entre las distintas carpetas del directorio.
        - Usuario selecciona uno de los archivos disponibles que se desea visualizar.
        - Aplicación abre el contenido a través de las aplicaciones por defecto que se encuentren en el dispositivo.

CU12: Reproducir contenido de otros dispositivos, en un dispositivo reproductor.

- **Requisitos:** R1, R2, R3, R4.
- **Propósito:** visualizar el contenido de los dispositivos UPnP descubiertos al encontrarse conectados a la misma red, y que ofrezcan contenido, a través de dispositivos UPnP o Miracast.
- **Actores:** Usuario, Aplicación, Dispositivo externo.
- **Precondición:** Al menos, servidor de contenido iniciado desde otra aplicación o algún dispositivo UPnP que ofrezca contenido. También un dispositivo UPnP o Miracast conectado a la misma red local para lanzar el contenido a dicho dispositivo.
- **Descripción:**
    - Desde la vista Reproductor:
        - Usuario refresca la lista de dispositivos UPnP que ofrecen contenido y se encuentran conectados a la misma red local, o espera a que se actualice dicha lista automáticamente.
        - Opcionalmente, el usuario puede elegir un dispositivo por defecto, pulsando el botón Cast, el que está al lado de menú.
        - Usuario selecciona uno de los dispositivos.
        - Usuario navega entre las distintas carpetas del directorio.
        - Usuario hace pulsación larga sobre el archivo o selecciona la opción que se encuentra a la derecha de este elemento, para mandar el contenido a un dispositivo reproductor.
        - Si no se ha elegido un dispositivo UPnP donde reproducir el contenido previamente, se muestra una ventana y el usuario elige dispositivo al que se enviará el contenido.
        - Opcionalmente, si se desea compartir el contenido a través de un dispositivo compatible con Miracast, Usuario pulsa botón Cast, selecciona opción Miracast y la aplicación les redirige a los ajustes de Wireless Projection, para duplicar la pantalla del dispositivo en el dispositivo seleccionado, pudiendo visualizar así el contenido desde este último.

- Opcionalmente, si se trata de audio o vídeo, el usuario puede controlar la reproducción desde la aplicación, pudiendo realizar las siguientes acciones: subir volumen, bajar volumen, pausar contenido, detener contenido, posicionar en el minuto que se desee, entre otras.

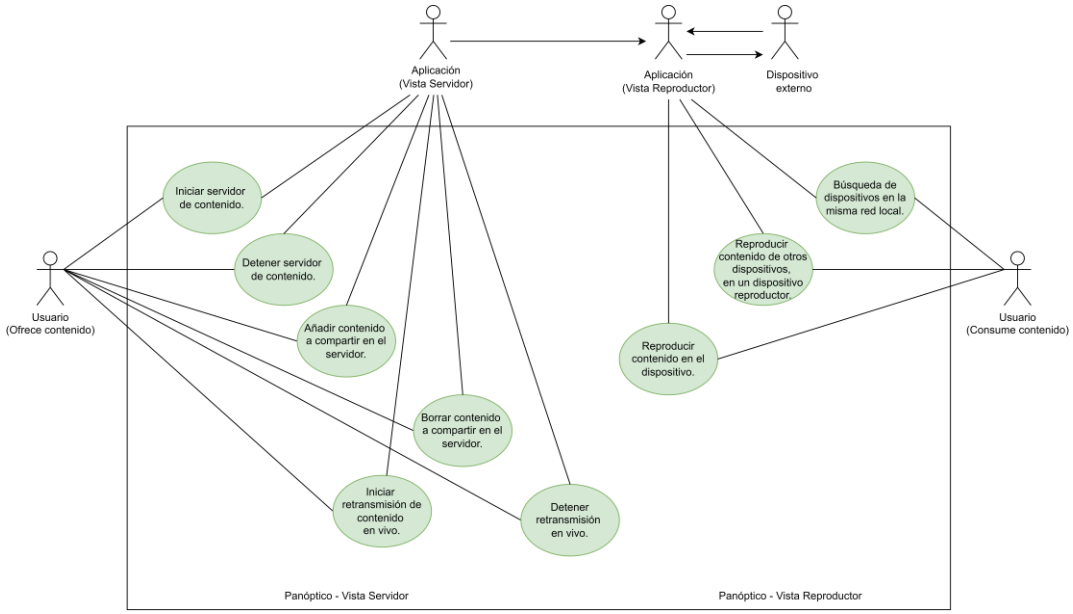

Ilustración 24. Diagrama para representar los siguientes casos de uso: CU5, CU6, CU7, CU8, CU9, CU10, CU11 y CU12.

4.1.3 Diseño de interfaz

Para establecer una idea del diseño que debería seguir la aplicación, se plantea inicialmente un diseño de baja fidelidad de las ventanas más importantes. Del mismo modo, se plantea el flujo de pantallas que seguirá la aplicación.

Como se puede observar en la ilustración 25, la interfaz de la aplicación tendrá un menú desde el que acceder a la ventana de ajustes, así como dos vistas que se pueden intercambiar con los botones de navegación inferiores.

En la vista Reproductor, se listarán aquellos dispositivos que ofrezcan contenido multimedia y se encuentren conectados en la misma red que el dispositivo. Una vez seleccionado uno de los dispositivos, se mostrará un directorio con las distintas carpetas que posea. En el caso de que ese dispositivo sea un servidor de contenido multimedia generado por la aplicación, contendrá las siguientes carpetas: Imágenes, Vídeos, Audio y Streaming. Y al acceder a cada una de ellas, se listarán aquellos archivos que se han compartido de ese tipo.

Ilustración 25. Diseño de baja fidelidad de la vista Reproductor de la aplicación Panóptico.

En la vista Servidor, se mostrará en la barra superior, accesos directos para encenderlo y apagarlo, así como para cambiar el nombre de este. Abrá un botón flotante desde el que añadir los distintos tipos de archivos y además, desde este mismo se podrá acceder a la vista Streaming, desde la cual se podrá ajustar distintos parámetros de la retransmisión del contenido en vivo. No obstante, aquellos archivos añadidos, podrán borrarse en cualquier momento, desde una vista de selección de archivos.

Ilustración 26. Diseño de baja fidelidad de la vista Servidor de la aplicación Panóptico.

Respecto al flujo de pantallas de la aplicación, quedaría un esquema bastante simple, pues se ha tratado de reducir al máximo la navegación entre pantallas y poner a disposición las distintas herramientas que ofrece, lo más accesible posible.

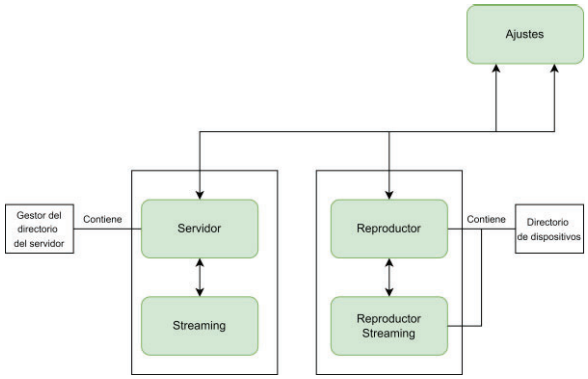

Ilustración 27. Diagrama del flujo de pantallas de la aplicación Panóptico.

4.2 Implementación

4.2.1 Creación de servicios web para la retransmisión de contenido

Para la creación de servicios web, ha sido necesario realizar dos implementaciones.

Por un lado, se ha utilizado un servidor web de contenido que acepta solicitudes a través del protocolo HTTP. Este protocolo, cuyas siglas en inglés significan "*Hypertext Transfer Protocol*", es el que permite realizar una petición de datos y recursos por parte del cliente, en este caso, los distintos dispositivos conectados a la misma red, recibiendo estos una respuesta del servidor. Por tanto, este servidor se encarga de guardar los archivos temporales seleccionados del dispositivo que adquiere este rol, para que se muestren a los dispositivos de la misma red y puedan ser consumidos. Para ello, se utilizan las peticiones propias que define el protocolo HTTP. Por parte del servidor, se emplean los métodos POST y PUT, ambos similares, aunque el primero se utiliza para actualizar un archivo temporal y otro para añadirlo, esto es, crear el recurso multimedia. Para que los dispositivos puedan reproducir el contenido multimedia, utilizarán el método GET, con el que obtienen los archivos multimedia del servidor.

Por otro lado, se ha implementado un servidor para la retransmisión de contenido en vivo, haciendo uso del protocolo WebSocket. Este, es una tecnología avanzada, capaz de abrir una sesión de comunicación interactiva, esto es, un canal de comunicación bidireccional y full-duplex, utilizando la capa TCP, entre el navegador del usuario y un servidor. Se envían datos en cualquier momento, mejorando el rendimiento y la latencia, pudiendo abrir y cerrar la conexión en cualquier momento. En este caso, para hacer uso de WebRTC para la retransmisión en vivo, el servidor manda al cliente el contenido para cargar la página, y a su vez crea el WebSocket. Por otro lado, en el lado del servidor, también se ha creado un WebSocket, para comunicarse con el cliente y haya una comunicación bidireccional.

Tanto el servidor web de contenido, como el WebSocket, se implementan con NanoHttpd, un servidor web de código abierto, pequeño y ligero, el cual es adecuado y está diseñado para ser incrustado en aplicaciones Java.

4.2.2 Publicación y descubrimiento de dispositivos UPnP

Tanto para el descubrimiento como para la publicación de dispositivos, se empleado la librería Cling, con la que se crea una instancia llamada UPnPService, a partir de la cual se puede acceder a los siguientes métodos de la API de la librería recogidos en el siguiente diagrama:

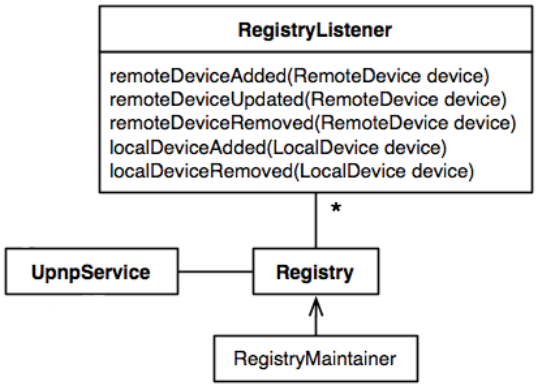

Ilustración 28. Diagrama de las interfaces Cling.Core. Fuente: 4thline.

Para ambos casos, se hace uso del módulo Registry, al cual se accede mediante a partir de UPnPService con un método llamado getRegistry(). Este módulo, se encarga principalmente de las siguientes funciones: descubrir los dispositivos UPnP de la red, administra suscripciones GENA (General Event and Notification Architecture) para conocer qué suscripciones entrantes y salientes a un servicio local que se actualiza o caduca y vence cuando es necesario, y proporciona la interfaz para añadir o eliminar instancias de RegistryListener. Este último elemento, se usa para ser notificado cuando un servicio con el que se desea trabajar está disponible en la red, tanto en un dispositivo local como remoto, y de cuándo este desaparece.

Para la búsqueda de dispositivos en la aplicación, se puede hacer de dos formas: por un lado, creando un objeto RegistryListener que actualice la lista de dispositivos que aparece o desaparecen de la red automáticamente, y, por otro lado, refrescando manualmente la lista de dispositivos. Para obtener la lista de dispositivos actualizados, tan solo se llama al método de getDevices() de Registry para recibir aquellos dispositivos que se encuentran en la red. Para acceder a la información necesaria de cada dispositivo, Cling ofrece distintas interfaces que permiten interaccionar con los servicios UPnP. También se distinguen entre dispositivos de los que se puede recibir contenido y dispositivos a los que mandar contenidos y reproducirlos, a los que se les ha denominado Renderer Devices. Dentro de estos Renderer Devices, no se detectan los dispositivos Chromecast o que empelan la tecnología Miracast, pues no se reconocen como dispositivos UPnP al funcionar con otros protocolos. Por ello, en lista de dispositivos a los que mandar contenido, se le ha añadido un acceso directo a una funcionalidad que viene en la mayoría de los dispositivos Android, conocida como Wireless Projection. Con esta función, lo que se hace es duplicar la pantalla del dispositivo y de esta manera, se consigue visualizar el contenido en aquellos dispositivos que la aplicación no detecta al utilizar otros protocolos, pero que la opción integrada en el sistema operativo Android, sí.

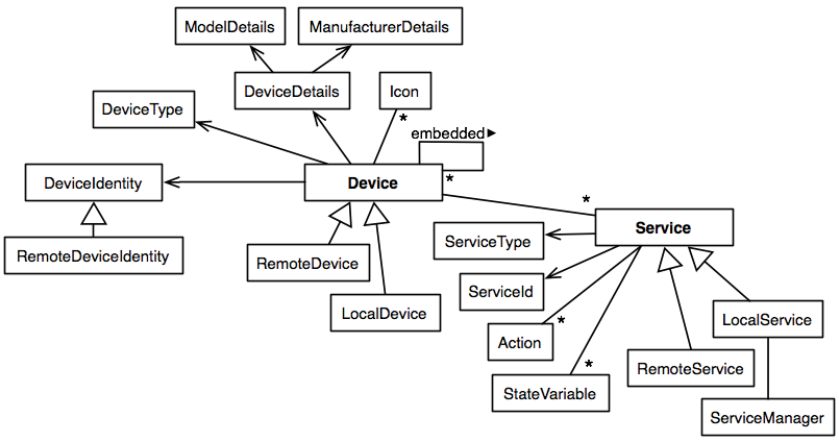

Ilustración 29. Diagrama de las interfaces Cling.Core - UPnP Devices. Fuente: 4thline

Para la publicación del dispositivo que se encarga servir el contenido multimedia y en vivo a través de la aplicación, el procedimiento es similar al anterior. Cuando se crea el módulo RegistryListener, también se crea un servicio llamado ServiceListener, cuya función es gestionar el servidor web. Este servidor web, atiende las peticiones HTTP para el contenido que se desea mostrar a otros dispositivos. Al crear este servidor, también se crea el dispositivo UPnP, un LocalDevice, y, por tanto, se hace uso de los métodos que aparecen en la ilustración 29 para Device.

Es importante añadir los metadatos necesarios, para que el dispositivo pueda ser descubierto. Por un lado, se crea los detalles del dispositivo con DeviceDetails, el cual ofrece la opción de añadir un nombre descriptivo y una información detallada del dispositivo, siendo esta información la que se mostrará a las aplicaciones UPnP. Opcionalmente, también se ha incluido tanto la información del modelo como la del fabricante, con ModelDetails y ManualfacturerDetails y se valida por si se detectara algún error en esta información. Por otro lado, creamos el DeviceType, esto es, el tipo de dispositivo junto con su versión, un entero simple. En este caso, lo hemos denominado MediaServer, cuya versión será 1, siguiendo el estándar que se basa en la especificación UDA (UPnP Device Architecture). También es necesario asignarle una identidad única, aunque se sugiere usar un UDN (Unique Device Name), para la implementación de la aplicación se ha optado por hacer uso de un UUID (Universally Unique IDentifier), que a efectos prácticos viene ser lo mismo. Lo que se requiere es que sea un identificador estable que no cambie una vez sea asignado y que se le identifique de forma que se sepa que se está tratando con un dispositivo en concreto y no otro. Este identificador se le asocia con DeviceIdentity. Por último, antes de crear el elemento LocalDevice, como se puede ver en la ilustración 29, cada dispositivo cuenta con un servicio, denominado como LocalService y su función es encapsular lo siguiente: los metadatos, qué acciones y variables de estado tiene, y cómo se puede invocar. De forma opcional, también se podría incluir un icono asociado al dispositivo, pero no se ha considerado relevante. Todos los metadatos que han sido creados, serán los parámetros necesarios para crear el elemento que se ha denominado LocalDevice, y, en consecuencia, un dispositivo UPnP. De esta manera, se queda publicado en la red local hasta que se apague el servidor, y, por ende, aparecerá en la lista de dispositivos UPnP que se pueden descubrir en dicha red.

4.2.3 Elección y retransmisión de los contenidos multimedia.

Para la retransmisión de contenido multimedia se seleccione para compartir con dispositivos conectados a la misma red, se hará uso de la librería Cling. Se parte de la premisa de que el servidor multimedia ya se ha iniciado y publicado como dispositivo UPnP, por lo que la instancia de Cling llamada LocalDevice, ya estaría creada.

Como se explica en la sección anterior, cuando se crea el LocalDevice, es necesario también crear previamente un servicio, llamado LocalService, que encapsula tanto los datos, como qué acciones y variables de estado tiene el dispositivo, y cómo se puede invocar. Este LocalService cuenta además con un gestor del servicio denominado ServiceManager, como se puede apreciar en la ilustración 29

Este ServiceManager es el que se usa para la gestión del contenido que se desea retransmitir. Este servicio extiende de una clase llamada AbstractContentDirectoryService, y, por tanto, cuenta con uno de sus métodos llamado browser. Este método será el que permite construir la estructura del directorio del contenido multimedia. Se ha decidido clasificar el contenido por carpetas, siendo estos los nombres de estas carpetas: “Imágenes”, “Vídeos”, “Música” y “Streaming”. Cada carpeta cuenta con un contador a la izquierda del número de elementos que incluye y si se hace pulsación larga en algunas de ellas, se activa un menú de selección que permite vaciar el contenido de las carpetas que se hayan seleccionado.

En el caso de la carpeta llamada “Streaming”, incluirá únicamente un elemento, y este es el que sirve de enlace a la retransmisión del contenido en vivo, si esta se encuentra activa. Al realizar esta retransmisión en vivo a través de una página web, no podrá ser reproducido por DLNA. Pese a ello, desde la aplicación se facilita la URL de la retransmisión para que pueda ser reproducido a través de cualquier navegador, y, si se hace uso de la aplicación para reproducir el contenido, podrá ser reproducido sin problema dentro de esta.

En cuanto al resto de contenidos, la aplicación permite seleccionar y elegir el contenido multimedia que se desea compartir con el resto de los dispositivos que se encuentren conectados a la red.

En función del contenido que se vaya agregando a estas carpetas, será el ServiceManager el que se encargue de actualizar el contenido de estas, permitiendo así, gestionar los archivos multimedia que se desea compartir con el resto de los dispositivos UPnP que se encuentren en la red.

4.2.4 Retransmisión de contenido en vivo

Para la retransmisión en vivo, se hará uso de la tecnología WebRTC. Por ello, es necesario iniciar previamente un servidor WebSocket. Además, también es necesario contar con un controlador WebRTC, que gestione la configuración de conexión peer-to-peer y los medios que se desean transmitir.

Desde la aplicación, el primer paso para realizar la implementación de WebRTC en aplicaciones Android es crear un objeto llamado PeerConnectionFactory. Este es el punto de entrada principal a la API de PeerConnection, esto es, la base sobre la que se hace todo. En él se puede instanciar unas opciones, entre las que se puede establecer la configuración de codificación de vídeo, que en este caso se emplea el códec de vídeo VP8. Este códec, se trata de un formato abierto y diseñado por On2 Technologi [24], y que, según Google, se usa principalmente en este tipo de conexiones con WebRTC. También se establece como estándar de compresión el H.264 High Profile. Este estándar, se emplea en aplicaciones de transmisión y almacenamiento, y es adecuado cuando se trata de emisión de vídeo en tiempo real[25].

Una vez listo PeerConnectionFactory, es necesario crear un objeto encargado de capturar el vídeo, al que se le denomina VideoCapturer, y que, en nuestro caso, varía en base a la selección previa en la aplicación de captura de la pantalla en vivo, o de la cámara. Cuando ya está preparado, se usa para crear una instancia de PeerConnectionFactory, llamada VideoSource, que contiene la información del vídeo para crear un el canal de vídeo, al que se le denomina VideoTrack. Este VideoTrack cuenta con un identificador por si luego fuera necesario eliminarlo durante la retransmisión.

Del mismo modo, hay que encargarse de crear el canal de audio, sin embargo, para este caso, no se cuenta con un objeto encargado de capturar el audio como hace el VideoCapturer. Sin embargo, sí se puede elegir la fuente de audio que utiliza. Esto es posible debido a que la configuración del audio se establece cuando se construye el PeerConnectionFactory. Para elegir qué fuente de audio utilizar del dispositivo, se ha tenido que modificar de la librería de WebRTC para Android, el módulo JavaAudioDeviceModule. Este módulo, se encarga de establecer parámetros como formato del audio, o la fuente del dispositivo que se emplea para grabar el audio, entre otros parámetros.

Por defecto, la fuente de audio está definida con el valor correspondiente, de la clase MediaRecorder.AudioSource de Android [37], de VOICE_COMMUNICATION. Este valor lo que hace es aprovechar la cancelación de eco o el control automático de ganancia si está disponible, y es adecuado para comunicaciones de voz tales como VoIP. Sin embargo, con esta configuración, no es posible recoger el audio emitido por los propios altavoces del dispositivo, pues cuenta con esa cancelación de ruido. Por ello, para los casos en los que se desee recoger la salida de audio del dispositivo, se puede aplicar el valor de DEFAULT, no aplicando así ningún ajuste al audio recogido. De esta manera, se puede hacer una selección previa en la aplicación de qué opción usar, aplicándose una fuente de audio que se adapte a las necesidades. Por tanto, únicamente es necesario crear la fuente de audio, el AudioSource necesario para crear así el canal de audio, llamado AudioTrack, a partir del PeerConnectionFactory, al igual que se hizo para el vídeo.

Con el VideoTrack y AudioTrack creado, ya solo es necesario añadir ambos al Media Stream y con esto preparado, es momento de crear el PeerConnection, esto es la conexión entre pares.

Para establecer una comunicación WebRTC, se hace uso del protocolo RTC. Por esta razón, es necesario crear una configuración previa, con un método de la API de PeerConnection para ello. Con esta configuración establecida, se crea el local peer, utilizando el método createPeerConnection de PeerConnectionFactory, cuyos parámetros son: la configuración RTC, un observador para encontrar los posible candidatos ICE que recibe, y un objeto que se encarga de detectar los MediaStream, del protocolo WebRTC, que se reciben. Sin embargo, para esta implementación, no se espera recibir ningún MediaStream, actuando este local peer como servidor de contenido. Por otro lado, se manda una señalización al servidor de señalizaciones con los siguientes parámetros SDP con el estado “true”: OfferToReceiveAudio y OfferToReceiveVideo, para crear así el offer.

Si el resultado es el esperado, se manda el tipo y la descripción del contenido por cada conexión peer-to-peer que se establece, esto es, por cada persona que recibe la retransmisión. Al detener la retransmisión se cierra la conexión peer-to-peer y, por tanto, el WebSocket.

Además, durante la retransmisión de contenido en vivo de la pantalla, es posible activar una opción para que el dispositivo que emite el contenido en vivo, pueda ser controlado remotamente por otros usuarios conectados a dicha retransmisión. Esto es posible debido a que en la conexión WebSocket, si está activada esta opción, se detectan y recogen los movimientos y opciones pulsadas por los usuarios conectados a la retransmisión, pudiendo crear una respuesta en nuestro dispositivo con la misma acción y así poder tener una interacción con el dispositivo que ofrece el contenido.

Para que pueda realizarse la retransmisión, es necesario que, en el lado del cliente, se conecten a través de un navegador. Para la aplicación, se usado un WebViewer, un elemento de Android que actúa como navegador y que permite acceder al enlace de la retransmisión. Cuando se accede a través del enlace, se reciben un archivo HTML que carga el contenido de la página, además de unos archivos Javascript, necesarios para que se establezca la conexión con el servidor WebSocket y crea así la conexión peer-to-peer con el local peer que actúa como servidor desde la aplicación donde se realiza la retransmisión en vivo.

En de uno de los archivos Javascript, se encuentran los métodos necesarios para realizar el mismo procedimiento que hemos realizado desde la aplicación. Esto es, se encarga de inicializar el Websocket y crear la conexión peer-to-peer siguiendo los pasos necesarios para recibir así el MediaStream que genera el dispositivo que actúa como servidor. Por esta razón en este lado del cliente, no se genera el MediaStream, pues no se desea mandar al lado que actúa como servidor de la retransmisión ningún VideoTrack o AudioTrack.

El resto de los archivos Javascript que se reciben, se encargan de adaptar la resolución y el tamaño de visualización del vídeo, haciéndolo compatible según el navegador desde el que se visualice la retransmisión. También se encargan de registrar los gestos y opciones seleccionadas, para mandarlas al servicio encargado de realizar las interacciones en el dispositivo desde donde se emite la retransmisión.

4.2.5 Reproducción de los contenidos recibidos desde la aplicación

Para reproducir los contenidos multimedia y en vivo de los distintos dispositivos UPnP, se hace uso de la librería Cling, con la que se crea una instancia llamada UPnPService, a partir de la cual se puede acceder a los siguientes métodos de la API de la librería recogidos en el siguiente diagrama:

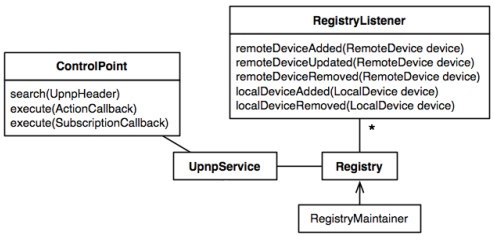

Ilustración 30. Diagrama de las interfaces Cling.Core. Fuente: 4thline

No solo se cuenta con la instancia del UPnPService, sino que además es necesario que ya se hayan descubierto dispositivos UPnP que ofrezcan contenido multimedia, para la reproducción de estos.

Al seleccionar uno de estos dispositivos, UPnPService registra que se ha elegido visualizar el directorio de ese contenido y a partir del servicio ServiceListener, que se crea con el RegistryListener, se puede obtener el AndroidUPnPService. Este último se crea, cuando inicia el servicio para arrancar el servidor web de contenido. Será este AndroidUPnPService, el que dé el acceso a la interfaz de ControlPoint que ofrece distintos métodos.
Por un lado, el método llamado search, que se usa cuando se inicia el servidor y se publica el dispositivo UPnP con la aplicación. En este punto, no se conoce ningún dispositivo y servicio UPnP que pueda estar disponible, por ello, lo que hace es transmitir un mensaje de búsqueda para todos los dispositivos UPnP que se encuentren en la red y decide si debe responder directamente (con datagramas UDP de notificación) al punto de control de envío. Los mensajes de búsqueda llevan un encabezado y los receptores consideran este encabezado cuando evalúan una respuesta potencial.

Por otro lado, también se tiene acceso al método llamado execute para ejecutar las diversas acciones que ofrecen los servicios. La ejecución de acciones se procesa de forma asincrónica, de manera que hay un ActionCallback, el cual será notificado cuando la ejecución fue un éxito (con valores de resultado) o una falla (con código de estado de error y mensajes).

Por tanto, se ejecuta este método para realizar búsquedas entre los directorios con la acción que ofrece Cling, llamado Browse. De esta manera podemos navegar entre los directorios, de manera que hay un ContentCallback que se encarga de recibir y listar los contenidos multimedia recibidos o de avisar si ha habido algún error.

Cada vez que se seleccione un dispositivo UPnP, gracias al ControlPoint, podremos recibir el contenido que ofrece y visualizarlo en la aplicación o bien mandar este contenido a un Renderer Device. Para mandar el contenido a un Renderer Device, también será necesario hacer uso de un ControlPoint, ya que será necesario controlar el contenido desde la aplicación y mandar acciones que ejecutar en este caso al dispositivo UPnP que se encargue de reproducir el contenido. Entre las acciones que se ejecutan y Cling ofrece para controlar la reproducción, se encuentran: Play, Stop, Pause, Seek, SetVolume, SetMute, SetAVTransportURI, GetMediaInfo, GetPositionInfo y GetTransportInfo.

Por otro lado, si se desea transmitir el contenido a un dispositivo con la tecnología Miracast, se hará por medio de la tecnología Wireless Projection que ofrece Android. Estos dispositivos no se recogerán entre los RendererDevices, ya que únicamente se listan los dispositivos UPnP que son descubiertos, pero sí que se incluye un acceso directo a este ajuste, en el que se listan los dispositivos compatibles con la tecnología Miracast. Una vez se seleccione el dispositivo y se duplique la pantalla en el dispositivo donde se desea visualizar el contenido, tan solo tiene que volver a la aplicación y mostrarlo desde esta misma.

4.2.6 Diagrama de la arquitectura software

A continuación, se puede ver un diagrama de la arquitectura software, en el que se relacionan todas las funcionalidades implementadas, de la aplicación Panóptico.

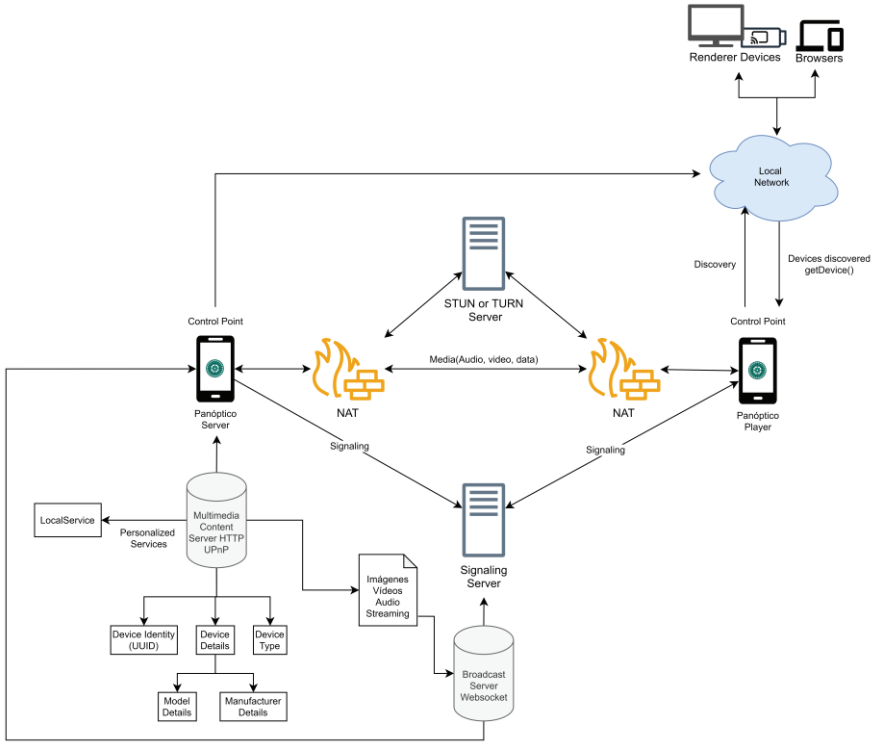

Ilustración 31. Diagrama de toda la arquitectura software de la aplicación Panóptico.

4.2.7 Funcionamiento e interfaz de la aplicación

En esta sección, se explica en detalle el funcionamiento de la aplicación según la ventana en la que se encuentre, así como el diseño que se ha implementado finalmente para poder ejecutar todas las funcionalidades de la aplicación de una forma intuitiva para el usuario.

4.2.7.1 Interfaz principal

Al iniciar la aplicación, se muestra una pantalla de inicio, que muestra el logo de la aplicación y sirve para que cargue todos los componentes necesarios de la aplicación.

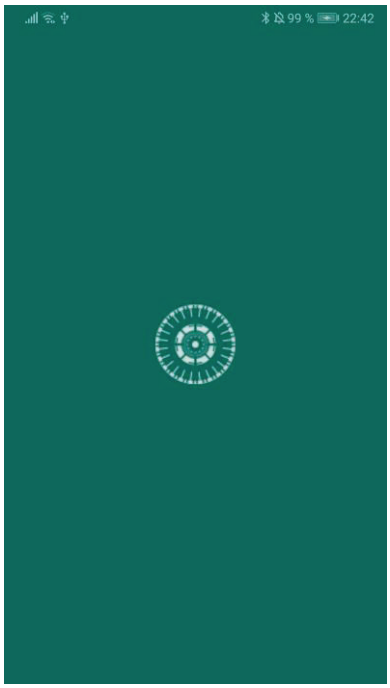

Ilustración 32. App Panóptico: Pantalla de inicio.

La interfaz principal se compone de una barra de navegación inferior que con dos secciones. La primera, denominada Reproductor, la cual muestra una lista con los dispositivos que se han descubierto en la red local y acceder así al directorio del contenido que ofrecen para poder reproducirlo desde la aplicación o mandarlo a otro dispositivo para que reproduzca el contenido. La segunda sección, llamada Servidor, ofrece la posibilidad de activar o desactivar el servidor, así como una vista del directorio de contenido multimedia que ofrece el servidor, y un botón flotante desde el que poder añadir el contenido multimedia que se desea compartir. Estas vistas se pueden ver en la ilustración 33 y 43.

Ilustración 33. App Panóptico: Vista Reproductor.

Por otro lado, en la barra de acciones superior se dispone siempre de un menú. Además, si se está en la sección de Reproductor, muestra junto a la opción de menú, un botón al que se le ha denominado Cast. Este botón muestra una ventana emergente que lista los distintos dispositivos UPnP capaces de reproducir contenido multimedia, que se descubren en la red, y de esta manera poder elegir y asignar uno de ellos como reproductor de contenido cuando se usa la opción de mandar contenido a un reproductor externo. Además de listar estos dispositivos, cuenta con un acceso directo a la opción de Wireless Projection del dispositivo, por si se desea emplear un dispositivo compatible con la tecnología Miracast en vez de UPnP, y en vez de mandar el contenido, se duplica la pantalla de manera que el contenido que se visualiza en la aplicación, también podrá verse desde ese dispositivo.

Por el contrario, si se está en la sección Servidor, a la izquierda de la barra se dispondrá de un botón con el que poder activar y desactivar el servidor de contenido multimedia junto al nombre del servidor, el cual, si se pulsa, te permite cambiar su nombre.

El menú, en cambio, siempre lista dos opciones: la de acceso a la ventana de ajustes y la de salir de la aplicación.

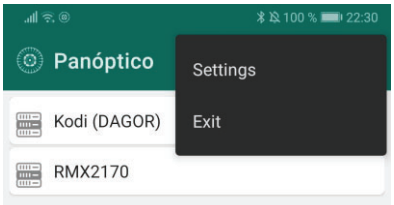

Ilustración 34. App Panóptico: Opciones del menú.

Además, durante el tiempo que se encuentre activa la aplicación, se mostrará una notificación permanente, esto es, que no se podrá descartar, siendo esta otra alternativa para cerrar la aplicación desde la opción de cerrar como indica el mensaje de la notificación de la notificación, pues en ningún momento se encontrará la aplicación en el gestor de aplicaciones recientes de la aplicación.

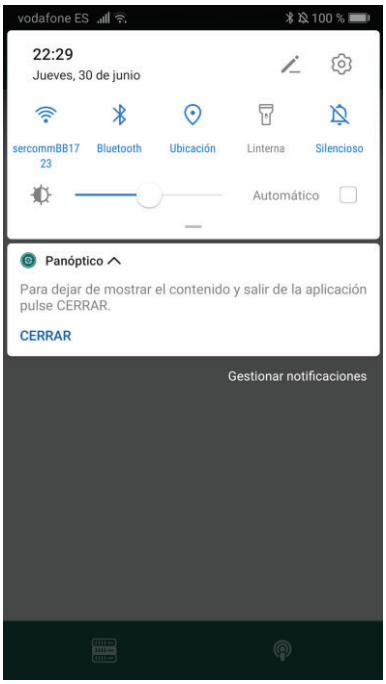

Ilustración 35. App Panóptico: Notificación permanente de la aplicación.

4.2.7.2 Ajustes

En esta ventana se listan dos secciones.

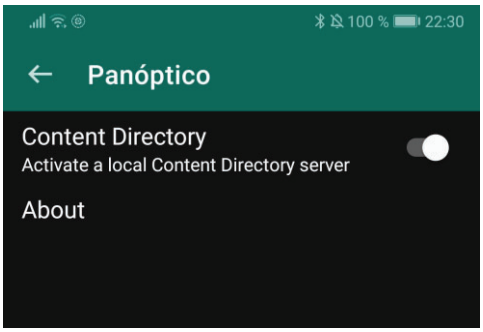

Ilustración 36. App Panóptico: Vista de Ajustes.

En la primera sección, además de poder activar o desactivar el servidor desde esta, incluye los siguientes ajustes del servidor: cambiar el nombre del dispositivo UPnP, cambiar el puerto del servidor de contenido y cambiar el puerto del servidor encargado de la retransmisión en vivo. Si se desea cambiar algunos de los puertos, aparecerá una ventana de aviso en el que se indica que va a reiniciar la aplicación, ya que, si no se reinicia, no se aplican los cambios de los puertos en la configuración de los servidores.

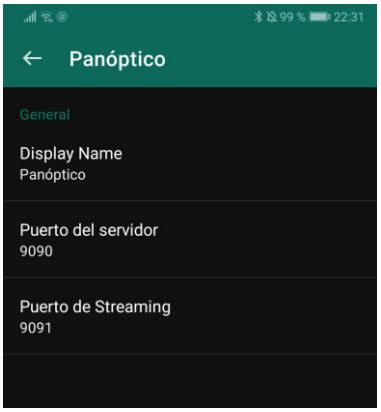

Ilustración 37. App Panóptico: Opciones de ajustes del Servidor de contenido.

En la segunda sección, llamada About, se los datos de la aplicación como la versión de la aplicación, la licencia, acceso al código fuente o las licencias de las librerías que se han empleado para el desarrollo de la aplicación.

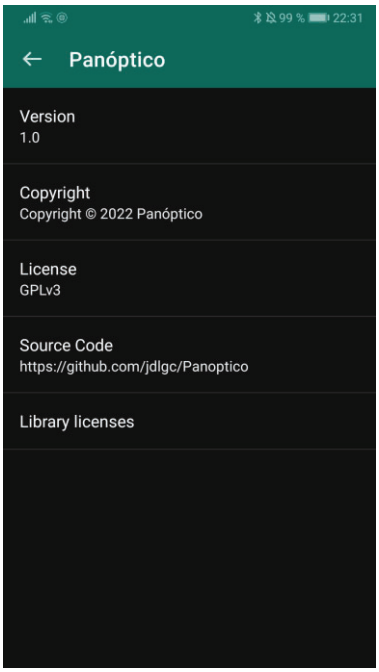

Ilustración 38. App Panóptico: Vista de información de la aplicación.

4.2.7.3 Vista Reproductor

Como se ha explicado previamente en la interfaz de la aplicación, en esta vista se listan los dispositivos UPnP descubiertos en la red además de mostrar junto a la acción de ajustes, el botón Cast en la barra superior de acciones.

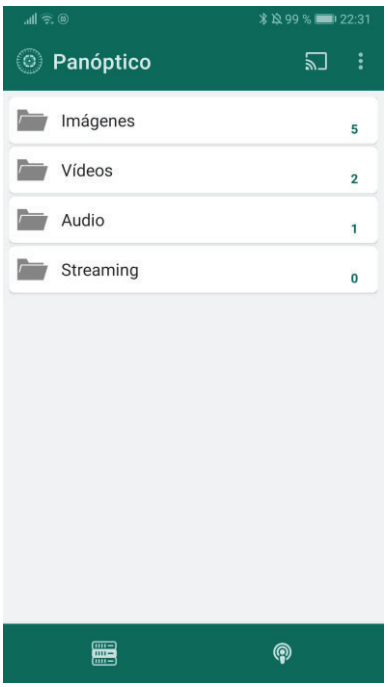

Ilustración 39. App Panóptico: Directorio de un dispositivo en la vista Reproductor.

Cuando se selecciona uno de los dispositivos descubiertos, se puede acceder al directorio de los contenidos multimedia que ofrece dicho dispositivo.

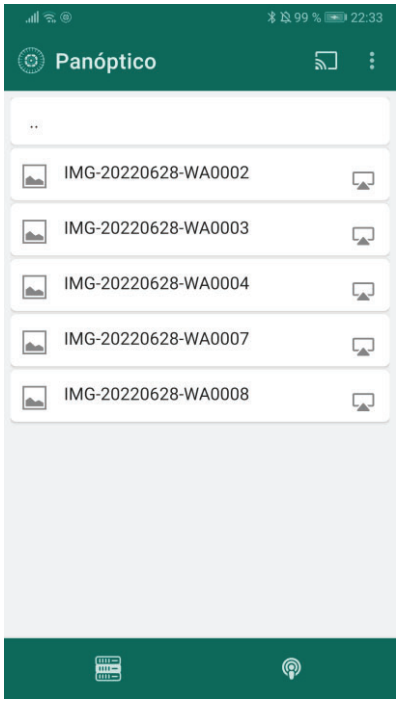

Ilustración 40. App Panóptico: Archivos del directorio de un dispositivo descubierto, en la vista Reproductor.

Si se pulsa sobre un elemento multimedia, por defecto, se usarán las aplicaciones oportunas del dispositivo para reproducir el contenido. Sin embargo, a la derecha de cada elemento, también se dispone de un botón, para lanzar el contenido al dispositivo UPnP para la reproducción de medios asignado desde la ventana emergente del botón Cast. Esta última acción, sería equivalente a hacer una pulsación larga sobre el elemento que se desea mandar al reproductor de contenido de la red en la que se encuentra.

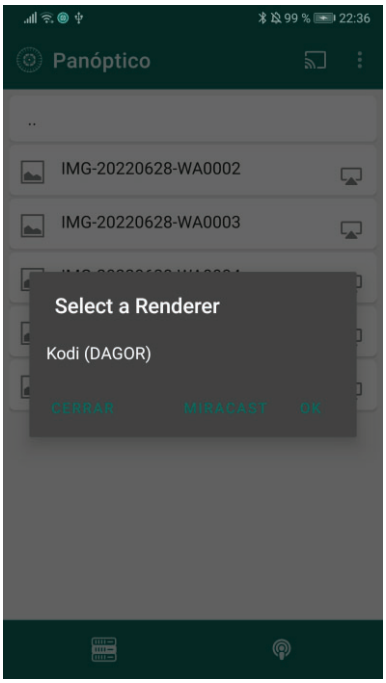

Ilustración 41. App Panóptico: Vista de dispositivos Renderer.

Si el elemento que se desea visualizar es Streaming, que se encuentra dentro de su correspondiente carpeta, cuando se pulsa sobre él, se abrirá una ventana emergente, desde la que se ofrece la posibilidad de compartir el código QR junto con el enlace de la retransmisión, o abrir la retransmisión desde el navegador. Sin embargo, si se pulsa la opción de abrir, la aplicación abrirá una ventana desde la que poder visualizar la retransmisión en vivo.

Si se deseara acceder al elemento Streaming, a través de un dispositivo que emplee por ejemplo la tecnología DLNA, no sería posible. Esto se debe, a que este elemento está únicamente preparado para ser reproducido a través de la aplicación. Sin embargo, existe dos alternativas para poder reproducirlo en otro dispositivo donde no esté instalada la aplicación: o bien en aquellos dispositivos que sean compatibles con la tecnología Miracast y desde la ventana emergente del botón Cast, ir a e la opción Miracast que te redirige a los ajustes de Wireless Projection y te permite duplicar la pantalla del dispositivo con la aplicación desde donde sí se puede reproducir la retransmisión en vivo, o bien accediendo con la URL de la retransmisión desde cualquier navegador que posea ese dispositivo.

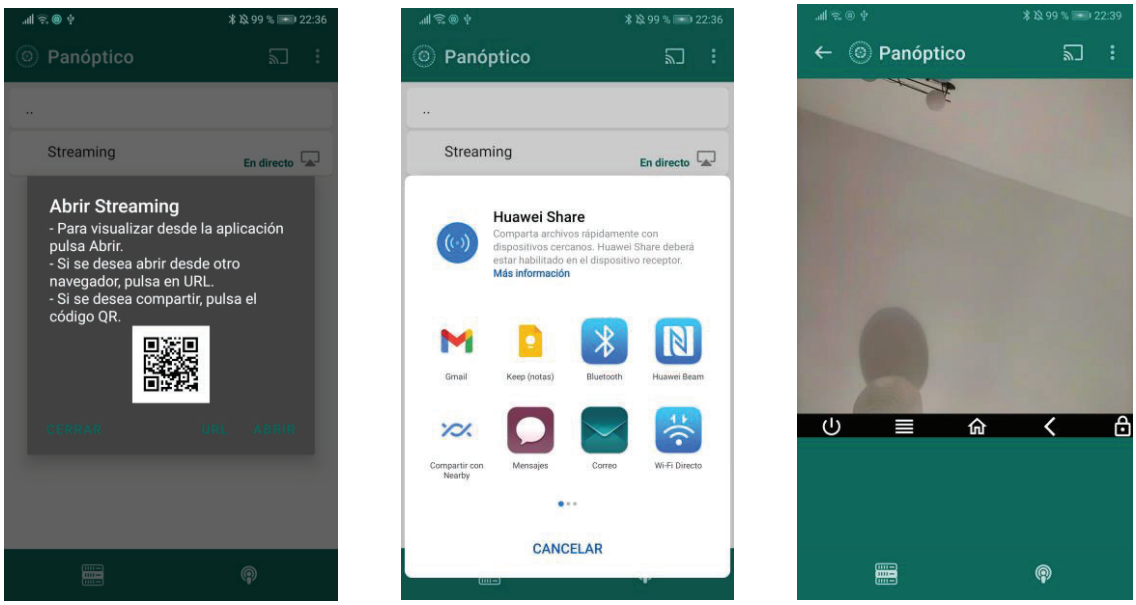

Ilustración 42. App Panóptico: Opciones para abrir un Streaming.

4.2.7.4 Vista Servidor

En la vista de servidor, como se explicaba en la interfaz de la aplicación, en la barra de acciones superior, además del menú, se muestra un botón para activar o desactivar el servidor junto al nombre del dispositivo. Si se pulsa sobre el nombre, se abre una ventana emergente desde poder cambiarlo, sin necesidad de tener que ir a Ajustes para ello.

Desde esta ventana, además, se puede visualizar el directorio del contenido que se mostrará a otros dispositivos mientras que el servidor esté activo.

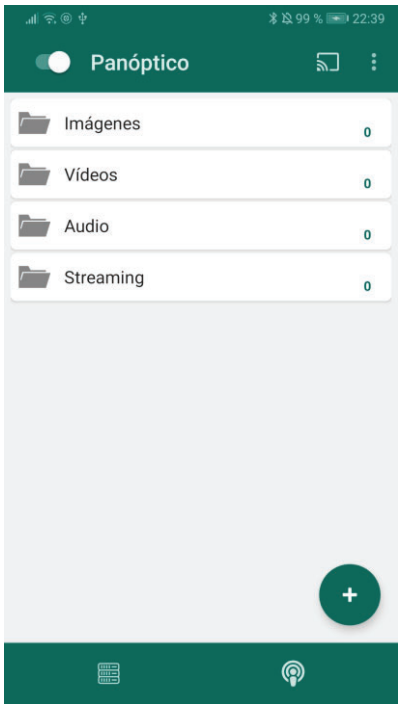

Ilustración 43. App Panóptico: Vista Servidor.

También cuenta con un botón flotante en la esquina inferior derecha, desde el cuál si se pulsa, se despliega un conjunto de botones desde los que se puede agregar distintos tipos de archivos multimedia al directorio del servidor, o bien acceder a la vista Streaming.

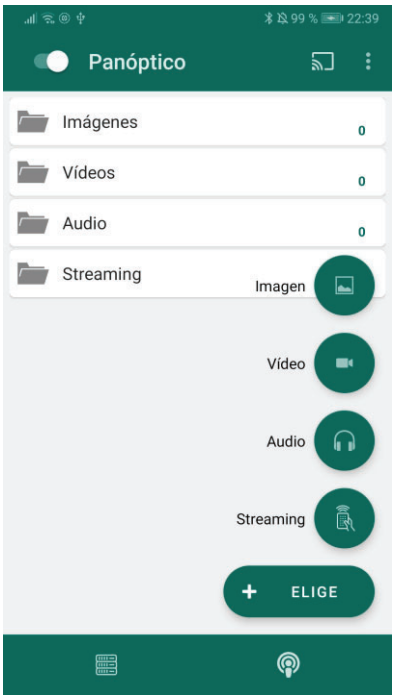

Ilustración 44. App Panóptico: Añadir archivos en la vista Servidor.

Para explicar cómo se añadirían estos archivos multimedia, se usará como ejemplo la opción de añadir imagen. Al pulsar sobre esta, se abrirá el explorador de archivos del sistema con todos los archivos con formato imagen que encuentre, de manera que se puede seleccionar las que se deseen añadir al directorio del servidor. De forma análoga se haría para añadir archivos de vídeo y audio.

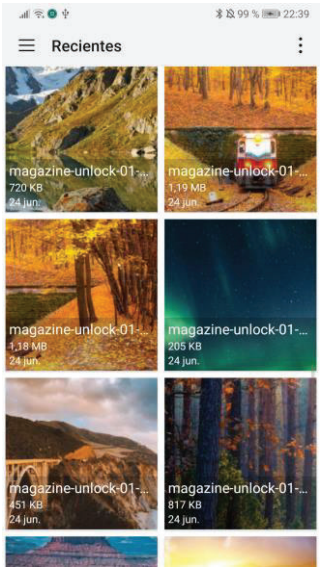

Ilustración 45. App Panóptico: Selección de archivos en la vista Servidor.

Al igual que se pueden añadir al directorio, se pueden eliminar de este. Para ello, existen distintas opciones. Si te encuentras dentro de una carpeta específica, y se desea eliminar un archivo multimedia de esa carpeta, tan solo es necesario hacer pulsación larga sobre ese elemento y se abrirá un menú de selección. En este menú de selección se puede ir seleccionando uno por uno, los archivos que se desean eliminar. Existe la opción de seleccionar o deseleccionar todos a la vez, que se encuentra junto al botón de borrar cuando aparece este menú de selección. Al pulsar el botón de eliminar, los archivos seleccionados se eliminarán de dicha carpeta.

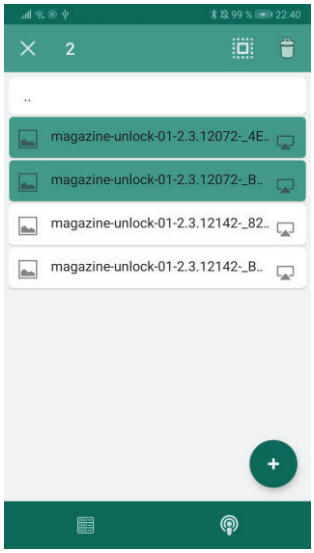

Ilustración 46. App Panóptico: vista de selección habilitada para la gestión de archivos del directorio Servidor.

Siguiendo el mismo procedimiento en la vista de carpetas del directorio, la opción de borrar lo que haría sería vaciar el contenido de las carpetas seleccionadas, salvo el de la carpeta Streaming, pues para ello, habría que detener la retransmisión desde la vista Streaming.

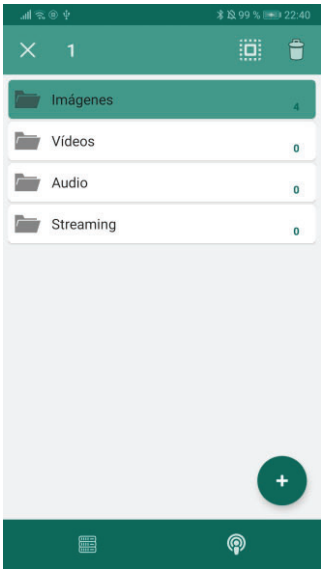

Ilustración 47. App Panóptico: vista de selección habilitada para la gestión de carpetas del directorio Servidor.

4.2.7.5 Vista Streaming

Desde esta ventana, se podrá iniciar y detener una retransmisión en vivo.

En la parte superior, se encuentra un código QR del enlace de la retransmisión junto con la dirección URL, desde la cual permite a cualquier dispositivo de la red a través de un navegador, conectarse a la retransmisión en vivo. Si se pulsa sobre estos, se mostrará la opción para el enlace o el código QR.

Además, la vista cuenta con dos desplegables. En el desplegable Vídeo, se podrá elegir entre compartir la pantalla o la cámara del dispositivo. Si se comparte la pantalla, se podrá habilitar o deshabilitar la opción de control remoto desde donde se visualice la retransmisión. Si, por el contrario, se selecciona la opción de cámara, esta se podrá intercambiar entre cámara frontal o cámara trasera.

En cuanto al desplegable Audio, se podrá elegir entre solo el audio del micrófono al que se le aplica una cancelación de ruido para mejor captura del sonido, o si por el contrario se desea recoger también el audio de los altavoces del dispositivo, esto es, el sonido interno del dispositivo.

El ajuste de las diversas opciones que se ofrecen según el contenido de vídeo que se elija, aparecen a la derecha, encima del botón flotante que se encuentra en la esquina inferior derecha, junto a la opción de silenciar o no el micrófono.

Por último, el botón flotante mencionado, nos permitirá iniciar o detener la retransmisión del contenido en vivo, con los ajustes de video y audio elegidos. Cuando la retransmisión se inicie, se mostrará además una vista previa de esta retransmisión, tal y como aparecerá al resto de dispositivos, cuando se conecten a esta.

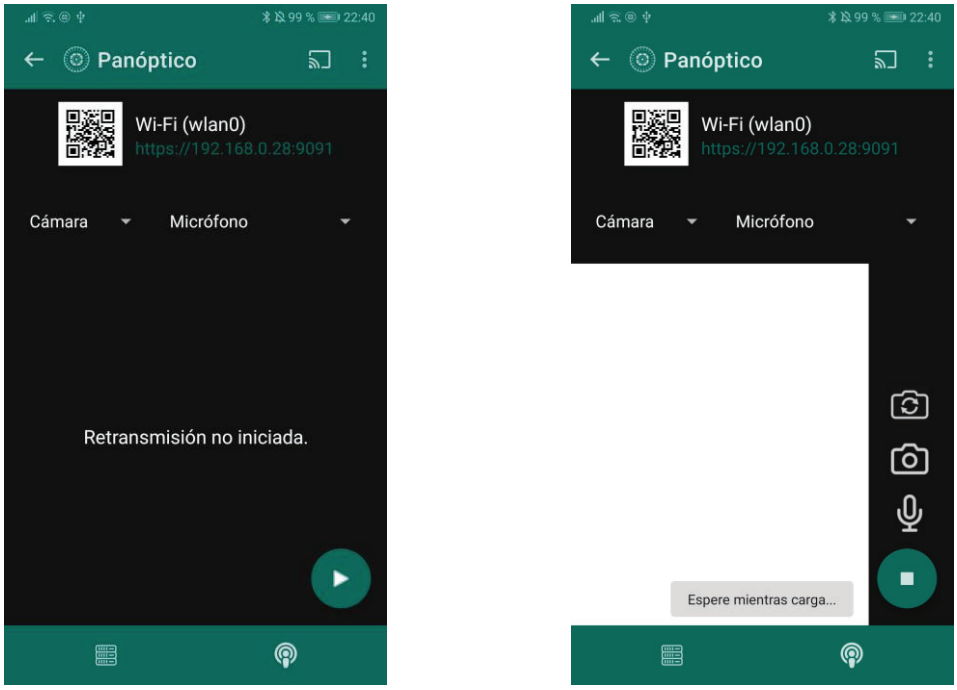

Ilustración 48. App Panóptico: Vista Streaming.

5 Pruebas y evaluación

Para probar las distintas funcionalidades que presenta la aplicación, se han definido una serie de pruebas, que se han realizado con distintos dispositivos y sujetos.

5.1 Iniciar servidor, modificar ajustes de este y apagarlo

Esta prueba, se ha de realizar desde la propia aplicación, comprobando su funcionamiento desde otros dispositivos que permitan el descubrimiento de dispositivos, como puede ser una aplicación con la tecnología DLNA o la propia aplicación instalada desde otro dispositivo Android.

La prueba consiste en activar o desactivar el servidor y por tanto comprobar si se publica o se oculta el dispositivo, entre los dispositivos de la red local. También probar a cambiar el nombre como dispositivo UPnP del servidor de contenido, así como los puertos correspondientes al servidor de contenido y servidor de retransmisión de contenido en vivo.

Para ejecutar la prueba, se puede realizar desde distintos puntos de la aplicación. Para encender o apagar el servidor, se puede hacer tanto desde la vista de servidor en la barra de acciones superior, como desde los propios ajustes. Lo mismo ocurre para cambiar el nombre como dispositivo UPnP del servidor de contenido. Sin embargo, para los ajustes de los puertos, únicamente se puede hacer desde Ajustes. Los valores que vienen por defecto, no presentan conflicto, siempre y cuando no coincidan entre ellos o con otros ya asignado en la red local.

Los resultados para todos los casos son satisfactorios. Cuando se inicia el servidor aparece como dispositivo UPnP dentro de la red local. Cuando se modifica el nombre, se ocultar y vuelve a mostrar para aparecer con el nuevo nombre asignado. Sin embargo, para que el cambio de los puertos sea efectivo, se requiere que la aplicación se reinicie. Por ello, se añadió una ventana emergente en la que informa al usuario sobre la necesidad de reiniciar para aplicar los cambios y que reinicia la aplicación automáticamente cuando se pulsa sobre Aceptar en la ventana.

La prueba además se ha realizado con distintos sujetos. A algunos de ellos se le comentaba la opción de poder activar o desactivar o cambiar el nombre desde la vista Servidor, y ejecutaban los cambios desde ahí. A los que no, lo utilizaban por intuición o bien se iban a la ventana de ajustes, para realizar los ajustes oportunos.

5.2 Añadir o borrar contenido multimedia a mostrar y en otros dispositivos compatibles con la tecnología UPnP o Miracast

Esta prueba, se ha de realizar desde la propia aplicación, comprobando su funcionamiento desde otros dispositivos que permitan el descubrimiento de dispositivos, como puede ser una aplicación con la tecnología DLNA o la propia aplicación instalada desde otro dispositivo Android.

La prueba en este caso, consiste en comprobar que se añade el contenido multimedia que se desea compartir a las carpetas de los directorios correspondientes, y de estos archivos añadidos, comprobar que se pueden eliminar, bien seleccionando uno por uno dentro de cada carpeta, o desde la vista de carpetas, vaciando así su contenido.

Para poner en práctica esta prueba, es necesario estar en la vista Servidor, desde la cual se tiene acceso a un botón flotante abajo a la derecha que nos ofrece distintas opciones en base al tipo de contenido multimedia que se desea añadir. Si se selecciona Imagen, se abre el explorador de archivos del dispositivo, mostrando únicamente aquellos que correspondan al tipo imagen. Desde aquí se puede seleccionar aquellas que se quiere incluir en el repositorio, y se le da a añadir, pudiéndose comprobar que se añaden correctamente y que se encuentran visibles desde otros dispositivos. Del mismo modo, se procede a hacer esta prueba con los archivos de vídeo y audio.

Para comprobar que se eliminan los archivos, se hace la prueba de tres formas diferentes. Por un lado, se eliminará una imagen en concreta y para ello, se tendrá que acceder a la carpeta de imágenes. Dentro de la carpeta, se hace pulsación larga sobre ese archivo, y se habilitará la vista de selección de archivos, por lo que solo bastará con pulsar el botón de borrar. Por otro lado, se va a vaciar el contenido de la carpeta de vídeos, desde dentro de esta. Para ello, se hace pulsación larga sobre uno de los archivos, habilitándose así la vista de selección, y se pulsa el botón que se encuentra al lado de borrar que permite seleccionar todos los archivos a la vez. Una vez seleccionado todos, se pulsa el botón de eliminar, para borrar todos los archivos de esa carpeta, automáticamente. Por último, se va a vaciar la carpeta de audio desde la vista del directorio con el listado de las carpetas de los distintos tipos de archivos. Se selecciona la carpeta haciendo pulsación larga para que se habilite la vista de selección y se pulsa el botón de borrar. En este último caso, se vacía la carpeta, y el contador de elementos de la carpeta, se queda a 0 sin borrar la carpeta como tal.

De cara a los resultados, se puede comprobar desde el otro dispositivo empleado para visualizar el contenido, que se actualiza, mostrando los elementos que se desean y borrando aquellos que dejan de estar disponible en el servidor de contenido. También se comprueba que se puede acceder al contenido multimedia desde la aplicación. Por último, se busca dispositivos UPnP al que mandar el contenido para que puedan ser reproducidos, pudiendo además controlar los archivos multimedia de vídeo y audio. También se prueba la opción para duplicar pantalla con Wireless Projection con dispositivos compatibles con la tecnología Miracast, cumpliendo así con uno de los requisitos establecidos.

Al ser un procedimiento similar al de otras aplicaciones de explorador de archivos, los sujetos que realizaron esta prueba, no tuvieron problemas con estas tareas.

5.3 Iniciar o detener una retransmisión en vivo y reproducirlo desde otros dispositivos

Esta prueba, se ha de realizar desde la propia aplicación, comprobando su funcionamiento desde otros dispositivos que cuenten con la tecnología Miracast, desde un navegador o la propia aplicación instalada desde otro dispositivo Android.

La prueba trata de comprobar si se inicia el servidor para la retransmisión en vivo y si es posible visualizar a través de distintas opciones. También se comprueba, las distintas opciones que ofrece una retransmisión en vivo, como puede ser en el caso de vídeo, elegir entre retransmitir la pantalla del dispositivo y el control remoto de este, entre las distintas cámaras del dispositivo u ocultar el vídeo que se emite. Del mismo modo, con las distintas opciones de audio, además de poder silenciar el micrófono o no. Por último, comprobar que cuando se detiene la retransmisión, deja de emitir dicho contenido al resto de dispositivos.

Para testear esta prueba, lo primero que se ha hecho ha sido comprobar que se inicia la retransmisión con los parámetros por defecto: la cámara como video a retransmitir y micrófono con cancelación de ruido activa. Para comprobar que se está retransmitiendo el contenido, se ha probado a visualizar el contenido desde la aplicación instalada en otro dispositivo, el navegador de Safari desde un iPad, y el navegador de Edge desde el ordenador. Todos pueden unirse a la retransmisión, visualizando el contenido correctamente, tanto vídeo como audio. Se comprueba, además, que puede intercambiarse la cámara durante la retransmisión, silenciar el audio u ocultar la pista de vídeo que se transmite. Por otro lado, tratamos de cambiar la fuente de vídeo a pantalla, y el audio para que recoja además el sonido interno del dispositivo. Para ambos cambios, el servidor, y, por tanto, la retransmisión, se reinicia, estando preparada la página para que se reinicie también y vuelva a reconectarse a la retransmisión. En este caso se puede visualizar la pantalla del dispositivo mientras se escucha la música que se está reproduciendo. También se puede activar y desactivar el control remoto, pudiendo interactuar con el dispositivo que está realizando la retransmisión y un espectador, por ejemplo, puede cambiar la canción que se está reproduciendo.

Por último, se detiene la retransmisión y se comprueba desde los 3 dispositivos que se estaba reproduciendo que pierde la conexión y, por tanto, no puede acceder al contenido de la solicitud a la página web de la red local.

Por tanto, esta prueba verifica que el funcionamiento de la retransmisión de contenido en vivo, tiene un comportamiento esperado.

Aunque se ha realizado desde distintos dispositivos, se ha usado distintos sujetos tanto para iniciar la retransmisión, como para la visualización de esta desde la aplicación, además de probar la opción de compartir el enlace del directo o el código QR.

6 Resultados, conclusiones y líneas de futuro

En este capítulo, se recogerán los resultados obtenidos tras la realización de este trabajo. Además, también se exponen las conclusiones obtenidas y las posibles líneas futuras de desarrollo que podrían llevarse a cabo.

6.1 Resultados y conclusiones

La implementación de esta aplicación para Android ha sufrido numerosos cambios en el planteamiento, así como, funcionalidades añadidas con respecto a la idea inicial y los objetivos establecidos en el capítulo 1.

Para su desarrollo, he necesitado realizar previamente una investigación de las posibles tecnologías que se podrían utilizar para el desarrollo de esta aplicación. Esta búsqueda ha sido de vital importancia, puesto que me ha permitido hallar la mejor solución posible en cuanto al desarrollo de la misma. Además, me ha aportado un gran conocimiento de tecnologías que desconocía, como es el caso de WebRTC, empleada para la funcionalidad de retransmisión en vivo. También, aunque ya contaba con conocimientos previos de programación en el lenguaje Java para Android, he aprendido nuevas formas de implementación, como el uso de fragmentos dentro de un Activity, en vez de usar varios Activities que realicen lo mismo. Usar el componente Navigation para la navegación entre distintos fragmentos, en vez de usar el administrador de fragmentos FragmentManager, así como, distintos recursos que encajaban con el diseño y las funcionalidades de la aplicación. Esto ha hecho que, además de aplicar conocimientos previos, pudiera aprender nuevas formas de programar a la par que nuevas tecnologías en su profundidad. Esto no hubiera sido posible si no se hubiese llevado a cabo este proyecto.

Como resultado final, se ha conseguido, por un lado, que la aplicación actúe como servidor de contenido multimedia entre dispositivos UPnP que se encuentran conectados a la misma red local, tal y como se había preconcebido. Por otro lado, se ha implementado la retransmisión de contenido en vivo, pero esta no puede ser reproducida a través de dispositivos con la tecnología UPnP. No obstante, la solución empleada con la tecnología WebRTC, ha permitido ampliar que no solo se pueda visualizar la retransmisión a través de la aplicación, sino que también quepa la posibilidad de poder ser reproducida desde cualquier navegador, siempre y cuando el dispositivo se encuentre conectado a la misma red local. Además, esta solución integra la posibilidad de poder usarlo como control remoto del dispositivo, cuya funcionalidad no había sido planteada desde un principio. Esto no hubiera sido posible de haber implementado una solución que sí hubiera sido compatible su reproducción en dispositivos UPnP de la red local.

Por tanto, la aplicación ha logrado cumplir las funcionalidades y los requisitos propuestos además de ampliar las funcionalidades previamente esperadas. De una forma u otra, se consigue visualizar el contenido multimedia compartido, no solo desde la aplicación, si no desde otras alternativas que hacen compatibles a cualquier dispositivo que no disponga del sistema operativo Android o de la aplicación. Por ejemplo, el contenido multimedia se puede visualizar desde VLC, un popular reproductor de dicho contenido, a través de la opción de visualizar contenido de la red local, Plug'n Play Universal. Por otro lado, para la reproducción del contenido en vivo, puede usarse cualquier navegador desde cualquier dispositivo que esté conectado a la misma red.

Se podría decir, por tanto, que la única limitación de esta solución, es disponer de un dispositivo Android con la aplicación para poder generar o compartir el contenido multimedia que se desea.

A nivel personal, esta aplicación me ha permitido, como he mencionado anteriormente, aprender y descubrir nuevas tecnologías, así como, aplicar conocimientos adquiridos durante el grado de Ingeniería Informática, llegando a lograr un resultado del cual me siento realmente satisfecho.

6.2 Líneas futuras de desarrollo

A medida que se ha ido desarrollando el proyecto y se han realizado pruebas de sus funcionalidades y de la aplicación con distintos sujetos, se han recogido una lista de posibles mejoras que añadir a la aplicación.

Una de ellas podría ser un acceso de búsqueda de archivos y carpetas entre los directorios de los dispositivos encontrados, o bien, del propio servidor. En relación con esto, la creación de carpetas y directorios, haría que fuera menos restrictivo y más personalizado a la hora de añadir contenido.

También podría ser interesante la integración de Cast SDK de Google, de manera que no sea necesario usar Wireless Projection para poder retransmitir el contenido multimedia o en vivo, duplicando la pantalla de un dispositivo Android, si no mandar el contenido directamente al dispositivo compatible con la tecnología Miracast.

Otra posible mejora, sería, por ejemplo, para el caso de la retransmisión en vivo, la petición de una clave de acceso, de manera que haga aún más privado el acceso a cada retransmisión dentro una misma red local.

Con estas posibles mejoras, podría pasarse a la implementación de esta aplicación en otros sistemas operativos, como puede ser iOS o incluso Windows. De esta forma, abriría el abanico de posibilidades con las funcionalidades que ofrece la aplicación actualmente.

Todos estos posibles añadidos, podrían ser continuados en la realización del Trabajo Final de Máster, añadiendo estas funcionalidades y otras futuras que pudiesen resultar útiles aplicando los nuevos conocimientos adquiridos durante esta etapa académica.

7 Análisis de impacto

En este capítulo se ha realizado un análisis del impacto potencial de los resultados obtenidos durante el desempeño de este trabajo. Por un lado, se ha llevado a cabo un análisis del impacto personal, y, por otro, un análisis del impacto social, esto es, el impacto generado en distintos campos sociales.

A nivel personal, ha tenido un impacto positivo en diferentes aspectos. Por una parte, ha consolidado los conocimientos adquiridos durante la carrera, del mismo modo que se han ampliado al descubrir nuevas tecnologías y formas de programación. También, ha tenido un impacto creativo al tener que plantear diferentes formas llevar a cabo el diseño y la ejecución de una solución a los objetivos iniciales del proyecto.

Por otra parte, también ha reforzado ciertas cualidades, como una mejor organización o el aprendizaje autodidacta de nuevas tecnologías que, seguramente, me serán útiles en un futuro. Sin lugar a duda, el resultado finalmente obtenido, ha hecho que me sienta satisfecho del trabajo realizado. Por último, a nivel personal, este trabajo llevado a cabo también supone el inicio de otros posibles proyectos futuros, enmarcados en el contexto de mi carrera profesional.

En cuanto al impacto social, quedan comprendidos los siguientes campos: empresarial, económico, medioambiental, educativo y cultural. Estos campos interactúan dentro de un marco social, es por ello que se hace referencia al impacto generado por cada uno de ellos dentro de un contexto o marco social, en el cual se articulan.

En primer lugar, si hacemos referencia al impacto social que se articula en los campos empresarial y económico, debemos tener en cuenta las posibilidades que nos ofrece la versatilidad de la aplicación. Esto es, la adaptación de la misma a distintos ámbitos, da lugar a una mayor apertura al mundo empresarial, ya que esta es capaz de adaptarse a distintas empresas con fines heterogéneos (fines educativos, de ocio…), y con ello, obtenemos la posibilidad de adquirir mayores beneficios económicos.

Continuando con el impacto medioambiental, la utilización de las nuevas tecnologías, en este caso, la aplicación aquí desarrollada, podría contribuir a una reducción de la contaminación acústica en espacios abiertos, como, por ejemplo, en la realización de conciertos o eventos de carácter similar que impliquen la alteración de las condiciones normales del ambiente.

De la mano con este impacto positivo a nivel medioambiental, nos encontramos con una repercusión también positiva en el ámbito cultural, ya que, el uso de la aplicación podría suponer un mayor acercamiento a la cultura (tanto en espacios abiertos, como podría ser el caso de un concierto, como en espacios cerrados como pueden ser los teatros).

Por último, en un contexto educativo, esta aplicación podría dar lugar a una mejora de la educación en cuanto al rendimiento académico se refiere. Ofreciendo la incorporación de nuevas tecnologías en el aula, la aplicación podría ser utilizada como una herramienta a través de la cual, los contenidos impartidos por los maestros y maestras, estuviesen al alcance de todos los alumnos y alumnas que componen el aula, mejorando así, las lecciones impartidas tanto a nivel visual como auditivo. Estos aspectos se pueden relacionar con dos de los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 de las Naciones Unidas, que son, por un lado, el objetivo número 4: “Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos” [38], y por otro lado el objetivo número 9: “Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación” [39]

Por tanto, el desarrollo de la aplicación, causaría a nivel social, un impacto notable en los distintos campos sociales aquí mencionados.

Bibliografía

[1] Screen Mirroring an Android Phone. ScreenBeam. [Internet] Disponible en https://www.screenbeam.com/learn-more/wireless-display/screen-mirroring-an-android-phone/#:~:text=Screen%20mirroring%20an%20Android,%20or,with%20a%20room%20of%20people

[2] Colaboradores de los proyectos Wikimedia. (2006, 4 de febrero). Universal Plug and Play - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre. [Internet]. Disponible en https://es.wikipedia.org/wiki/Universal_Plug_and_Play

[3] Colaboradores de los proyectos Wikimedia. Digital Living Network Alliance - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre. [Internet] Disponible en https://es.wikipedia.org/wiki/Digital_Living_Network_Alliance

[4] Docencia presencial curso 2021-22-ETS de Ingenieros Informáticos (UPM). (2021, 1 de octubre). ETS de Ingenieros Informáticos de la Universidad Politécnica de Madrid. [Internet] Disponible en https://www.fi.upm.es/?pagina=2659

[5] Jay, A. (2020, 3 de agosto). What Is A Video Wall & How Does It Work? (& Installation Guide). Mvix Digital Signage. [Internet] Disponible en https://mvixdigitalsignage.com/blog/all-you-need-to-know-about-video-walls/

[6] m3sv. (2018). Plain UPnP - UPnP / DLNA server and browser (Versión 3.0.2). [Aplicación móvil]. Google Play. https://play.google.com/store/apps/details?id=com.m3sv.plainupnp

[7] InShot Inc. (2020). Screen Mirroring - Miracast – Apps (Versión 1.3.0.1). [Aplicación móvil]. Google Play. https://play.google.com/store/apps/details?id=screen.mirroring.screenmirroring

[8] TIAGO FRANCISCO MARTINHO, UNIPESSOAL LDA (2022). Duplicar Pantalla · Móvil en TV. (Versión 20220628)[Aplicación móvil]. App Store. https://apps.apple.com/es/app/duplicar-pantalla-móvil-en-tv/id1468495939

[9] Mobile Operating System Market Share Worldwide (s. f.). Statcounter Global Stats. [Internet] Disponible en https://gs.statcounter.com/os-market-share/mobile/worldwide (accedido el 30 de junio de 2022)

[10] realme C11 2021 - realme (España). (s. f.). realme (Europe) - Dare to Leap. [Internet] Disponible en https://www.realme.com/es/realme-c11-2021

[11] Compra un iPhone SE. (s. f.). Apple (ES). [Internet] Disponible en https://www.apple.com/es/shop/buy-iphone/iphone-se

[12] Download Android Studio and SDK tools | Android Developers. (s. f.). Android Developers. [Internet] Disponible en https://developer.android.com/studio

[13] Velimirovic, A. (2022, 13 de enero). What is UPnP (Universal Plug and Play)?.phoenixNAP Blog. [Internet] Disponible en https://phoenixnap.com/blog/what-is-upnp#:~:text=UPnP%20(Universal%20Plug%20and%20Play)%20is%20a%20networking%20protocol%20that,devices%20on%20the%20same%20network

[14] Colaboradores de los proyectos Wikimedia. (2006, 4 de febrero). Universal Plug and Play - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre. [Internet] Disponible en https://es.wikipedia.org/wiki/Universal_Plug_and_Play

[15] OCF - UPnP Standards & Architecture. (s. f.). Open Connectivity Foundation (OCF). [Internet] Disponible en https://openconnectivity.org/developer/specifications/upnp-resources/upnp/#standards

[16] DLNA Certified® Device Classes - DLNA. (s. f.). Wayback Machine. [Internet] Disponible en https://web.archive.org/web/20101222205822/http:/www.dlna.org/digital_living/devices/

[17] Frequently Asked Questions About DLNA - DLNA. (s. f.). Wayback Machine. [Internet] Disponible en https://web.archive.org/web/20101222213043/http:/www.dlna.org/about_us/faqs/

[18] API de WebRTC - Referencia de la API Web | MDN. MDN Web Docs. [Internet] Disponible en https://developer.mozilla.org/es/docs/Web/API/WebRTC_API

[19] A. Kava. What is WebRTC: The Future of Video Communication. videosdk.live. [Internet] Disponible en https://www.videosdk.live/blog/webrtc

[20] Signaling and video calling - Web APIs | MDN. MDN Web Docs. [Internet] Disponible en https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Signaling_and_video_calling

[21] Configure wireless projection from Android to a Miracast-enabled large screen. Microsoft Support. [Internet] Disponible en https://support.microsoft.com/en-us/office/configure-wireless-projection-from-android-to-a-miracast-enabled-large-screen-17a8b653-d4ae-4aa1-833e-7bf7904aa620

[22] MediaProjection | Android Developers. Android Developers. [Internet] Disponible en https://developer.android.com/reference/android/media/projection/MediaProjection

[23] MediaCodec | Android Developers. Android Developers. [Internet] Disponible en https://developer.android.com/reference/android/media/MediaCodec

[24] Colaboradores de los proyectos Wikimedia. VP8 - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre. [Internet] Disponible en https://es.wikipedia.org/wiki/VP8

[25] Colaboradores de los proyectos Wikimedia. H.264/MPEG-4 AVC - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre. [Internet] Disponible en https://es.wikipedia.org/wiki/H.264/MPEG-4_AVC

[26] 4thlinea. 4th Line. GitHub. [Internet] Disponible en https://github.com/4thline

[27] WebRTC. Android | WebRTC. GitHub. [Internet] Disponible en https://webrtc.github.io/webrtc-org/native-code/android/

[28] nanohttpd. GitHub. [Internet] Disponible en https://github.com/NanoHttpd/nanohttpd

[29] Firebase. Firebase Crashlytics | Firebase Documentation. Firebase. [Internet] Disponible en https://firebase.google.com/docs/crashlytics

[30] Firebase. Google Analytics | Firebase Documentation. Firebase. [Internet] Disponible en https://firebase.google.com/docs/analytics

[31] zxing. ZXing ("Zebra Crossing") barcode scanning library for Java, Android. GitHub. [Internet] Disponible en https://github.com/zxing/zxing

[32] A. Anitua Valluerca. Transmisión de Vídeo y Audio: Protocolos de Streaming. Telefónica Servicios Audiovisuales. [Internet] Disponible en https://www.telefonicaserviciosaudiovisuales.com/articulos-de-divulgacion/protocolos-de-streaming/

[33] offbye. DroidDLNA. GitHub. [Internet] Disponible en https://github.com/offbye/DroidDLNA

[34] KernelCrap. android-dlna. GitHub. [Internet] Disponible en https://github.com/KernelCrap/android-dlna

[35] SIY1121. ScreenCastSample. GitHub. [Internet] Disponible en https://github.com/SIY1121/ScreenCastSample

[36] trishika. DroidUPnP. GitHub. [Internet] Disponible en https://github.com/trishika/DroidUPnP

[36] bbogush. web_screen. GitHub. [Internet] Disponible en https://github.com/bbogush/web_screen

[37]MediaRecorder.AudioSource | Android Developers. Android Developers. [Internet] Disponible en https://developer.android.com/reference/android/media/MediaRecorder.AudioSource#VOICE_COMMUNICATION

[38] Educación - Desarrollo Sostenible. Naciones Unidas. [Internet] Disponible en https://www.un.org/sustainabledevelopment/es/education/

[39] Infraestructura - Desarrollo Sostenible. Naciones Unidas. [Internet] Disponible en https://www.un.org/sustainabledevelopment/es/infrastructure/

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Thu Jun 30 23:35:05 CEST 2022
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)