UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

MJPEG Live Streaming Multidispositivo

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Pablo Martínez Artés

Tutor: José Salvador Oliver Gil

[2016-2017]

Resumen

Somos conscientes de que en los últimos años se ha popularizado la adquisición de dispositivos móviles a un ritmo desorbitado. Pero no solo eso. El consumo tecnológico por parte de los usuarios no se ha limitado al teléfono convencional, sino que ha seguido la estela comercial que las grandes empresas han dejado al alcance del consumidor. Así pues vemos cómo la sociedad actual ha integrado en el uso cotidiano "*tablets*" y demás dispositivos táctiles provistos de aplicaciones de todo tipo: juegos, navegadores, aplicaciones de ofimática etc.

En esta línea de actuación donde se propone la creación de una aplicación para *Android* que responda a otro tipo de necesidades. En este caso queremos desarrollar una herramienta que permita emitir en directo las imágenes capturadas por nuestro Smartphone a través de la cámara. Es decir, ofrecemos al usuario la ayuda para que éste, sin la utilización de equipos externos pueda utilizar su/s smartphones como puntos de emisión en directo para componer un sistema de videovigilancia y gestionarlos a través de una interfaz web.

Para realizar esta aplicación deberemos aprender cómo programar para el sistema operativo *Android* y qué oportunidades nos ofrece. Por esto, al empezar desde cero, documentaremos el proyecto adecuadamente para poder comprender la realización del mismo.

Por tanto, podemos afirmar que la presente memoria pretende informar al lector de todos los pasos realizados para concluir el proyecto de forma satisfactoria, incluyendo una estimación del coste temporal y económico que supondrá su realización y el resultado obtenido mencionando si hemos cumplido con los objetivos.

Por último, comentaremos las conclusiones extraídas tras la realización del proyecto incluyendo un pequeño apartado en el cual comentaremos posibles mejoras de cara a las posibles actualizaciones derivadas a partir de la propia aplicación.

Palabras clave: interfaz, web, cámara, android, videovigilancia, smartphone, aplicación.

Abstract

It's not a secret that we've been observing lately an impressive increase in the number of mobile devices per user, not just cell phones but also tablets and touch phones. As its use grows companies decide to develop software for this kind of devices, both games and useful applications on day-to-day affairs.

In this report we will look at the steps performed to develop a mobile application on devices running the Android operating system. An application focused on everyone who is interested in send the camera smartphone images via local network to form a video surveillance system accessing a web interface to interact with these images.

In order to develop this application we will learn how to program for the Android operating system and what opportunities it offers. So, to start scratch, we try to document this project in a proper way to make it easy to understand.

Therefore, we can say that this document is intended to inform the reader about the steps taken to complete the project successfully, including an estimation of the time and economic costs of its implementation and the result mentioning if we have met the objectives.

Finally, we will quote the conclusions obtained after the project performance, including a small section in which we discuss possible improvements for developing new features derived from the original application.

Keywords : interface, web, camera, android, surveillance system, smartphone, application.

Tabla de contenidos

Tabla de ilustraciones

1. Introducción

Con el crecimiento que ha sufrido el mercado de la tecnología móvil muchas empresas se han sumado al desarrollo para el sistema operativo *Android*. No sólo grandes empresas sino particulares y amantes de la programación también se cuentan en la creación de aplicaciones para acercarse al usuario y ofrecer un servicio que mejore la experiencia de éste con su propio dispositivo.

Desde el pasado diciembre del año 2016 las cifras de aplicaciones en *Google Play Store* indican que existen más de 2.600.000 aplicaciones listas para descargar con un clic y la estadística más reciente perteneciente a junio del año 2017 indica que esta cantidad ascendía a 3.000.000 de aplicaciones en *Google Play Store*, es decir, un incremento del **15.38%** en la cantidad de aplicaciones disponibles en el mercado de Google. Estas cifras por sí solas ya son dignas de mención y por tanto, para cerrar esta visión de crecimiento referenciamos también que las estadísticas muestran que en el pasado año 2016 también supuso un incremento ingente de la cantidad de descargas realizadas en las que teniendo en cuenta a su vez el mercado de Apple, la *App Store* de iOS, se alcanzaba la friolera cantidad de 13 mil millones más con respecto al año 2015 [ANDRO4ALL][STATISTA].

Por otro lado el interés acerca de profundizar y explotar la tecnología de streaming de vídeo no es ya, sólo percibible en España sino en prácticamente todo el mundo. Las plataformas que hoy día utilizan esta tecnología como su núcleo de servicio tales como, *Netflix* o *Twitch.tv* son prueba de que este interés crece día a día.

Muchas son las personas que desearían un control visual en directo sobre ciertas áreas de su casa. Por eso, proponemos una aplicación móvil llamada LiveDroid mediante la cual podremos retransmitir las imágenes capturadas por la cámara del smartphone a través de la red local que posee la propia casa y posteriormente visualizar este *streaming* a través de la interfaz web que también ofrece la aplicación.

En el segundo punto se podrá visualizar los motivos por los que se ha propuesto la realización de este proyecto, explicando su nacimiento y la idea general que tendrá. Se enumerarán los objetivos principales y secundarios que conciernen al proyecto, que serán de obligatorio cumplimiento para garantizar que se está llevando a cabo con éxito.

En el tercer punto se situará al lector en el sector tecnológico en el que se encuentra el proyecto, introduciendo en detalle todas las tecnologías que existen actualmente y que pueden ayudar en el desarrollo del mismo. También se hará una comparativa con el software existente en el mercado relacionado con el streaming de vídeo.

En el cuarto punto, después de haber definido los objetivos del proyecto y presentado el marco tecnológico que lo rodea, se especificará el sistema. En este mismo punto se realizará una planificación temporal, indicando una estimación de la

duración aproximada del proyecto en un diagrama de Gantt. También se realizará una estimación económica donde se calcularán todos los costes asociados.

En el quinto punto se presentarán las etapas necesarias para el desarrollo del sistema, análisis, diseño e implementación. En primer lugar se llevará a cabo un análisis del sistema donde se hará hincapié en los requisitos para comprenderlos mejor. Posteriormente, en la fase de diseño, mediante el lenguaje *UML (Unified Modeling Language)* se crearán distintos tipos de diagramas que ayudarán a representar diferentes aspectos de la aplicación. Los diagramas de clases para representar la estructura y los diagramas de actividades para representar su comportamiento. Por último. en la fase de implementación se mostrarán los aspectos más relevantes de esta última etapa enumerando los pasos necesarios para la puesta en marcha del sistema.

En el sexto punto comprobaremos el correcto funcionamiento de la aplicación desarrollada. Se llevarán a cabo pruebas para realizar un análisis y poder evaluar los resultados obtenidos.

En el séptimo punto, se comentarán las conclusiones obtenidas al finalizar el proyecto y se abrirán nuevas líneas de trabajo futuro o mejoras de la aplicación que pueden servir para la creación de nuevos proyectos.

En el octavo punto, encontraremos las referencias de los datos que vamos exponiendo a lo largo de la memoria.

En el noveno punto, se situarán ciertas clases del proyecto consideradas relevantes y de especial interés.

Por último podrá observarse la bibliografía utilizada durante la realización del proyecto.

2. Motivación y objetivos

2.1. Motivación

La idea de crear este proyecto surgió después de un día de clase en el que se hizo especial hincapié en la reutilización de elementos en informática, se requería de un ejercicio en el que se pudiera distinguir una necesidad que fuera cubierta por la reutilización de dispositivos, y tras una pequeña investigación en mi entorno cercano, descubrí que destacaba la necesidad de vigilar una zona de la casa por diferentes motivos, tales como un hijo pequeño, una mascota o comida preparándose en la cocina, sumado a que no existen equipos de vigilancia de bajo coste, y que las aplicaciones actuales de streaming suelen apuntan hacia altos niveles de API en

Android como por ejemplo la versión 4 *KitKat* , no permiten reutilizar smartphones más antiguos ya que en estos esas aplicaciones no funcionarian.

Otro motivo no menos importante que propició esta iniciativa fue el saber que en los hogares normalmente se encuentran móviles en desuso y que estos pueden convertirse en recursos como cámaras, reduciendo enormemente el coste, y aumentando considerablemente los puntos de emisión posibles.

Otro de los motivos fundamentales de la elección de esta aplicación como proyecto ha sido la oportunidad de aprender a desarrollar para dispositivos móviles y profundizar en el desarrollo de tecnologías web como HTML CSS y JavaScript, ya que, su aprendizaje fortalece el currículo y abre puertas para el futuro profesional de un ingeniero informático.

2.2. Objetivos

Una vez que ha quedado claro por qué hemos tomado la decisión de realizar este proyecto veamos cuál es la idea u objetivo principal del mismo:

"*Desarrollar una aplicación móvil sobre el sistema operativo Android que nos sirva de herramienta para facilitar a los usuarios la utilización de smartphones en desuso o el propio como cámaras de vigilancia, una cámara ip, y poder interactuar con estos flujos de imágenes a través de una interfaz web*"

Para poder conseguirlo es necesario cumplir con una serie de objetivos secundarios que citamos a continuación:

- Uno de los objetivos es el enfoque pragmático de la aplicación. Es importante que el usuario pueda interactuar con la aplicación de forma cómoda de manera que le resulte intuitivo.
- De igual forma que el primer objetivo mencionado, necesitamos que la interfaz web o página sea amigable, cómoda e intuitiva para el usuario ya que en esta residirá el mayor número de interacciones.
- Otro de los objetivos de este proyecto es adquirir los conocimientos necesarios de la programación para dispositivos móviles.
- Evitar bloqueos de la aplicación por anomalías de la misma.
- Profundizar en el aprendizaje de las tecnologías web.

3. Estado del arte

3.1. Introducción al estado del arte

En este punto pretendemos realizar un estudio y análisis sobre la situación del mercado alrededor de esta aplicación en un ámbito informático. Por tanto se estudiarán las diversas alternativas disponibles en la actualidad que pueden servir para desarrollar el proyecto.

Inicialmente se analizará de forma global el software existente en el mercado relacionado con el objetivo de este proyecto, el cual es el streaming de video.

Posteriormente se estudiarán los distintos sistemas operativos para dispositivos móviles que existen en el mercado indicando ventajas y contras de cada uno y detallando los motivos por el cual se decidirá sobre uno u otro.

Por último estableceremos un resumen de las tecnologías escogidas para cada parte del sistema que vamos a diseñar.

3.2. Estado del arte tecnológico

Una vez ya se sabe para qué sistema operativo se va a desarrollar el proyecto y en que versión, se va a hacer una revisión del aspecto tecnológico en posibles aplicaciones existentes en el mercado que persigan el mismo objetivo. Por eso a continuación se detallarán algunas de las opciones con las que cuenta el usuario.

3.2.1. Camera Streamer - IP Camera

Camera Streamer - IP Camera es una aplicación desarrollada por la empresa *RT Software Studio* en el 2015 con el objetivo de convertir el móvil del usuario en una cámara ip. Actualmente se puede encontrar esta aplicación en su versión 2.1.3 para Android.

Las características de esta aplicación son las siguientes:

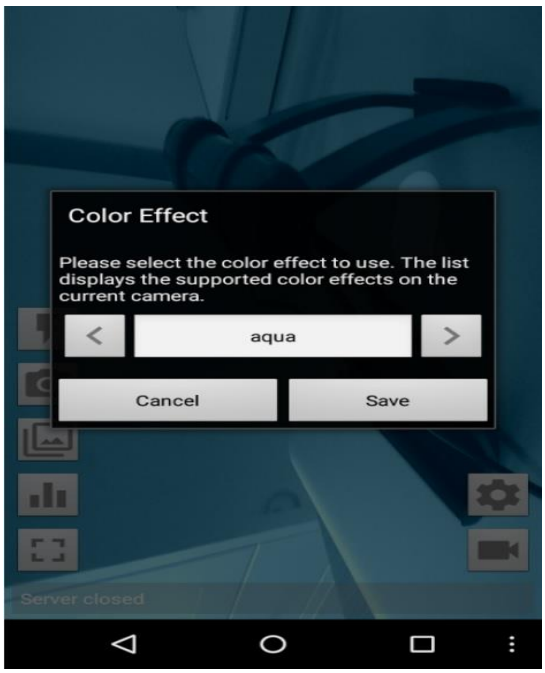
- Es completamente gratuita
- Posee una interfaz sencilla y amigable para el usuario
- Se disponen de diversas opciones para la configuración del stream
- Acceso simple al stream de la aplicación

Por otro lado remarcar algunos aspectos negativos de la aplicación:

- Las opciones de cámara seleccionables dependen del dispositivo en cuestión, es decir, no siempre tendremos las mismas opciones.
- Nulas opciones para realizar interacciones con el stream que ofrece la aplicación, solo podemos visualizarlo.

En resumen *Camera Streamer - IP Camera* es una aplicación eficaz en su objetivo ya que cumple las expectativas que ofrece, incluyendo elementos opcionales para configurar el stream antes de poder visualizarlo. El objetivo es parecido, pero este proyecto va orientado para aquellas personas que deseen concentrar un número de cámaras determinado en una misma interfaz web, y dándoles la opción de poder realizar ciertas interacciones con la misma.

A continuación se mostrarán algunas capturas de esta herramienta:

Actividad del stream	*Efectos de color para el stream*
	
Ilustración 1	*Ilustración 2*

Efectos de rotación del stream	*Configuración de los puertos*

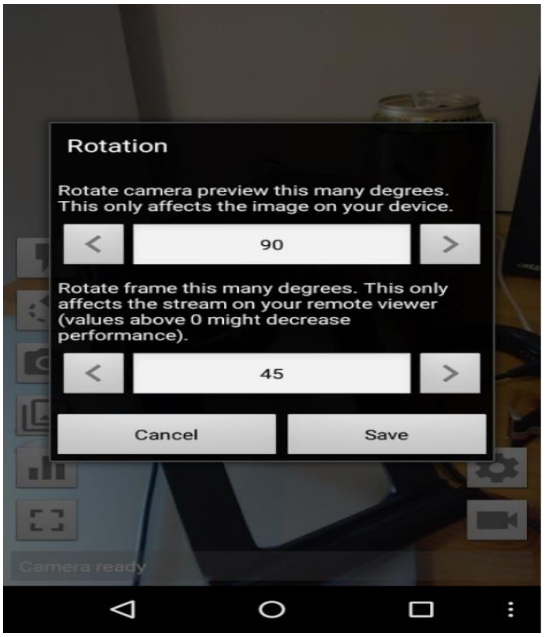
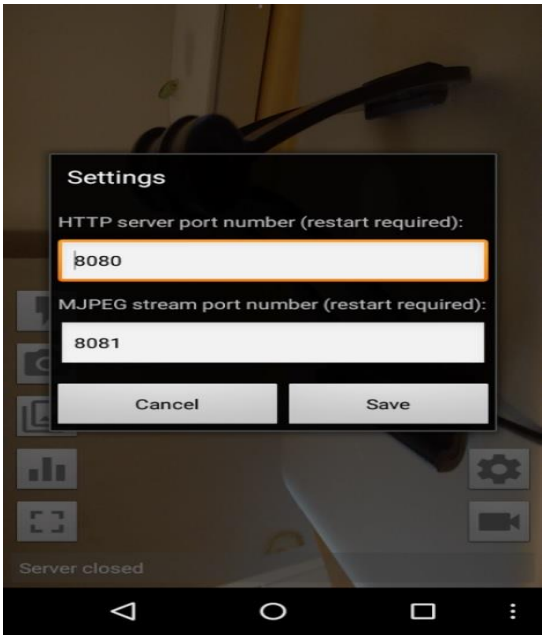

	

Ilustración 3	*Ilustración 4*

3.2.2. AtHome Camera

AtHome Camera es una aplicación desarrollada por el autor *Ichano* en Julio del 2017 y actualmente se encuentra en su versión 3.6.0. Orientada para el sistema operativo Android, aunque también dispone una versión para el sistema operativo iOS y cuyo objetivo es ofrecer al usuario una plataforma para registrar los dispositivos en los que instale esta aplicación y convertirlos en cámaras ip, además contiene diversas funcionalidades encaminadas a la seguridad del hogar.

Entre las características que destacan en esta aplicación citamos las siguientes:

- Funciona bajo red WiFi tanto como por redes 3G/4G.
- Comunicaciones encriptadas.
- Funcionalidad de "visión nocturna" para mejorar la calidad de imagen con mayor oscuridad en el ambiente.
- Detección de movimiento y avisos en forma de notificación push.

Por otro lado, en cuanto a los aspectos negativos de esta aplicación:

- Limitación del número de cámaras a visualizar en la propia interfaz de cuatro.
- Algunas opciones no pueden utilizarse si no compras una cámara ip.
- Imposible reutilización de smartphones antiguos debido a que el API level apunta a Android 4 KitKat como versión mínima del sistema operativo.

En resumen *AtHome Camera* es una aplicación muy completa y funcional, es algo compleja para aquellas personas que no sean habituales en el uso del smartphone ya que emplea tecnologías como el almacenamiento en la nube, también puede tener una complejidad demasiado grande para aquellas personas que solamente quieran una visualización simple y rápida ya que requiere de un registro para introducir tus datos y cámaras disponibles.

A continuación se mostrarán algunas capturas:

Adición de cámaras	*Opciones de almacenamiento en la nube*

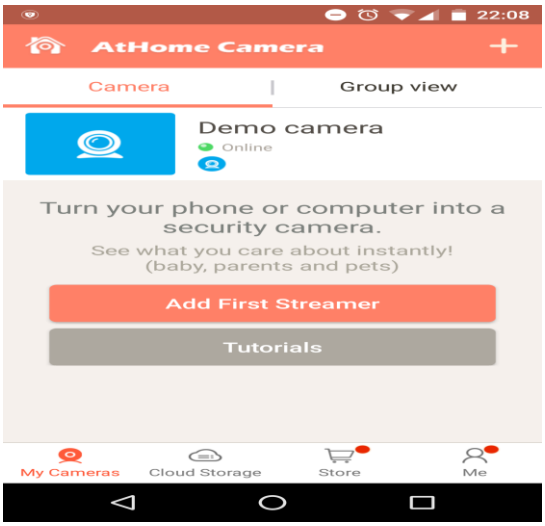
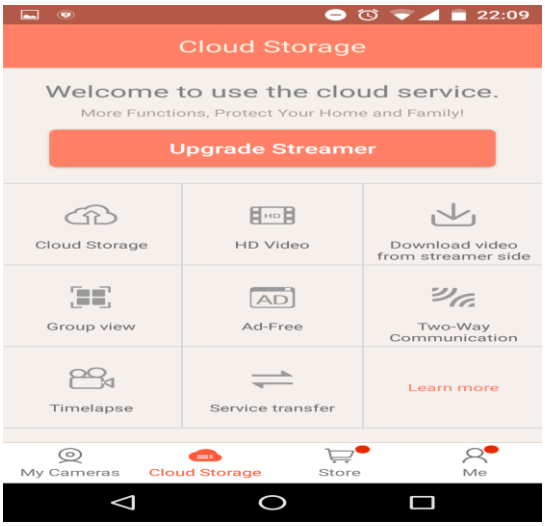

	

Ilustración 5	*Ilustración 6*

Mercado interno de la aplicación	*Opciones de la cuenta del usuario*

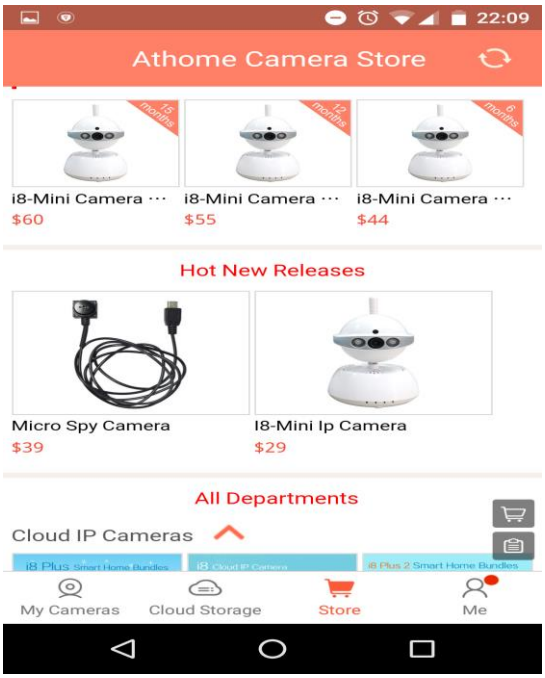
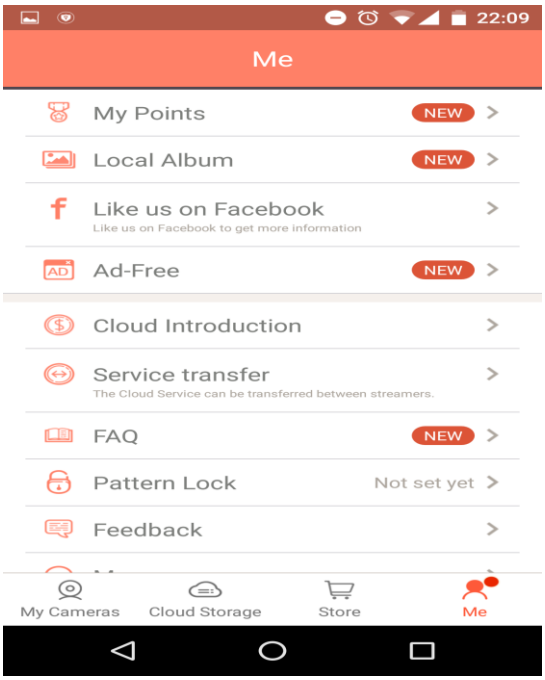

	

Ilustración 7	*Ilustración 8*

3.2.3. CameraFi Live

CameraFi Live es una aplicación por la empresa *Vault Micro Inc.* en el año 2017 actualmente se encuentra en su versión 1.8.8.0801 y se encuentra disponible para la plataforma de Android. Su objetivo se centra en la transmisión de la cámara del móvil o una cámara USB a conocidas plataformas como YouTube y Facebook.

Esta aplicación también se encuentra disponible para el sistema operativo iOS de Apple.

Esta aplicación es gratuita con compras integradas dentro de la aplicación. Las características que más destacan de esta aplicación son las siguientes:

- Soporta transmisiones de video de alta calidad.
- Posee funciones de zoom.
- Transmisión de diversas fuentes de vídeo simultáneamente.
- Superposición de texto e imágenes en las transmisiones.

Los aspectos negativas que se pueden encontrar en esta aplicación son:

- Transmisión orientada a las plataformas Youtube y Facebook.
- Obligación de poseer cuentas en dichas plataformas y configurarlas correctamente para recibir la transmisión.
- Interfaz de usuario algo compleja para los usuarios inexpertos.

En conclusión, *CameraFi Live* es una aplicación muy útil siempre y cuando nuestra intención sea llevar el streaming hacia las redes sociales, aunque esta aplicación no

se adecúa al objetivo del proyecto, es utilizada por muchos usuarios y sus herramientas son variadas e interesantes.

A continuación se podrán ver varias capturas:

Configuración del aspecto del stream y opciones

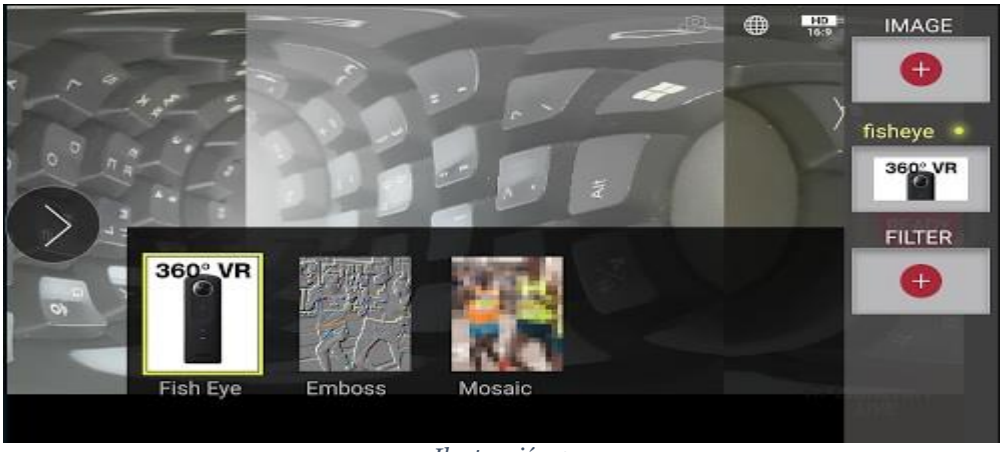


Ilustración 9

Selección del servidor de stream

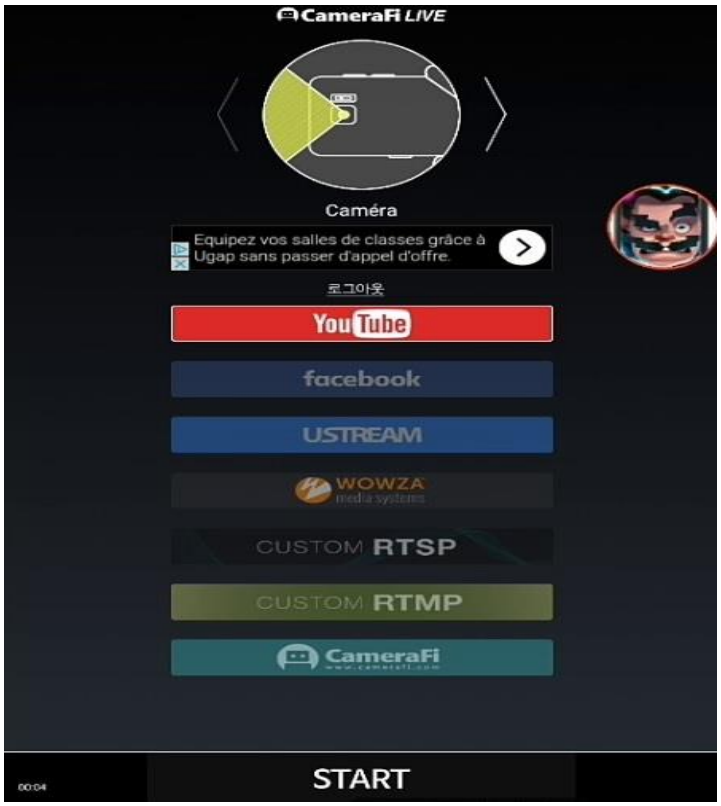


Ilustración 10

3.2.4. Resumen del estado del arte tecnológico

A continuación mostraremos en la tabla una comparativa de las aplicaciones que hemos analizado existentes en el mercado a modo de resumen:

	Camera Streamer	AtHome Camera	CameraFi Live	LiveDroid
Tipo de usuario	Medio	Medio	Medio	Medio
Plataformas	Android	Android, iOS	Android, iOS	Android
Versión Android	+2.3.3	+4.0	+4.3	+2.3.3
Compras Integradas	✗	✓	✓	✗
Configuración del streaming	✓	✓	✓	✓
Tecnología de streaming	MJPEG	RTMP,RSTP	RTMP,RSTP	MJPEG
Interacción con el streaming	✗	✓	✓	✓
Dependencias	Ninguna	Cámara ip para algunas opciones	Cuenta Facebook, cuenta YouTube, cuenta Ustream	Ninguna
Interfaz web	✗	✓	✗	✓

Tabla 1

3.3. Sistemas Operativos en dispositivos móviles

3.3.1 Android

Ilustración 11

"*Android es un conjunto de software de código abierto para dispositivos móviles y su correspondiente proyecto de código abierto liderado por Google.*"[SOURCEANDROID]

El sistema operativo *Android*, inicialmente, es un sistema operativo pensado para teléfonos móviles así como para *iOS, Symbian, Blackberry OS* y *Windows Phone*. Pero la característica que lo diferencia del resto es que el núcleo de este sistema operativo está basado en *Linux* que es libre, gratuito y multiplataforma.

Android permite desarrollar aplicaciones en *Dalvik*, una variación de *Java* y proporciona todas las interfaces necesarias para acceder de forma sencilla a las funciones del teléfono a la hora de desarrollar aplicaciones (GPS, llamadas, etc.).

Esta sencillez y la posibilidad de desarrollar o programar mediante herramientas o entornos de programación gratuitos, hacen que existan muchas aplicaciones.

Pero aun así, una de las mejores características de *Android* es que es un sistema de código abierto y libre por tanto no requiere de licencias ni software de pago. Esto es muy favorable para las empresas porque minimizan costes.

Actualmente la última versión disponible en el mercado es la versión *Android 7.0 (Nougat)* y trae consigo muchas mejoras importantes respecto de sus antecesoras como son:

- Optimización de la memoria RAM para aplicaciones en segundo plano.
- Incorporación de Java 8.
- Soporte multiventana nativo.
- Mejoras en el uso de la batería con su nuevo modo DOZE que detecta el movimiento del dispositivo siendo transportado.

A continuación mostramos la distribución de versiones en Enero de 2016[ANDRO4ALLDIST]:

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.5%
4.1.x	Jelly Bean	16	8.8%
4.2.x		17	11.7%
4.3		18	3.4%
4.4	KitKat	19	35.5%
5.0	Lollipop	21	17.0%
5.1		22	17.1%
6.0	Marshmallow	23	1.2%

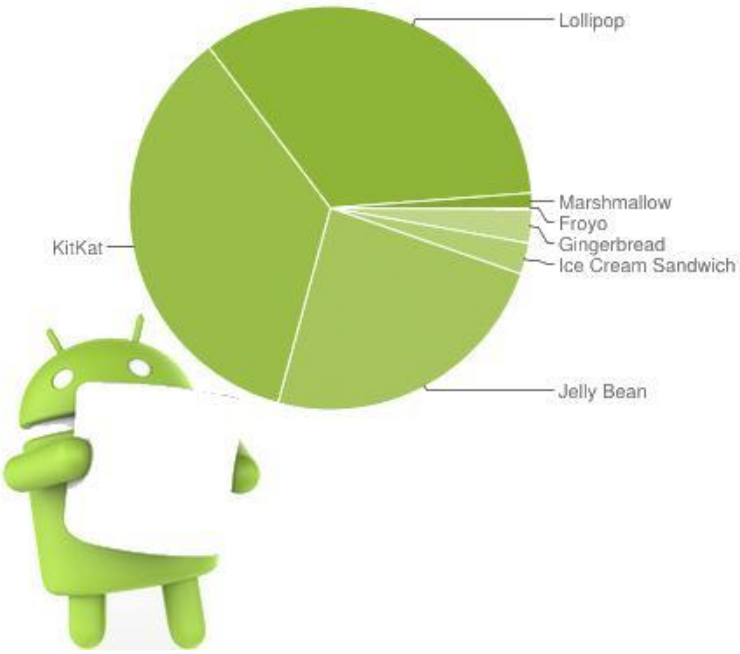


Ilustración 12

Como podemos observar la mayoría de los dispositivos según esta distribución está repartida entre la versión 4.4 (*KitKat*) y 5.0 (*Lollipop*), no obstante nuestro objetivo es cubrir sobretodo los dispositivos que han ido cayendo en desuso y que en anteriores años se encontraban repartidos entre la versión 2.3.3 (*Gingerbread*) y la versión 4.0.3 (*Ice Cream Sandwich*) por lo tanto nuestra versión elegida como Minimus Sdk sería 2.3.3 (*Gingerbread*) con un nivel de api 9-10 que es su correspondiente, con ello podemos asegurarnos que no sólo funcionaría en estos dispositivos sino también en los posteriores.

El entorno de desarrollo que se utiliza de forma oficial para desarrollar aplicaciones es *Ecplise* pero actualmente existe el IDE *Android Studio* que se encuentra en su versión 2.3.3, y que supone una gran mejora para el desarrollo debido a que su enfoque se centra exclusivamente en el desarrollo en Android.

En cualquier caso *Android* nos ofrece unas características inmejorables, pero hay que tener en cuenta ciertos inconvenientes por ejemplo el diseño de la interfaz mediante layouts de la aplicación ya que no es intuitivo y es costoso. Otro posible inconveniente es la desaprobación o mejora de funciones de una versión posterior, ya que un método, librería o función que funcione en una versión anterior de Android puede haber sido mejorada en posteriores y por tanto inutilizable.

3.3.2. iOS

Ilustración 13

iOS (*Iphone Operating System*) es un sistema operativo creado por la empresa *Apple Inc.* En sus orígenes fue desarrollado para el dispositivo móvil *iPhone*, pero acabó por ser incluido en otros dispositivos como *iPod Touch, iPad y Apple TV*.

En primer lugar nos encontramos con que es un sistema operativo cerrado, es decir, que no se puede instalar en otro hardware que no pertenezca a la misma empresa y siguiendo en la misma línea tampoco se puede hacer uso de aplicaciones desarrolladas por otros programadores mientras que Apple no las apruebe.

La última versión disponible estable es *iOS 10*, y que en la propia web de *Apple* la describen de la siguiente manera:

"Con iOS 10, todo lo que antes te gustaba ahora te gustará más, porque la actualización más importante hasta la fecha viene llena de novedades. Sé original expresando lo que sientes en Mensajes. Disfruta aún más del viaje con el nuevo diseño de Mapas. Revive tus mejores momentos como no te imaginabas en Fotos. Y pon a Siri a trabajar, ahora en más apps que antes. ¿Te parece mucho? Es solo el principio."

Aun así, en el caso de que este sistema sea el sistema operativo elegido, nos centraríamos en la versión que más dispositivos englobe puesto que uno de los objetivos más importantes del proyecto es la reutilización de dispositivos.

Su interfaz está basada en el concepto de manipulación directa, por tanto es intuitiva y cuenta con la posibilidad de usar gestos multitáctiles. Tiene una respuesta inmediata y es muy fluida,

iOS es una derivación del Mac OS X (Sistema Operativo de Apple para ordenadores), y este se basa en Darwin BSD que a su vez está basado en *Unix* con lo que proporciona una mayor estabilidad y rendimiento. Por tanto podemos deducir que iOS es un sistema operativo de tipo *Unix*.

Para desarrollar aplicaciones para este sistema operativo son necesarios lenguajes de programación como C, C++ u Objective-C. Estos son muy conocidos entre

programadores pero para poder proceder se necesita de un kit de desarrollo o *SDK* que se llama *Xcode* y se ejecuta sobre *Mac SO X.*

Por otro lado para poder crear una aplicación es necesaria una licencia de desarrollo por la que hay que pagar, impidiendo minimizar costes y enfocando este punto como el principal inconveniente.

Aun así, teniendo en cuenta toda esta información la elección de este sistema operativo no se descarta porque si, es cierto que encarece el coste del proyecto pero también es verdad que existe una gran cantidad de documentación en constante actualización en función a la evolución del sistema operativo.

3.3.3. BlackBerry OS

Ilustración 14

BlackBerry OS es un sistema operativo para dispositivos móviles BlackBerry desarrollado por la empresa *RIM* (*Research In Motion*). Este sistema operativo estaba orientado al ámbito profesional dado que disponía de herramientas útiles como correo electrónico y agenda.

La última versión conocida y estable de BlackBerry es la versión 10 pero esta está disponible para muy pocos modelos, por tanto esto puede ser un inconveniente a tener en cuenta ya que el objetivo del es llegar al máximo público posible tratando de reutilizar incluso dispositivos en desuso. Por lo que, teniendo en cuenta que no se concentra tampoco un número excesivo de usuarios en este sistema operativo parece inconsistente la elección del mismo.

BlackBerry OS como *iOS* es de código cerrado puesto que solo puede ser ejecutado en terminales de esta misma empresa pero proporciona documentación y APIs para los programadores que quieran desarrollar por cuenta propia.

Otra característica importante es que mediante la sincronización con herramientas como *Lotus Note, Novell GroupWise y Microsoft Exchange Server* así como la agenda y el correo electrónico mencionado anteriormente se refuerza aún más el hecho de que esté orientado al ámbito profesional.

Este sistema operativo está escrito en *Java y C++* aunque su micro núcleo está basado únicamente en *Java.* Es un sistema que soporta multitarea y dispositivos especiales como trackpad, trackballs, trackwell y touchscreen pueden interactuar con este.

3.3.4. Windows Phone

Ilustración 15

Windows Phone es el sistema operativo para dispositivos móviles de *Microsoft*. Es la evolución directa de *Windows Mobile*. Está orientado a un mercado generalista no como su versión anterior es por esto que muchas herramientas orientadas al ámbito profesional que existían en *Windows Mobile* no están disponibles en esta nueva versión.

Actualmente la última distribución de este sistema operativo se transformó en lo que hoy día entendemos como *Windows 10 Mobile*, el sucesor de *Windows Phone (8.1)*. Este sistema está ya diseñado para smartphones y tabletas.

Hoy en día cuenta con un SDK en su versión 10 que proporciona encabezados, librerías, metadatos y herramientas necesarios para poder compilar aplicaciones, en su conjunto el IDE oficial para desarrollar aplicaciones sería el *Visual Studio 2017* para poder sacar el óptimo provecho de sus herramientas y las API más recientemente incluidas.

Un punto muy a favor de este sistema operativo es que dentro de esta versión las aplicaciones pueden ser creadas para la toda la plataforma de windows de forma universal.

3.3.5. Symbian

Ilustración 16

Symbian OS es el sistema operativo en propiedad de *Nokia* y su objetivo principal era poder competir con sistemas operativos como *Windows Mobile* y en la actualidad

con *Android* de *Google*, *iOs* de *Apple*, *Windows Phone* de *Microsoft* y *Blackberry OS* de *BlackBerry*. No obstante, en Octubre de 2011 se comunicó que su soporte llegaría hasta el año 2016 ya que no fué capaz de ser un competidor frente a los sistemas operativos de última generación anteriormente mencionados.

La última versión conocida y estable de este sistema operativo es *Symbian Belle FP2* y está caracterizado por ser robusto. Fue concebido para ahorrar recursos de memoria y de procesador logrando un rendimiento útil de la batería superior.

Symbian OS permite que el desarrollo de sus aplicaciones sea escrito en el lenguaje de programación *C++* ya que posee un micro núcleo basado en este mismo lenguaje. Pero esto no quita que no se puedan utilizar otros como *Java* o *Visual Basic* para desarrollar aplicaciones. Lo que quiere decir que para un programador que desee involucrarse en un proyecto para este sistema operativo no necesitará aprender ningún lenguaje de programación específico. Pero aun así, posee distintas interfaces de usuario en función al dispositivo móvil y esto dificulta la tarea del programador.

En el 2010 un estudio de *Gartner* situaba en el año 2014 a *Symbian OS* en la cabeza de los sistemas operativos más utilizados.

Forecast: Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units)

OS	2009	2010	2011	2014
Symbian	80,876.3	107,662.4	141,278.6	264,351.8
Market Share (%)	46.9	40.1	34.2	30.2
Android	6,798.4	47,462.1	91,937.7	259,306.4
Market Share (%)	3.9	17.7	22.2	29.6
Research In Motion	34,346.8	46,922.9	62,198.2	102,579.5
Market Share (%)	19.9	17.5	15.0	11.7
iOS	24,889.8	41,461.8	70,740.0	130,393.0
Market Share (%)	14.4	15.4	17.1	14.9
Windows Phone	15,031.1	12,686.5	21,308.8	34,490.2
Market Share (%)	8.7	4.7	5.2	3.9
Other Operating Systems	10,431.9	12,588.1	26,017.3	84,452.9
Market Share (%)	6.1	4.7	6.3	9.6
Total Market	**172,374.3**	**268,783.7**	**413,480.5**	**875,573.8**

Source: Gartner (August 2010)

Ilustración 17

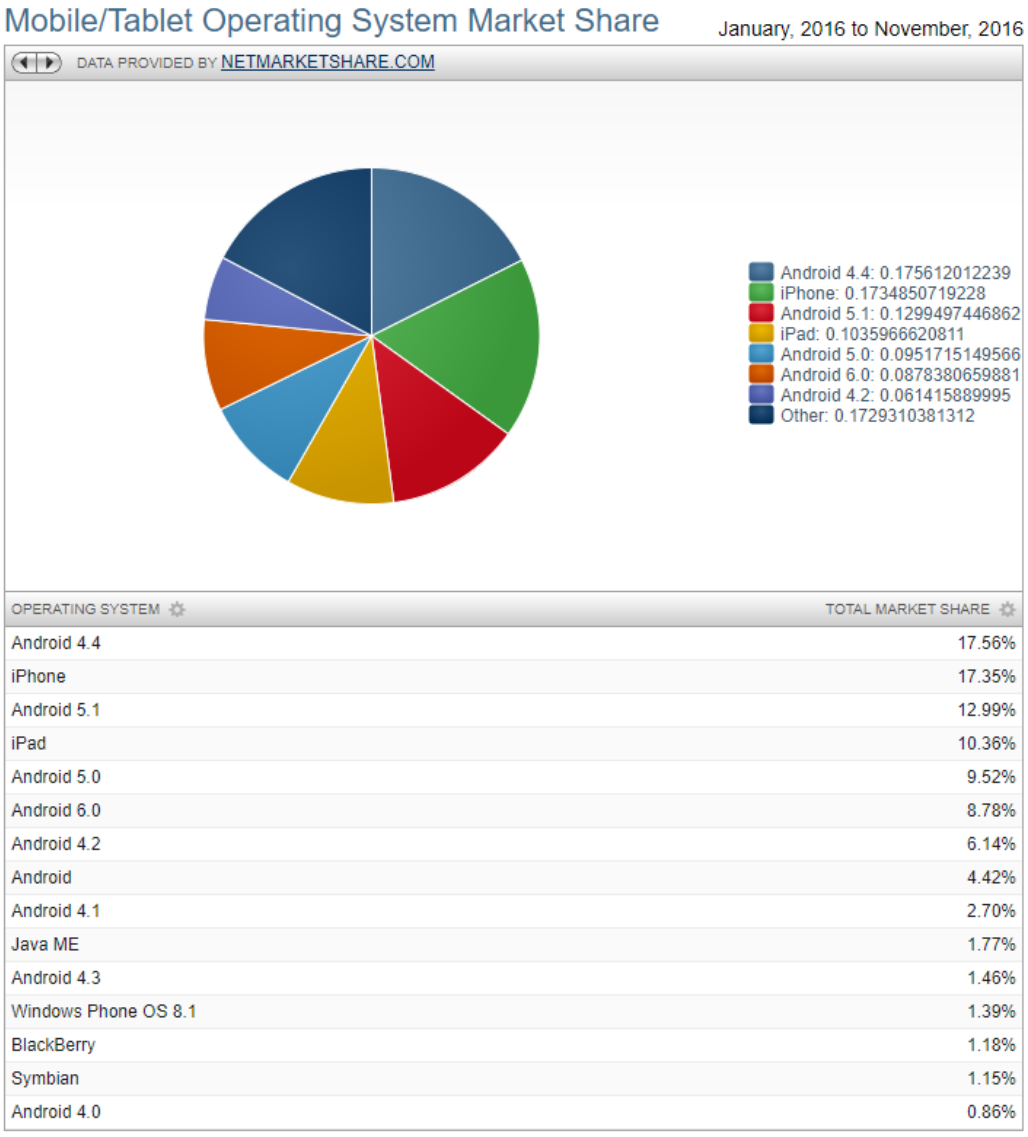
Debido a la muerte de este sistema operativo en el año 2016 al no obtener más soporte por parte de sus desarrolladores, a pesar de que se convirtió en un sistema de desarrollo abierto, hace que esta opción prácticamente se descarte para su elección.

3.3.6. Elección de sistema operativo

LLegados a este punto ya se puede tomar una decisión en cuanto al sistema operativo para el cual se desarrollará el proyecto.

Se han estudiado todos los sistemas operativos expuestos y se ha llegado a la conclusión de que el más conveniente es *Android*, pero ¿Por qué motivos?

Android es el sistema operativo que cuenta con más usuarios actualmente como se puede ver en la siguiente figura mostrada a continuación según el año 2016:



OPERATING SYSTEM	TOTAL MARKET SHARE
Android 4.4	17.56%
iPhone	17.35%
Android 5.1	12.99%
iPad	10.36%
Android 5.0	9.52%
Android 6.0	8.78%
Android 4.2	6.14%
Android	4.42%
Android 4.1	2.70%
Java ME	1.77%
Android 4.3	1.46%
Windows Phone OS 8.1	1.39%
BlackBerry	1.18%
Symbian	1.15%
Android 4.0	0.86%

Ilustración 18

Tal y como se puede observar, *Android* está muy por encima del resto ocupando un 64.43% del total de la cuota de mercado respecto al resto de sistemas operativos, lo que consigue que dejemos fuera a *Windows Phone, BlackBerry* y *Symbian* puesto que al poseer una escasa clientela no interesa desarrollar para ellos ya que se busca llegar al máximo público posible y en el que podamos encontrar mayor número de

dispositivos relegados por desuso y antigüedad, plasmado en el 4.42% de cuota de mercado que tiene la línea *Android* que señala las versiones anterior a *Android 4.0*.

Otro de los motivos por los que se escoge *Android*, es la facilidad de encontrar un dispositivo en desuso que posea el consumidor y que porte el citado sistema operativo, ya que los terminales con dicho sistema operativo han sido siempre más económicos en el mercado ofreciendo, eso sí, menores prestaciones. Al contrario de lo que ocurre con *iOS* donde los terminales al ser de una mayor calidad, significaba un mayor gasto económico que gran parte de los consumidores no podían permitirse.

Y el último factor, que no menos importante, es la popularidad y facilidades que ofrece a la hora de desarrollar una aplicación puesto que nos ofrece un entorno de desarrollo muy completo de forma gratuita, y una gran cantidad de documentación que hará que el desarrollo sea llevadero, simple e intuitivo.

3.4. Arquitectura de transmisión

Un aspecto muy importante para el proyecto será que arquitectura vamos a seguir para realizar la transmisión de vídeo, que protocolos participan y de que forma trabajan, para de esta forma, poder realizar una elección correcta acorde a los objetivos y funcionamiento que proponemos para este proyecto. Entre las siguientes arquitecturas que veremos a continuación, observaremos las diferencias entre una arquitectura basada en los protocolos RTSP/RTP/RTCP y una arquitectura basada en HTTP Streaming.

3.4.1. Arquitectura de transmisión basada en RTSP/RTP/RTCP

Como indica en el encabezado de la sección esta arquitectura está basada en el protocolo de transporte *RTP (Real-Time Transport Protocol)*. Es un protocolo diseñado para la entrega de audio y vídeo a través de redes IP. Este protocolo es utilizado ampliamente en sistemas de comunicación y de entretenimiento. Fué diseñado por el equipo de desarrollo *IETF (Internet Engineering Task Force)* siendo publicado como estándar por primera vez en 1996 como la *RCF 1889* [RCF1889] y posteriormente actualizado en 2003 en la *RFC 3550* [RCF3550].

RTP va acompañado del protocolo *RTCP (RTP Control Protocol)* diseñado para proporcionar mecanismos de control utilizados para informar sobre la calidad en la distribución de los datos, es decir, se basa en transmisiones periódicas de paquetes de control para otorgar un seguimiento sobre el flujo ofrecido por *RTP*.

Como hemos mencionado *RTP* permite la administración de flujos multimedia sobre IP y funciona sobre el protocolo *UDP* ya que mediante este gana una mayor velocidad con respecto al protocolo TCP.

A continuación mostramos el formato de la cabecera *RTP*:

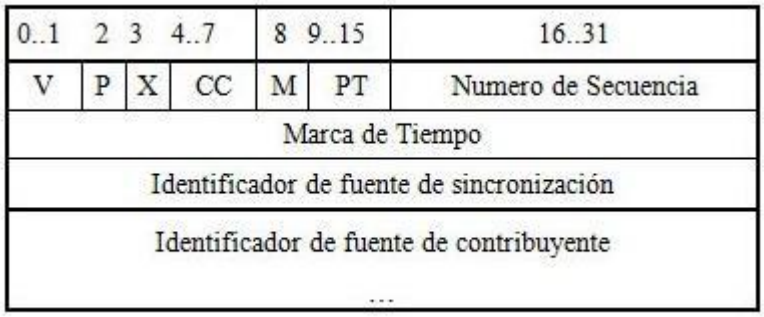


Ilustración 19

Ahora introducimos el protocolo *RTSP (Real-Time Streaming Protocol)* que trabaja sobre la capa de aplicación y fue desarrollado por *RealNetWorks, Netscape Communications y Columbia University*, fué publicado en el RFC 2326. Este protocolo es utilizado para el envío de datos que trabajan en tiempo real y es capaz de trabajar junto al anterior protocolo mencionado *RTP*. Su principal trabajo es el de proporcionar un ambiente para el envío de datos de tiempo real bajo demanda como el audio y vídeo. El origen del flujo de datos puede ser tanto en directo cómo almacenado y es capaz de trabajar sobre los protocolos *UDP, UDP Multicast y TCP*.

RTSP pretende dar el mismo soporte para audio y vídeo como HTTP lo hace para los textos y gráficos, cada flujo de datos está identificado por una *URL RTSP*. No obstante, *RTSP* y *HTTP* difieren ya que *HTTP* es un protocolo sin estado mientras que *RTSP* debe mantener los estados de las sesiones que realiza para poder satisfacer peticiones y flujos, adicionalmente, *HTTP* es un protocolo asimétrico ya que el cliente realiza las peticiones y el servidor las responde mientras que en *RTSP* se pueden realizar peticiones desde ambos extremos. Los servicios y operaciones soportados por *RTSP* son los siguientes:

- OPTIONS: El cliente o el servidor le comunican a la otra parte que opciones aceptan.
- DESCRIBE: El cliente consigue una descripción de un contenido identificado por una URL RTSP.
- ANNOUNCE: Actualiza la descripción en tiempo real.
- SETUP: El cliente le pregunta al servidor donde conseguir los datos.
- PLAY: El cliente le pide al servidor que comience a mandarle los flujos configurados en SETUP.
- PAUSE: El cliente detiene el envío sin liberar los recursos en el servidor. 4
- TEARDOWN: El cliente solicita al servidor que detenga el envío del flujo especificado y libere todos los recursos asociados a él.
- GET_PARAMETER: Consigue el valor de un parámetro de una presentación o flujo.
- SET_PARAMETER: Establece el valor de un parámetro de una presentación o flujo.
- REDIRECT: El servidor informa al cliente de que debe conectarse al servidor indicado en la cabecera.
- RECORD: El cliente comienza a grabar datos de la presentación.

A continuación mostramos una posible topología en una imagen:

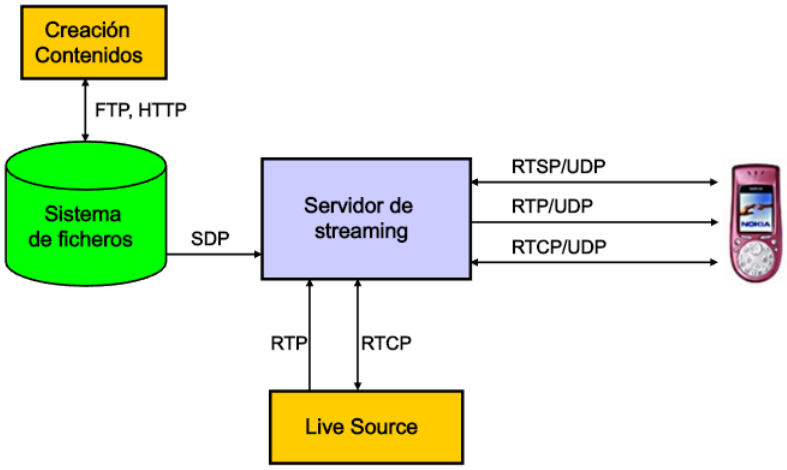


Ilustración 20

3.4.2 Arquitectura HTTP Streaming

HTTP Streaming es una técnica utilizada para enviar constantes actualizaciones a un cliente web. Esta se mantiene abierta como una conexión persistente entre el cliente web y el servidor web para que cuando el servidor tenga nueva información, pueda enviarla hacia el cliente. Es por tanto realmente una conexión persistente puesto que la misma solo terminaría debido a problemas en la red, con la salida del usuario del cliente web o con el cierre de la aplicación.

Dentro de esta arquitectura existen soluciones estandarizadas como *HDS (HTTP Dynamic Streaming)* creada por *Adobe* o *HLS (HTTP Live Streaming)* que viene de manos de Apple.

Ambas arquitecturas desarrollan la tecnología *Adaptative BitRate Video* que consiste en la variabilidad de la calidad del vídeo consumido por el cliente según el estado de su conexión a internet, por lo que a mayor ancho de banda posee el cliente mejor puede llegar a ser la calidad del vídeo consumido, al contrario, si su ancho de banda es peor la calidad del vídeo también será peor.

En estas soluciones el servidor cuando el vídeo está codificado, múltiples ficheros son creados para diferentes anchos de banda en diferentes resoluciones, con lo cual una vez se va detectando la cantidad de ancho de banda que posee el cliente se le sirven estos ficheros segmentados, siempre en función de su velocidad.

A continuación mostramos una imagen a modo de ejemplo:

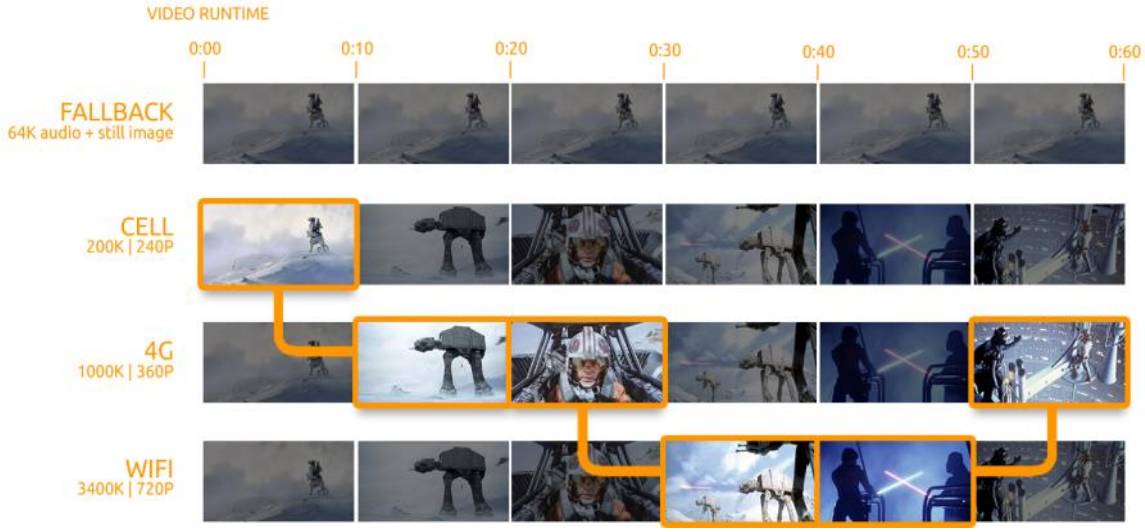


Ilustración 21

Estas soluciones se basan en la aplicación de tres elementos:

- El **componente servidor**: Responsable de recoger el flujo de datos de entrada que ofrecen los medios, como una cámara, y su trabajo es codificarlos digitalmente y encapsularlos en un formato adecuado para la entrega.
- El **componente de distribución**: Consta de Servidores web convencionales, los cuales, son responsables de aceptar las solicitudes entrantes de los clientes y entregar los datos preparador anteriormente por el servidor.
- El **software cliente**: Este, es responsable de determinar el medio adecuado para solicitar la descarga de los recursos y más tarde reensamblarlos para que estos puedan ser representados en un stream continuo.

3.4.3. Selección de la arquitectura

Como hemos visto anteriormente, las diferentes arquitecturas tienen características a favor y en contra que las tornan más o menos adecuadas para el proyecto que se está planteando, en esta sección consideraremos cuál es la más conveniente para con el proyecto.

Se ha podido observar que en la arquitectura basada en los protocolos *RTP/RTCP y RTSP* existe un control de la conexión con diversos métodos que ayudan al manejo de la misma incluyendo una herramienta de control de estado como lo es el protocolo *RTCP*. Por otra parte *HTTP* sabemos que es un protocolo que carece de estado por lo que no tendríamos un control sobre la conexión como lo tiene la primera arquitectura comentada.

Sabemos que la arquitectura basada en *RTP/RTCP y RTSP* basa su conexión generalmente por el protocolo *UDP* lo que significa que la entrega de los paquetes no está asegurada, mientras que el protocolo *HTTP* trabaja sobre *TCP/IP* que asegura que todos los paquetes serán entregados, retransmitiéndolos si es necesario.

La arquitectura basada en *HTTP* al trabajar sobre *TCP/IP* por lo que la lógica para implementar multicast es más compleja que para la arquitectura basada en *RTP/RTCP y RTSP* ya que esta trabaja sobre *UDP.*

Por otro lado, la arquitectura basada en *RTP/RTCP y RTSP* conlleva una mayor complejidad para el desarrollo de la lógica de negocio y es necesario un mayor uso de especificaciones, como la CPU, memoria, batería...

Sin embargo hay un gran inconveniente en el uso de la arquitectura basada en *RTP/RTCP y RTSP* y es que no tiene soporte para la versión que queremos desarrollar en *Android (2.3.3)* por lo que para su uso se debería aumentar el nivel de API del proyecto hasta la correspondiente a la versión a *3.1* ya que es a partir de esta cuando se comienza a dar soporte a esta tecnología, y esto supone dejar fuera de alcance un gran número de terminales potencialmente útiles para el uso de esta aplicación.

Por lo tanto, teniendo en cuenta todos los datos, debido a su mayor sencillez, su menor consumo de recursos, y su posibilidad de implementación en bajas versiones de *Android* elegiremos como arquitectura, la arquitectura basada en *HTTP Streaming*.

3.5. Codificación del vídeo

Para poder transmitir las imágenes es necesario emplear códecs de vídeo que permitan comprimir y descomprimir vídeo digital. El problema que abordan los códecs es que la información del vídeo en sí es demasiado grande como para que un ordenador normal pueda manejarlo, por ello, los códecs se definen como algoritmos de compresión y descompresión en tiempo real y su finalidad es lograr un almacenamiento mucho menos al de la información del vídeo.

Para ello estudiaremos los códecs H.264 y M-JPEG (Motion-JPEG) ya que ambos conforman códecs de fácil uso además de no presentar un alto coste en requisitos para su empleo.

3.5.1. H.264

Desarrollado por el *ITU-T Video Coding Experts Group (VCEG) y el ISO/IEC Moving Picture Experts Group (MPEG)* su intención principal fue crear un estándar capaz de proporcionar una buena calidad de imagen con tasas binarias notablemente inferiores a los estándares previos añadiendo el hecho de no incrementar la complejidad de su diseño.

Este códec estuvo enfocado hacia el vídeo de baja calidad para videoconferencia y aplicaciones por internet, pero no mostraba salida para ámbitos más profesionales donde se exigen resoluciones más elevadas, por ello se trabajó por conseguir las

extensiones de gama de fidelidad *FRExt* [FRExt] para poder tener mayor cabida en los ambientes profesionales en los que antes no podía ser utilizado.

En sí mismo H.264 no supone una tecnología decisiva con respecto a las normas de codificación de vídeo anteriores. La pequeña diferencia se halla en la menor cuantía de información necesaria que se necesita para almacenar en los vídeos codificados por este mismo códec.

3.5.2. MJPEG (Motion-JPEG)

Ésta compresión de vídeo fue originalmente desarrollada por *Multimedia PC Applications* y se basa en la compresión de cada fotograma del vídeo o campo entrelazado, por separado como una imagen JPEG, este códec es usualmente utilizado en dispositivos portátiles como pueden ser las cámaras digitales.

Es usado frecuentemente para videocámaras basadas en IP, a través de flujos o streams HTTP, utilizando para ello el tipo de contenido *MIME multipart/x-mixed-replace*. Lo que nos permite separar en diferentes e individuales respuestas HTTP en un marcador especificado. La gran mayoría de navegadores en sus últimas versiones cuentan con soporte nativo para poder visualizar estos flujos.

3.5.3 Selección del códec de vídeo

Una vez vistos los posibles códecs hemos de elegir el más adecuado para el proyecto y para ello realizaremos una comparación entre ellos.

Por una parte veamos que ventajas e inconvenientes conseguimos con el códec H.264:

- Reduce el consumo de ancho de banda y de almacenamiento considerablemente.
- Bitrate de vídeo adaptativo según el ancho de banda del usuario.
- Compresión de audio y vídeo.
- Mayor complejidad de configuración.

Por otra parte veamos que ventajas e inconvenientes conseguimos utilizando M-JPEG:

- Consistente gran calidad de imagen (aunque dependerá de la/s resolución/es de la cámara).
- Mayor robustez, sin un fotograma es perdido, este no afectará al vídeo.
- Sin sonido.
- Menor complejidad de uso.

Debido a la naturaleza del proyecto, un mayor uso de ancho de banda no supondrá un factor decisivo ya que emplearemos la propia red LAN. Como planteamos la

videovigilancia como objetivo, el sonido tampoco es decisivo aunque es un punto a favor del códec *H.264*, no obstante, aumenta notoriamente la complejidad del proyecto ya que las cámaras de un dispositivo móvil tan solo ofrecen como datos imágenes por lo que habría que realizar un acceso al micrófono del dispositivo y configurar una sincronización entre el sonido y las imágenes de la cámara.

Ya que la calidad del flujo dependerá en el caso de *MJPEG* de la resolución de la cámara que se utilice, nos encontramos frente a un arma de doble filo, debido a que una cámara con una pobre resolución nos ofrecerá peor calidad de imagen, sin embargo las mínimas resoluciones disponibles en los dispositivos que entren a partir de la versión *2.3.3 de Android* pueden ofrecer resultados aceptables para los flujos resultantes.

Por lo tanto por su flexibilidad y facilidad, además que nos permitirá transmitir a la resolución de la propia cámara y que nos da mayor seguridad ya que la pérdida de un fotograma no supone grandes riesgos para el flujo, elegiremos *MJPEG (Motion-JPEG)* como códec de compresión y descompresión de vídeo.

3.6. Servidor web

Para poder ofrecer los servicios que propone el proyecto debemos analizar cuál es el servidor que mejor se acomoda a las necesidades del proyecto y considerar dónde ubicar el mismo, ya que, aunque lo convencional es situarlo como un elemento externo, también existe la opción de instanciarlo de forma local.

3.6.1 Servidor Web Convencional y Servidor Web Personal

La diferencia entre los conceptos de un servidor web convencional y un servidor web personal radica básicamente en que el servidor web convencional es normalmente inicializado en una máquina pensada para hacer la función especializada de servidor ya que cuenta con características dedicadas y propias para cumplir adecuadamente con las demandas que entran, por otra parte, el servidor web personal es inicializado en la propia máquina personal para que ésta haga su papel de servidor, lo que puede suponer una limitación de los recursos necesarios como el ancho de banda o la velocidad de procesamiento.

De forma natural los servidores web personales son de manera general utilizados como una herramienta de desarrollo, sin embargo, el contexto de este proyecto acepta el uso de dicho concepto, ya que el alcance de uso del proyecto para el usuario será su propia red LAN lo que implica que en la gran mayoría de casos el usuario se encontrará en su domicilio o en un entorno cercano como puede serlo su oficina.

Tenemos una serie de factores que apoyan el uso del servidor web personal, es decir, ejecutarlo en la propia máquina, que dentro de este proyecto equivaldrá a ejecutarlo en el propio terminal móvil, a continuación mostramos dichos factores:

- Sencilla ejecución del servidor por parte del usuario.
- Inexistente necesidad de configuración por parte del usuario.
- Costes reducidos utilizando soluciones open source.
- Supresión de la necesidad de otra máquina para la ejecución del servidor y por lo tanto directamente una reducción de los costes.

3.6.2 Alternativas para el servidor web

A continuación detallamos las opciones para elegir el servidor que implementaremos para el proyecto.

- **APACHE**

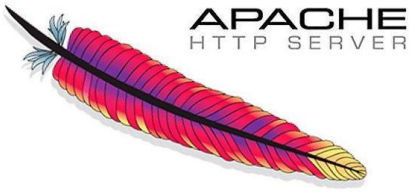


Ilustración 22

Apache es un servidor web *HTTP* de código abierto desarrollado por *Apache Software Foundation*. La última versión estable y conocida de este servidor es la versión 2.4.27.

Apache está programado en *C* y es utilizado para subir páginas web tanto estáticas como dinámicas a la *WWW(World Wide Web)* y suele ser empleado por los programadores web como servidor de pruebas y testeo del código que se está desarrollando.

Entre sus características principales destacamos que es multiplataforma, que tiene una alta aceptación en la red y que es modular.

- **XAMPP**

Ilustración 23

XAMPP (X Apache MySQL PHP Perl) es un servidor independiente de plataforma desarrollado por *Apache Friends*. Su última versión estable y conocida es la versión 7.1.7

Este servidor consiste en una base de datos *MySQL*, un servidor web *Apache* e intérpretes de lenguaje *Perl* y *PHP*.

Este entorno es muy popular en el desarrollo con *PHP*, es completamente gratuito, sencillo de instalar y configurar. Se puede utilizar bajo *Windows, Mac OS y Linux*.

Otra de sus características principales es que incorpora módulos como *OpenSSL* y *phpMyAdmin*.

- **WAMP**

Ilustración 24

WAMP (Windows Apache MySQL PHP) es una versión análoga a *XAMPP* y también incluye en el pack el servidor web *Apache* en su versión 2.2e, *MySQL* en la versión 5.5.24 y *PHP* en su versión 5.4.3.

Este entorno como *XAMPP* también es gratuito, sencillo de instalar y configurar. Pero solo es compatible para sistemas operativos *Windows*.

También incorpora características como *Xdebug, phpMyadmin* y *SQLBudy*.

- **AndServer basado en Jetty**

Ilustración 25

AndServer es un servidor diseñado para *Android* basado en *Jetty* y desarrollado por su autor yanzhenjie, que es el nombre del usuario utilizado en su cuenta de GitHub. Este servidor es capaz de servir tanto páginas web dinámicas como páginas web estáticas, es posible desarrollar en él un API HTTP, posee la capacidad de subida de ficheros al terminal móvil y robustez frente a alta concurrencia.

Como anteriormente hemos comentado este servidor está basado en *Jetty* que se define como un servidor *HTTP* absolutamente basado en *Java* y en un contenedor de *Servlets* escrito en *Java*. Este proyecto software libre se publica bajo la licencia Apache 2.0.

Jetty comenzó en 1995 alojandose en el servidor *MortBay* desarrollándose las versiones 1.x y 2.x hasta el año 2000. Entre 2000 y 2005 cambió su alojamiento hacia *sourceforge.net* mientras las versiones 3.x, 4.x y 5.x. En enero de 2009 los principales componentes de *Jetty* fueron trasladados a *Eclipse.org* y pasándose a *Eclipse Foundation*.

Los principales objetivos de *Jetty* son dirigidos a la creación de un servidor web sencillo, eficiente, empotrable y pluggable. El reducido tamaño de *Jetty* lo hace apropiado para ofrecer servicios web en una aplicación *Java.*

3.7. Tipos de aplicaciones en Android

Al desarrollar aplicaciones para *Android* existen diversas alternativas por las que podemos optar. Todas ellas poseen una serie características y limitaciones, en las que el enfoque técnico varía.

La elección de un tipo u otro condicionará a la aplicación sobre el diseño visual de la misma y la interacción de esta.

Existen básicamente tres tipos de aplicaciones móviles dependiendo del enfoque que necesitemos para responder a las necesidades del proyecto:

- Aplicaciones móviles nativas.
- Aplicaciones móviles web.
- Aplicaciones móviles híbridas.

3.7.1. Aplicaciones móviles nativas

Las aplicaciones móviles nativas son aquellas que se desarrollan de forma específica para un determinado sistema operativo, en este caso *Android*, con el conjunto de herramientas otorgado por dicho sistema llamado *SDK (Software Development Kit)*, de esta forma *Android* es programado en *Java*, en el caso de *Windows Phone* en *.Net* y en el caso de *iOS* en *Objective-C*.

A continuación se muestra un conjunto de ventajas e inconvenientes que ofrece este tipo de aplicación móvil:

Ventajas:

- Acceso completo al dispositivo, es decir, nos permite utilizar todas las características hardware del terminal, como pueden ser sus sensores (acelerómetro, giróscopo, GPS, entre otros sensores), cámara, micrófono o el servicio de notificaciones.
- No requieren de conexión a internet para funcionar, y su integración con el dispositivo hace que estas ofrezcan una experiencia más fluida para el usuario.
- Gran facilidad de distribución y actualización ya que estas aplicaciones pueden estar disponibles en las plataformas de distribución del sistema operativo, en el caso de *Android*, la *Google Play Store*.

Inconvenientes:

- El código cliente no es reutilizable entre las diferentes plataformas, ya que estamos usando el propio *SDK* del sistema operativo.
- Tienden a tener un coste mayor de desarrollo.
- Es necesario el uso de diferentes herramientas de desarrollo, ya que para cada plataforma suelen usarse diferentes herramientas de programación o *SDK*.

3.7.2. Aplicaciones móviles web

Las aplicaciones móviles web son desarrolladas con lenguajes muy conocidos por los programadores, como lo son *HTML, Javascript y css*. Este tipo de aplicaciones son ejecutadas dentro del propio navegador web del terminal a través de una *URL* por lo cual no es necesario instalarlas y el contenido de la aplicación es adaptado a la pantalla adquiriendo un aspecto de navegación APP.

Las siguientes características e inconvenientes son ofrecidas por este tipo de aplicación móvil:

Ventajas:

- El código cliente es reutilizable para múltiples plataformas. Con lo cual puede ser distribuida entre las diferentes plataformas de distribución.
- Acceso a parte del hardware del dispositivo.
- Instalación nativa pero construida con *Javascript, HTML y CSS.*
- Conlleva un proceso de desarrollo más sencillo y más económico.

Inconvenientes:

- Pese a que conserva parte de acceso al hardware del dispositivo, este es muy limitado.
- Requiere de conexión a internet.
- El tiempo de respuesta es mayor que el de una aplicación nativa.
- Ya que este tipo de aplicaciones no son publicadas en una plataforma de distribución requieren más esfuerzo para darse a conocer.

3.7.3. Aplicaciones móviles híbridas

Las aplicaciones móviles híbridas son una combinación de las dos anteriores. De esta forma podría decirse que recogen las mejores características de cada una dellas. Estas aplicaciones son desarrolladas con los lenguajes utilizados en las aplicaciones móviles web, es decir, *HTML ,* CSS y *Javascript* lo que permite su uso en diferentes plataformas. También son capaces de acceder a gran parte de las características hardware del dispositivo. Y su principal ventaja es que a pesar de los lenguajes utilizados en ella, este tipo de aplicaciones son capaces de ser distribuidas en las app store de los diferentes sistemas operativos.

Como indicamos abajo, mostramos un resumen de ventajas e inconvenientes ofrecidos por este tipo de aplicaciones móviles:

Ventajas:

- Acceso a gran parte de las características hardware del dispositivo.
- Mismo código base para múltiples plataformas.
- Instalación nativa pero construida con lenguajes web como *HTML, CSS* y *Javascript*.
- Es posible distribuirla en diferentes plataformas de distribución.

Inconvenientes:

- Experiencia del usuario más propia de la aplicación móvil web que de una aplicación móvil nativa.
- El diseño visual no siempre está relacionado con el sistema operativo en el que se muestra.

3.7.4. Selección del tipo de aplicación móvil

Ahora ya conocemos los diferentes tipos de aplicaciones android que podemos desarrollar y cuales son las ventajas e inconvenientes de cada una, por lo que procederemos a hacer la mejor selección del tipo para el proyecto que se propone.

Debido a los objetivos del proyecto el tipo de aplicaciones móviles web nos queda prácticamente descartado debido a que como hemos visto, tiene un pobre acceso a

los componentes hardware del dispositivo móvil, y de entre ellos la cámara de la cual tendríamos un difícil y limitado acceso.

Las opciones que nos restan son la construcción de una aplicación nativa o una aplicación híbrida. Ya que otro de los objetivos del proyecto es alcanzar el mayor número de terminales, por lo general, en desuso y que recaen sobre versiones de android muy inferiores a las actuales, las aplicaciones híbridas nos perjudicarían debido a que los navegadores que interpretan los lenguajes web de los terminales más antiguos, nos arrebatarían la libertad de programar con amplitud en estos lenguajes, debido al abandono del soporte a los navegadores de estos terminales.

Por otra parte buscamos también como objetivo el aprendizaje de la programación dentro del sistema operativo android, por lo cual, la selección de las aplicaciones móviles nativas resulta la opción más conveniente para acometerlo.

Por descarte y conveniencia, el tipo de aplicación elegido será el de la aplicación móvil nativa.

3.8. Resumen de las elecciones tomadas

A continuación se realizará un resumen de todas las elecciones que han sido tomadas a lo largo de este punto.

Se desarrollará la aplicación para el sistema operativo *Android* a partir de la versión 2.3.3, en *Java*, mediante el IDE *Android Studio* en su versión 2.3.3.

Se utilizará la codificación *MJPEG* para la compresión del vídeo y se transmitirá a través de la arquitectura de *HTTP Streaming*.

Será utilizado el servidor web *AndServer* para servir la interfaz web del proyecto, ya que este mismo podemos empotrarlo en la propia aplicación que vamos a realizar.

Y utilizaremos el tipo de desarrollo para una aplicación móvil nativa, para realizar un aprendizaje sobre el *SDK* correspondiente al sistema operativo *Android* y poder alcanzar de manera más eficientes los componentes hardware del dispositivo.

4. Planificación y especificación

LLegados a este punto, el problema ya ha sido expuesto y a lo largo de este capítulo se realizará un análisis en detalle de los requisitos y la propuesta de solución planteada. Una vez realizado esto se analizarán los costes temporales y económicos previstos para finalizar el proyecto con éxito incluyendo un diagrama de Gantt para organizar las tareas. Por último se realizará un estudio de la viabilidad del proyecto.

4.1 Análisis de requisitos

A continuación se detallará cada uno de los requisitos que debe cumplir el proyecto. Estos requisitos serán las funcionalidades de nuestro sistema.

Por tanto enumeraremos los requisitos del proyecto:

- La aplicación deberá dar la opción al usuario de transmitir las imágenes del dispositivo mediante un flujo de datos a través de la red de área local.
- La aplicación deberá dar la opción al usuario de configurar el puerto de salida del flujo de datos.
- La aplicación deberá dar la opción al usuario de configurar de elegir entre ambas cámaras del dispositivo, si este posee varias.
- La aplicación deberá dar la opción al usuario de activar o desactivar el componente de luz flash del dispositivo.
- La aplicación deberá dar la opción al usuario de configurar la resolución de las imágenes transmitidas a través del flujo de datos.
- La aplicación deberá dar la opción al usuario de configurar la calidad de las imágenes transmitidas a través del flujo de datos.
- La aplicación debe dar la opción al usuario de activar o desactivar el servidor web integrado desde la propia aplicación.
- La aplicación debe ofrecer al usuario la ruta de acceso al flujo de datos, es decir su dirección IP más el puerto configurado, y al servidor web una vez este sea activado.
- El servidor web de la aplicación debe ofrecer al usuario una interfaz web para poder interaccionar con los flujos de datos.
- La interfaz web debe permitir al usuario la creación de diferentes contenedores para poder visualizar los flujos de datos que desee incluir, añadiendoles a estos las diferentes direcciones que poseerán los terminales que transmiten sus imágenes.
- La interfaz web debe proporcionar la capacidad al usuario de renombrar los contenedores que visualizarán los flujos de datos.
- La interfaz web debe proporcionar la capacidad al usuario de hacer capturas de imágenes sobre el flujo de datos visualizado en los contenedores y guardarlas temporalmente.
- La interfaz web debe proporcionar la capacidad al usuario de consultar las capturas que ha ido realizando sobre los flujos de datos.

- La interfaz web debe proporcionar la capacidad al usuario de eliminar todos o un solo contenedor que visualicen los flujos de datos.
- La interfaz web debe proporcionar al usuario un registro de las acciones que va realizando en la propia interfaz web.
- La interfaz web debe proporcionar la capacidad al usuario de eliminar todo el registro de acciones si así lo desea.
- La interfaz web debe ofrecer una herramienta al usuario para el cálculo de diferencias entre imágenes.
- La interfaz web debe ofrecer una herramienta al usuario para la edición de imágenes mediante filtros y su posterior guardado.
- La interfaz web debe proporcionar la capacidad al usuario de realizar una búsqueda sobre otros dispositivos que tengan la aplicación instalada y estén transmitiendo.
- La interfaz web debe visualizar avisos en forma de notificaciones cuando el usuario inicie acciones o si incurre en acciones que no puede realizar por algún motivo.
- La interfaz web debe ser de diseño *responsive* y deberá ser sencilla e intuitiva para mejorar la experiencia del usuario.
- El dispositivo móvil que contendrá la aplicación deberá ser un smartphone o tablet con una versión igual o superior a la 2.3.3 del sistema operativo *Android* instalada.

4.2. Especificación del sistema

Como hemos podido observar en el punto 4.1 se han citado los requisitos y necesidades que se requieren para el proyecto. Por tanto a continuación se realizará una propuesta de solución adecuada para el sistema.

Haciendo síntesis, el sistema que se solicita debe poder ejecutarse en smartphones o tablets siempre que dispongan de una cámara, y debe permitir el uso de todas las herramientas, así como la transmisión de imágenes de la cámara del dispositivo por la red de área local utilizándose el propio dispositivo como servidor del flujo de datos, además de ser capaz de inicializar servidores web incrustados para servir al usuario una interfaz web para la interacción con dichos flujos de datos y la utilización de un servicio a través de un *API HTTP*. Para realizar este sistema, será necesario realizar dos partes, la primera consistirá en el desarrollo de la aplicación android y la segunda parte será el desarrollo de la interfaz web a través de la cual el usuario poseerá las herramientas para interactuar con los flujos de datos.

En cuanto a la primera parte, la aplicación para móvil, se desarrollará para el sistema operativo *Android* en su versión 2.3.3 *(Gingerbread)* o superior. Deberá tener acceso a una red de área local para poder transmitir las imágenes del terminal y poder servir la interfaz web al usuario.

Por otro lado la segunda parte, correspondiente a la interfaz web, será diseñada aplicando los lenguajes de programación web, estos son, *HTML,* CSS y *Javascript*. También se hará uso de diversas librerías existentes para facilitar el desarrollo de la

funcionalidad de la interfaz web. Por una parte utilizaremos el framework de *bootstrap* para simplificar el diseño responsive de la interfaz web, siendo utilizada en su versión 4. En cuanto al resto de librerías se corresponderán con funcionalidades extendidas del lenguaje de programación *Javascript* de entre las cuales destacamos *jQuery, fabric, noty* y *clipboard*. Esta interfaz web será desarrollada para el navegador web Google Chrome ya que es el navegador que actualmente domina el mercado dada su ingente cantidad de usuarios, posee los últimos estándares de *HTML5,* por lo que es posible utilizar sus herramientas más avanzadas. Se realizan sobre él constantes actualizaciones, sus herramientas de desarrollo son avanzadas y extremadamente útiles para los desarrolladores web e incluimos su rapidez y fluidez a la hora de servir páginas web gracias a su motor de renderizado webKit y su ligero diseño interno.

4.3. Planificación y estimación de costes

Una vez establecida la especificación, para poder evaluar los costes que supondrán el proyecto, es necesario elaborar una tabla con las distintas tareas que se llevarán a cabo a los largo del mismo.

4.3.1. Coste temporal

		Valores			Total días ponderado
Tarea	Descripción	**O**	**P**	**B**	
T1	INICIO	**9**	**24**	**15**	**15,5**
T1.1	Consideración de los requisitos del sistema	6	14	10	10
T1.2	Especificación del sistema	3	10	5	5,5
T2	ESTUDIO DOCUMENTAL PREVIO	**11**	**50**	**21**	**24,16**
T2.1	Estudio de sistemas operativos para dispositivos móviles	3	11	4	5
T2.2	Estudio de tecnologías actuales	4	25	10	11,5
T2.3	Estudio sobre arquitectura de transmisión	3	9	5	5,33
T2.4	Búsqueda y análisis de aplicaciones similares existentes	1	5	2	2,33
T3	ANÁLISIS DEL SISTEMA	**13**	**30**	**18**	**19,15**
T3.1	Análisis de requisitos	3	5	3	3,33
T3.2	Análisis de los casos de uso	2	8	4	4,33
T3.3	Análisis de la interfaz de la aplicación	3	7	3	3,66
T3.4	Análisis de la interfaz web	5	10	8	7,83
T4	DISEÑO DEL SISTEMA	**14**	**36**	**22**	**22,98**
T4.1	Diseño de casos de uso	2	8	4	4,33
T4.2	Diseño de la arquitectura	3	6	4	4,16
T4.3	Diseño de la interfaz de la aplicación móvil	3	9	5	5,33

T4.4	Diseño de la interfaz web	6	13	9	9,16
T5	IMPLEMENTACIÓN DEL SISTEMA	**43**	**97**	**71**	**70,65**
T5.1	Instalación de herramientas y software de desarrollo	1	2	1	1,16
T5.2	Implementación de la interfaz web	22	50	38	37,33
T5.3	Implementación de la aplicación móvil	20	45	32	32,16
T6	PRUEBAS DEL SISTEMA	**5**	**12**	**8**	**8,16**
T6.1	Pruebas funcionales de la aplicación móvil	2	5	4	3,83
T6.2	Pruebas de rendimiento de la aplicación móvil	3	7	4	4,33
T7	DOCUMENTACIÓN FINAL	**20**	**40**	**30**	**30**
	TOTAL	**115**	**289**	**155**	**170,66**

Para estimar el coste temporal de este proyecto se utilizará la técnica de la estimación con tres valores o *PERT*, que consiste en identificar tres posibles valores, el valor más optimista, el pesimista y el más probable. Ésta técnica es usualmente utilizada para escenarios inciertos.

Teniendo en cuenta esta información, se seguirá la siguiente fórmula para realizar el cálculo del tiempo estimado:

Tiempo = (Valor Optimista + 4(Valor Probable) + Valor Pesimista) / 6*

Para realizar una mejor abreviatura en la tabla consideraremos como:

O: El valor optimista.

P: El valor pesimista.

B: El valor probable.

Una vez elaborada y definida la lista de tareas con su correspondiente estimación temporal, y teniendo en cuenta que solo existe un recurso humano para todo el proyecto, conlleva que no se puedan paralelizar tareas, pasaremos a definir lo que es un diagrama de Gantt y a continuación plasmaremos el diagrama teniendo en cuenta la estimación temporal citada anteriormente *(Añadimos ilustración explicativa en la siguiente página)*.

- Diagrama de Gantt

"*Un diagrama de Gantt es una herramienta para la gestión de proyectos que nos brinda una representación gráfica de la información relacionada con un cronograma. Consiste en un diagrama de barras donde las actividades del cronograma o componentes de la estructura de desglose del trabajo se enumerar de la forma descendente en el lado izquierdo del diagrama, las fechas aparecen a lo largo de la*

parte superior, y la duración de las actividades se muestra como barras horizontales ordenadas por fecha" .[PMBOKGANTT]

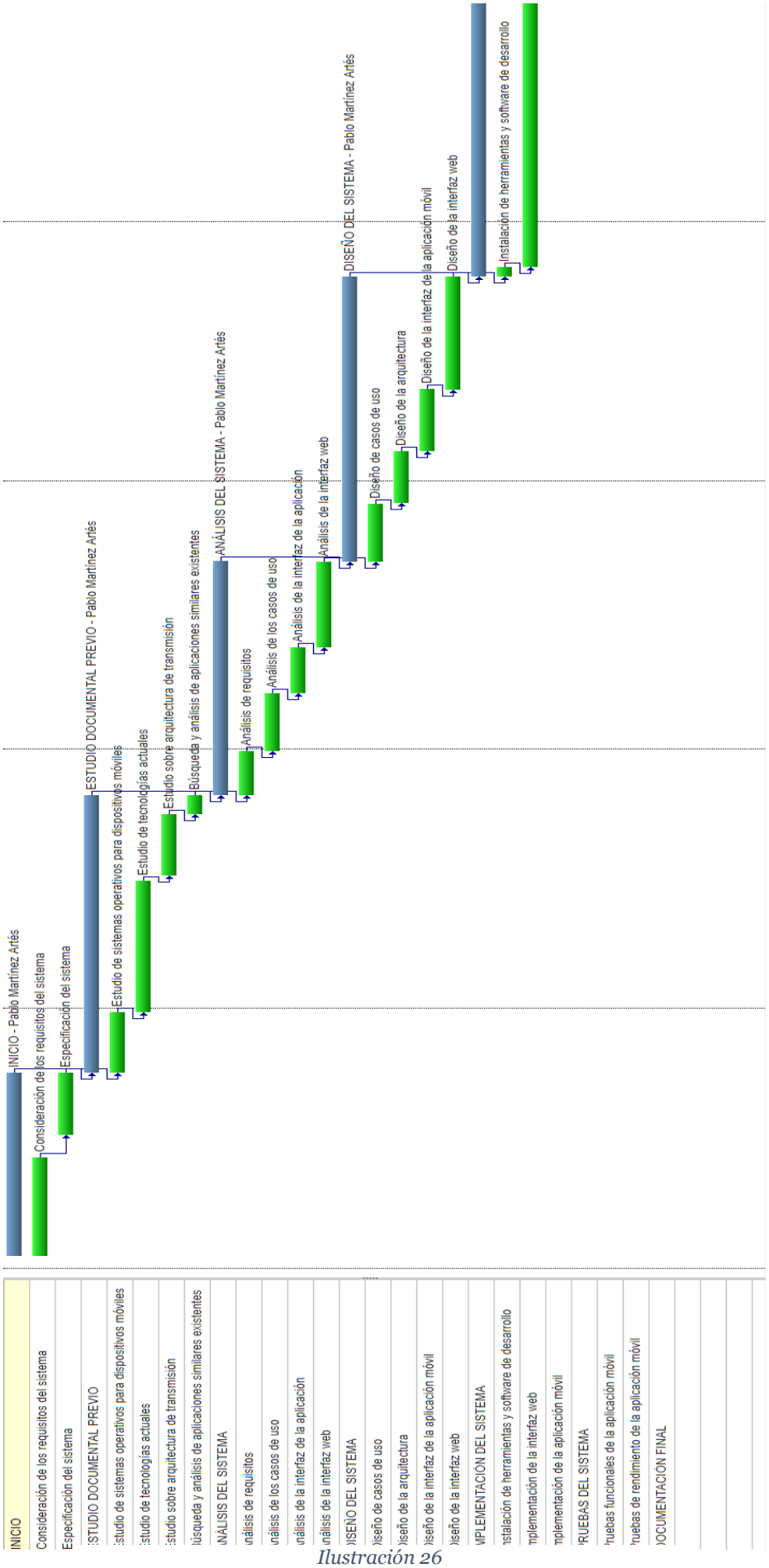


Ilustración 26

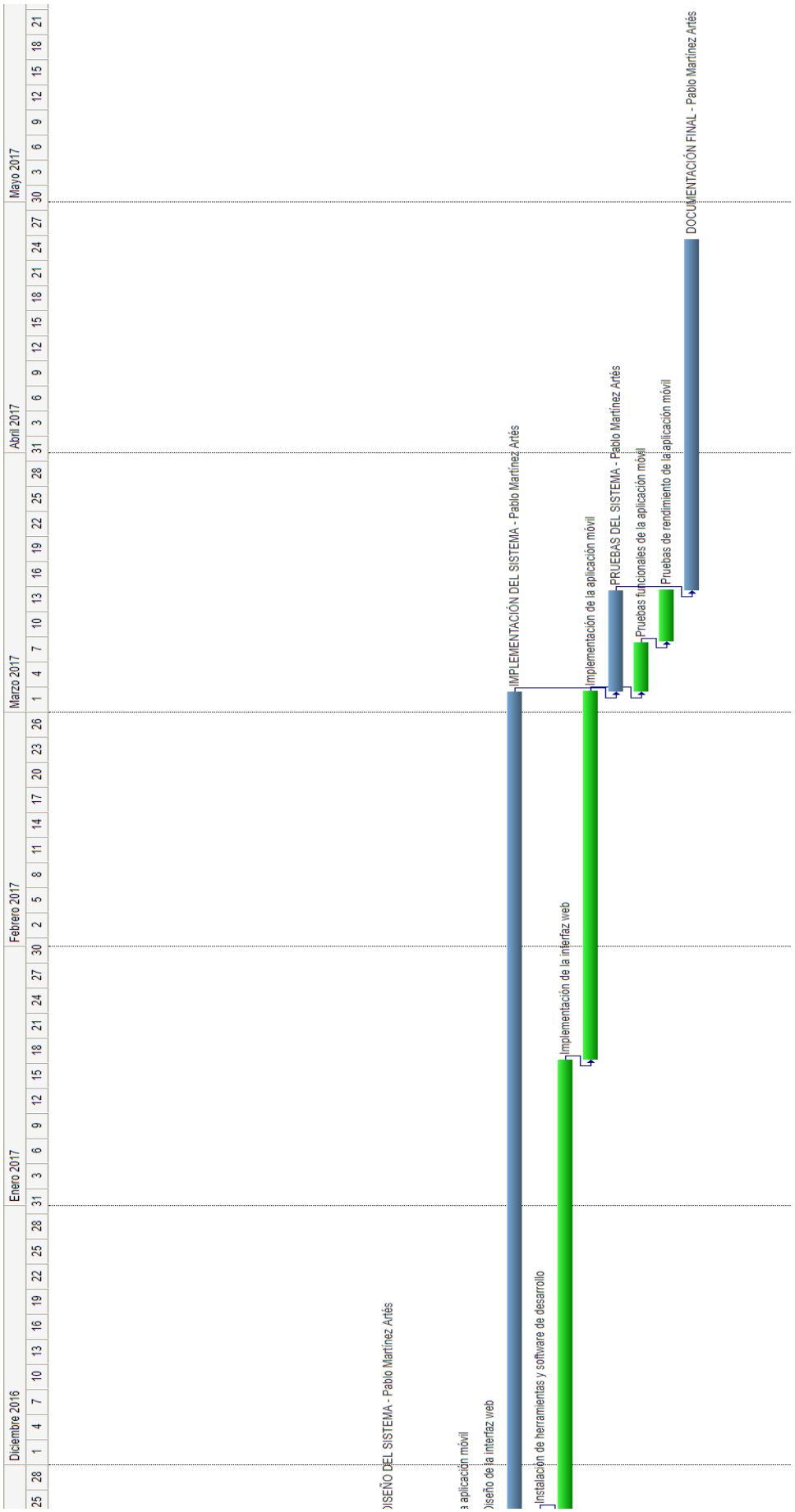


Ilustración 27

4.3.2. Coste del personal

En este apartado analizaremos el coste del personal asociado al proyecto y como se ha mencionado en punto anterior contamos solo con un único recurso humano que desempeñará todas las tareas.

Según el Boletín Oficial del Estado (BOE 2016) el sueldo base anual que se establece para un ingeniero informático incluyendo dos pagas extra es de 19.561,64€ [BOE SUELDO], al cual se le debe añadir lo correspondiente a los costes de la seguridad social que es un 28,3% [COSTESEGSOC].

	Sueldo Base Anual	Sueldo Base Mensual
Total sin aplicar la tasa de la Seguridad Social	19.561,64€	19.561,64 / 12 = 1.630,13€
Tasa de la Seguridad Social	1.630,13 x 0.283 = 461,33€	
Total Sueldo Mensual		1.630,13 + 461,33 = 2091,46€

Sabiendo que el coste de un ingeniero informático al mes es de 2091,46€, que su disponibilidad para trabajar es del 100% y conociendo que la planificación temporal estimada anteriormente en el diagrama de Gantt da como resultado un periodo de 8,72 meses, podemos establecer que el coste total de personal será de:

Coste de personal = 2091,46 x 8,72 = 18.237,53€

4.3.3. Coste del software

En este punto se van a detallar los posibles costes asociados al software empleado tanto en la fase de desarrollo como en la fase de producción del proyecto. Hay que tener en cuenta que mucho software existente en el mercado cuenta con licencias de pago y gratuitas, por lo que a continuación expondremos los costes asociados al software utilizado en este proyecto incluyendo la amortización.

Vamos a suponer que normalmente el software se amortiza en tres años, y sabiendo que la duración estimada del proyecto es de 8,72 meses, la amortización a 3 años supondrá un 24,22% del valor.

SOFTWARE DE DESARROLLO Y PRODUCCIÓN				
Software	**Descripción**	**Amort.**	**Precio**	**Coste**
Windows 10 Professional	Sistema operativo del PC	Incluido en el PC		
Android 2.3.3 o superior	Sistema operativo del móvil	Incluido en el dispositivo		
Android Studio	Entorno de desarrollo de la aplicación móvil	24,22%	0€	0€
Sublime Text	Procesador de texto orientado a la programación	24,22%	0€	0€
Google Chrome	Navegador que actuará de cliente	24,22%	0€	0€
Office Hogar y Estudiantes 2016	Procesador de texto	24,22%	149€	36,1€
Draw.io Diagrams	Extensión de Google Drive para realizar diagramas UML	24,22%	0€	0€
TOTAL				**36,1€**

4.3.4. Coste del hardware

A continuación se realizará un estudio sobre el coste hardware asociado al proyecto que será necesario para poder llevar a cabo el mismo.

Para el desarrollo de este proyecto se ha necesitado un ordenador personal donde poder instalar las herramientas necesarias, entornos de desarrollo, procesadores de texto y el navegador que actuará de cliente. Por otro lado también se han necesitado diversos dispositivos móviles para poder probar la aplicación, un teclado, un ratón y un monitor que por ser equipo informático su amortización se cumple a los tres años.

El ordenador personal es un ordenador que tiene más de ocho años de vida, aunque se le han ido intercambiando diversas piezas con el tiempo, sin embargo dado que estos cambios se produjeron hace más de tres años asumimos que ya ha sido amortizado y no lo incluiremos en el coste total.

Los dispositivos móviles poseen más de tres años de vida por tanto, tampoco serán incluidos en el coste total.

Suponiendo que la amortización de este tipo de dispositivos se cumple a los dos años y que la duración estimada del proyecto es de 8,72 meses, sabemos que el coste de los dispositivos supondrán un 36,33% del coste del valor del producto. A continuación mostramos la tabla con toda la información a forma de resumen del coste hardware al final.

HARDWARE DE DESARROLLO Y PRODUCCIÓN			
Hardware	**Amortización**	**Precio**	**Coste**
Intel Core i5-4670 3.4Ghz	Ya amortizado	227€	0€
LG Nexus 5 2,26GHz Adreno 330 450MHz 2GB RAM 16GB HDD	Ya amortizado	100€	0€
Samsung Galaxy Mini GT-S557OI	Ya amortizado	10€	0€
AeroCool Templarius Arma Gaming Mouse	36,33%	16,75€	6,08€
Monitor Samsung S24D330H 24" LED	36,33%	135€	49,05€
TOTAL			**55,13€**

4.3.5. Coste total del proyecto

Una vez tenemos calculados los costes totales del hardware, software y del personal procederemos a calcular cuál será el coste total del proyecto, para ello incluimos la siguiente tabla donde mostramos un breve resumen de todos los costes por sección. Por otro lado también hay que considerar los costes indirectos u *overhead* del proyecto hasta ahora obviados, y que por norma general suele equivaler a un 20% de los costes directos.

COSTES TOTALES	
Tipo de coste	**Coste**
Coste de personal	18.237,53€
Coste de software	36,1€
Coste de hardware	55,13€
Total (Sin overhead)	18.328,76€
Overhead (20%)	3.665,75€
Total	**21.994,51€**

4.4. Estudio de la viabilidad

Por último para cerrar este capítulo, vamos a realizar un estudio sobre la viabilidad del proyecto desde un enfoque, económico, legal y técnico. Lo que se pretende con esto es conocer si llevar a cabo el proyecto será factible o por el contrario no lo será pudiendo así evitar pérdidas económicas o temporales.

4.4.1. Estudio de viabilidad económica

El estudio de viabilidad económica hace referencia a una compatibilidad financiera para la puesta en marcha de un proyecto. Pero en este caso, el proyecto no va a ser ejecutado por un Ingeniero Informático graduado y se ha realizado un cálculo del coste que tendría el personal como si lo fuese. En este caso el proyecto lo realizaría un estudiante de grado en ingeniería informática a coste cero, ya que ninguna empresa ha solicitado su realización y es un proyecto de ámbito académico.

En cuanto a los costes de hardware y software también hay que mencionar que la mayor parte del software ya estaba instalada o incluida en el ordenador o dispositivo utilizado y en el caso de necesitar licencia se ha usado versiones de prueba o licencias de la Universidad Politécnica de Valencia. Por otro lado ya se disponía de todo el hardware necesario para la realización del proyecto y todo el uso de librerías externas se acomodan bajo licencias libres, resultando ser de forma adicional *open source*, por tanto la viabilidad económica está garantizada.

4.4.2. Estudio de la viabilidad técnica

Riesgo	Probabilidad	Impacto	Medidas
Fallo o avería del hardware de desarrollo	Baja	Medio	Existe un riesgo muy bajo de que se pierdan días de trabajo y datos por una avería en el hardware utilizado para realizar el proyecto pero aun así el riesgo que conlleva merece que tomemos medidas de prevención Para evitarlo se debería de hacer de manera continuada copias de seguridad. Por otro lado, otra medida podría ser tener equipo supletorio para poder seguir con la ejecución del proyecto sin necesidad de esperar a que se repare el equipo averiado.
Especificación incorrecta de los requisitos del sistema	Alta	Alto	Ante este más que posible riesgo y de alto impacto es necesario tomar medidas de prevención como por ejemplo mantener una revisión continua. Hasta que esto no suceda no dará comienzo el desarrollo del proyecto. Por otro lado conforme se vaya avanzando en la realización del mismo se mostrará el estado al cliente.
Carencia de conocimiento de las tecnologías a usar por los desarrolladores	Media	Medio	Este riesgo a pesar de tener relevancia porque en caso de darse puede inducir a un retraso en la finalización del proyecto, no es realmente un riesgo importante puesto que el único recurso humano disponible para desarrollar el proyecto es conocedor de la mayor parte de las tecnologías a utilizar. Por

			otro lado, la documentación que aportan facilitarán la enseñanza del desarrollador.
Baja por despido o enfermedad por parte de algún miembro del equipo de desarrollo	Baja	Alto	Este riesgo junto a una especificación incorrecta de los requisitos del sistema es uno de los que más impacto supondrá en nuestro proyecto, ya que al haber en el equipo de desarrollo un solo recurso humano, si este coge la baja temporal por enfermedad u otra razón, todo el tiempo que no esté trabajando será el tiempo que se retrase la finalización del proyecto, lo que se traduce en un aumento del coste final
Retraso en alguna de las fases del proyecto	Media	Medio	Este riesgo suele ser el más común entre todos los proyectos. Es bastante probable que se den retrasos en algunas fases del proyecto aumentando así el coste final del mismo. Por esto como medida preventiva contras este riesgo se propone incluir más horas de dedicación.

4.4.3. Estudio de viabilidad legal

Por último para finalizar el capítulo debemos realizar un estudio sobre si el proyecto cumple con los requisitos legales necesarios. Hay que destacar que como se trata de un proyecto únicamente de ámbito académico no afecta a la totalidad del desarrollo del sistema pero aun así es necesario realizar un análisis por si en un futuro nos fuera conveniente mercantilizar el producto obtenido.

Por una parte a lo largo de la memoria no hemos mencionado la intencionalidad de guardar datos personales de los usuarios, sin embargo, podría resultar muy interesante añadirlo y en ese caso debemos asegurarnos que se cumpla con la LOPD (Ley Orgánica de Protección de Datos):

- Informar a los titulares de los datos personales para obtener su consentimiento.
- Garantizar la protección de los datos mediante encriptación u otras medidas de seguridad.
- Inscribir todos los ficheros asociados en el Registro General de Protección de Datos notificando a la Agencia Española de Protección de Datos su existencia. (Artículo 26 LOPD. RD 1720/2007).

5. Desarrollo del proyecto

Una vez realizadas tanto la estimación temporal como económica y visto si es viable o no la realización de este proyecto desde el punto de vista económico, técnico y legal podemos pasar a continuación a profundizar el cómo se ha desarrollado la aplicación. Analizaremos en primer lugar el sistema detallando los casos de uso que se obtuvieron de los requisitos funcionales y no funcionales. En segundo lugar, en la fase de desarrollo podremos visualizar el comportamiento del sistema mediante diagramas de secuencia y por último, en la fase de implementación se tratará de entrar en detalle en los aspectos más relevantes del código para esclarecer el funcionamiento de cada tarea.

5.1 Análisis

A continuación en la siguiente fase se realizará el análisis del sistema. Esta parte es muy importante hacerla con cuidado puesto que es fundamental para conseguir que no ocurran fallos en las etapas posteriores del desarrollo del proyecto. El análisis se realizará a partir de los requisitos que obtuvimos de la especificación del sistema, de los cuales concluimos que necesitaremos realizar una sola aplicación móvil que de servicio a una interfaz web que pueda utilizar el usuario. Es necesario, pues, establecer los actores que intervienen con la aplicación y las funcionalidades que pueden llevar a cabo. Por esto, definimos el siguiente diagrama de Casos de Uso del sistema *(Añadimos ilustración explicativa en la siguiente página)*.

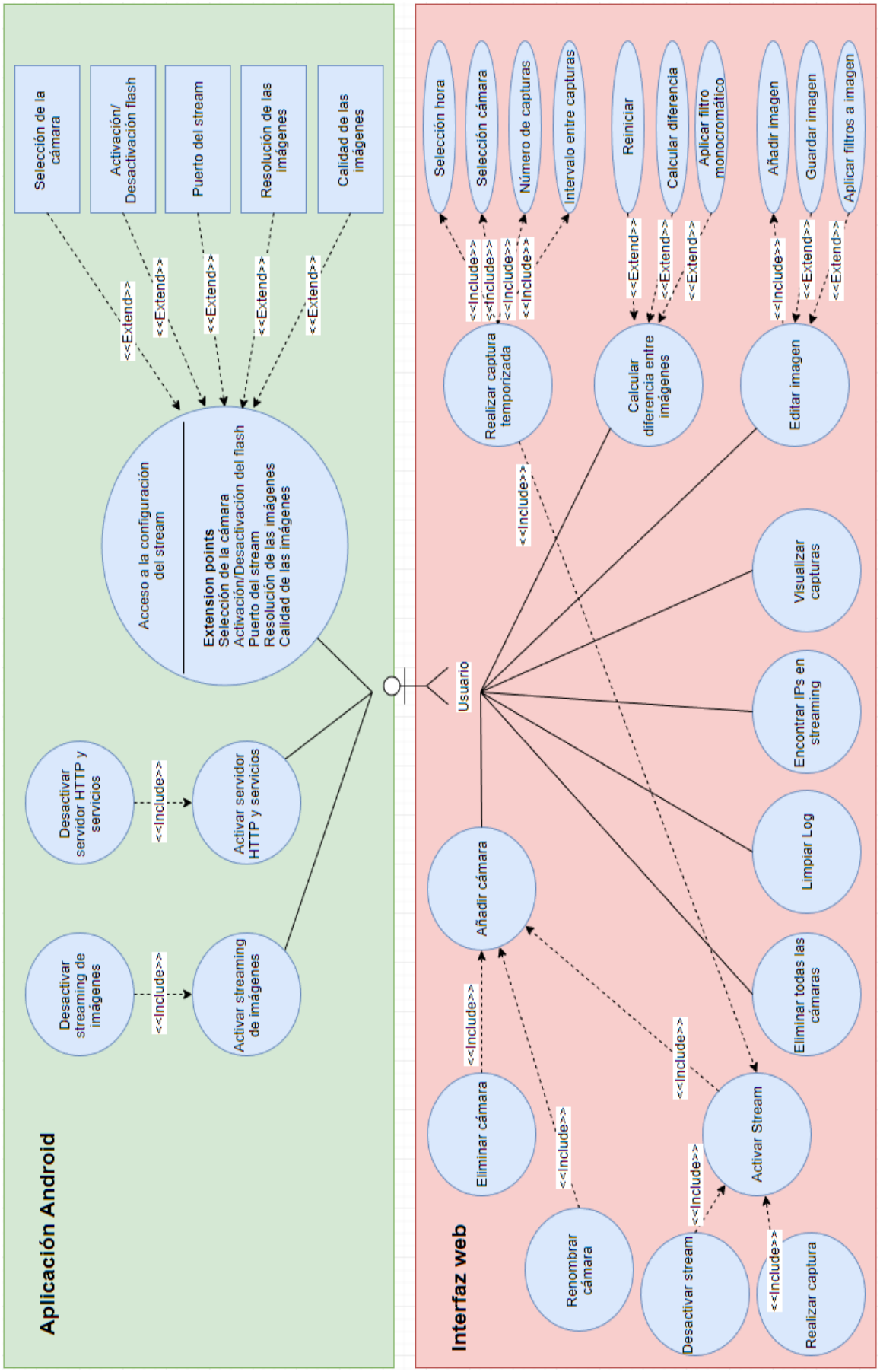


Ilustración 28

Como podemos observar en la anterior imagen el diagrama de casos de uso establece que será un solo actor el que interactúe con el sistema y podrá realizar una serie de funcionalidades englobadas en dos grandes bloques: Los correspondientes a las funcionalidades incluidas en la aplicación móvil y las correspondientes a la interfaz web servida por los servidores web incluidos dentro de la propia aplicación.

A continuación podemos ver con más detalle cada una de las funcionalidades recogidas en los casos de uso:

Activar streaming de imágenes

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá iniciar el streaming de imágenes del dispositivo.
Precondición	El usuario debe de estar conectado a la red de área local.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón con el símbolo de "*play*" situado en la parte izquierda de la actividad principal.

Desactivar streaming de imágenes

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá iniciar el streaming de imágenes del dispositivo.
Precondición	El usuario debe de haber iniciado anteriormente el streaming en el dispositivo.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón con el símbolo de "*stop*" situado en la parte izquierda de la actividad principal.

Acceso a la configuración

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá acceder a la actividad de preferencias del stream.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón con el símbolo en forma de tuerca situado en la parte inferior izquierda de la actividad principal.

Selección de la cámara

Actor	Usuario.

Descripción	A través de este caso de uso el usuario podrá seleccionar la cámara trasera o delantera del dispositivo.
Precondición	El dispositivo debe poseer al menos una cámara para poder seleccionar una.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre la opción **cámara** en la actividad de preferencias, donde posteriormente se desplegará un menú con las cámaras disponibles en el dispositivo.

Activación/Desactivación del flash

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá activar o desactivar el flash de la cámara.
Precondición	El usuario debe poseer flashlight para poder seleccionarlo correctamente
Flujo Normal	El usuario puede realizar esta acción pulsando sobre la opción **Flash Light** en la actividad de preferencias, en la que se mostrará activado con un checkbox marcado en la parte derecha de la opción, y desactivado cuando el checkbox situado a la derecha esté desmarcado.

Puerto del Stream

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá seleccionar el puerto deseado para el stream del dispositivo.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre la opción **Server Port** de la actividad de preferencias, donde se desplegará una ventana para seleccionar numéricamente el puerto deseado.

Resolución de las imágenes

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá seleccionar la resolución de las imágenes que se van a transmitir.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre la

	opción **JPEG Size** de la actividad de preferencias, donde se desplegará un menú con las resoluciones disponibles dependiendo de la cámara seleccionada.

Calidad de las imágenes

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá seleccionar la calidad de las imágenes que se van a transmitir
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre la opción **JPEG quality** situada en la actividad de preferencias, donde se desplegará una ventana numérica y el valor.

Activar servidor HTTP y servicios

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá ejecutar los servidores web, uno para servir la página web y otro para actuar como *API HTTP*.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón con el símbolo de apagado gris situado en la parte inferior derecha de la actividad principal.

Desactivar servidor HTTP y servicios

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá detener los servidores web que sirven la página web y los servicios.
Precondición	El usuario debe haber ejecutado los servidores anteriormente.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón con el símbolo de encendido verde situado en la parte inferior derecha de la actividad principal.

Añadir cámara

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá crear un contenedor para visualizar el stream en la interfaz web.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **ADD CAMERA** de la interfaz web.

Eliminar cámara

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá eliminar el contenedor cámara.
Precondición	El usuario debe haber creado anteriormente el contenedor.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **REMOVE** perteneciente al contenedor que desea eliminar.

Renombrar cámara

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá renombrar el nombre que posee un contenedor cámara.
Precondición	El usuario debe haber creado anteriormente el contenedor.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **RENAME** perteneciente al contenedor que desea renombrar y escribiendo el nuevo nombre que desea para el contenedor en el campo que aparecerá en la parte inferior del contenedor.

Activar stream

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá comenzar a visualizar el stream de un dispositivo transmitiendo.
Precondición	El usuario debe haber creado un contenedor de cámara y especificar en el campo "*IP address*" la dirección del dispositivo elegido.
Flujo Normal	El usuario puede realizar esta acción colocando la

	dirección IP del dispositivo y el puerto seleccionado en las preferencias, en el campo "*IP address*" situado en la parte inferior del contenedor, y posteriormente pulsar sobre el contenedor donde se sitúa el icono "*play*" rojo.
Flujo alternativo	Si el usuario coloca de forma errónea el formato de la dirección ip, o si el usuario no añade el puerto, el sistema le avisará mediante notificaciones de alerta. Por otra parte si el usuario coloca correctamente una dirección Ip y puerto pero que no corresponden a ningún dispositivo en streaming el sistema le avisará mediante una notificación de error.

Desactivar stream

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá detener el stream de un contenedor cámara.
Precondición	El usuario debe haber iniciado previamente el stream en el contenedor cámara.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el contenedor que visualiza el stream.

Realizar captura

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá realizar una captura del stream.
Precondición	El usuario debe haber iniciado previamente el stream en el contenedor cámara.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **CAPTURE** perteneciente al contenedor del cual desee realizar la captura.
Flujo alternativo	Si el usuario no ha iniciado el stream en el contenedor al que pertenece el botón de capture, se le indicará mendiante notificaciones que primero debe inicializarlo en dicho contenedor.

Realizar captura temporizada

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá realizar una captura con temporización sobre el stream.

Precondición	El usuario debe de haber iniciado primero el stream del contenedor del que desea realizar la captura. El usuario debe seleccionar la hora a la que desea realizar la captura. El usuario debe indicar de que contenedor cámara desea realizar la captura.
Flujo Normal	El usuario puede realizar esta acción desplegando el menú de configuración de las capturas temporizadas, situado en la parte media izquierda de la interfaz web, configurando apropiadamente las opciones y posteriormente presionando sobre el botón **START**. El sistema avisará al usuario del comienzo de la captura temporizada si todo ha ido correctamente, y una vez finalice, también el usuario será avisado del término de la captura temporizada.
Flujo alternativo	Si el usuario no configura la hora a la que se debe realizar la captura y esta tiene al menos una diferencia de diez segundos con la hora actual, o si no selecciona la cámara deseada, o si esta no esta inicializada, el sistema avisará al usuario de las correcciones necesarias para iniciar esta herramienta.

Selección hora

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá configurar la hora a la que se realiza la captura temporizada mediante un *Time Picker*.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el *Time Picker* situado en el menú de configuración de captura temporizada parejo a la opción **Hour Selection**.

Selección cámara

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá seleccionar el contenedor cámara del cual desea realizar la captura temporizada.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el selector situado en el menú de configuración de captura temporizada parejo a la opción **Camera Selection**.

Número de capturas

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá seleccionar el número de capturas que desee realizar mediante la captura temporizada.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción estableciendo un valor en el campo situado en el menú de configuración de captura temporizada parejo a la opción **Number of snapshots**.

Intervalo entre capturas

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá seleccionar el intervalo que se establece entre una captura y otra mediante la captura temporizada.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción estableciendo un valor en el campo situado en el menú de configuración de captura temporizada parejo a la opción **Interval between snapshots**.

Eliminar todas las cámaras

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá eliminar todos los contenedores cámara.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **REMOVE ALL CAMERAS** de la interfaz web. Aparecerá un diálogo donde puede confirmar su acción.
Flujo Alternativo	El usuario puede seleccionar la opción **No** del diálogo y en ese caso ninguna acción se realizará.

Limpiar Log

Actor	Usuario.

Descripción	A través de este caso de uso el usuario podrá eliminar todo el contenido del Log informativo de la interfaz web.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **CLEAR LOG** de la interfaz web y que posteriormente el sistema avisará al usuario mediante una notificación.

Encontrar IPs en streaming

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá visualizar cuales son las IPs de los dispositivos que están transmitiendo.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **FIND IPs** de la interfaz web, y visualizará las IPs encontradas en el recuadro inferior.
Flujo Alternativo	Si el usuario no tiene ejecutado el servidor HTTP y servicios, el sistema le notificará con un error que no ha podido acceder al servidor.

Visualizar capturas

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá observar las capturas realizadas a los streams visualizados en los contenedores.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción accediendo al menú de capturas situado en la parte media izquierda de color gris de la interfaz web. Pudiendo seleccionar todas las capturas disponibles o las capturas referentes a una sola cámara.

Calcular diferencias entre imágenes

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá acceder a la herramienta de cálculo de diferencias entre imágenes.
Precondición	*Ninguna*

Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **Make differences between images** que desplegará la sección de la herramienta en la interfaz web.
Flujo Alternativo	Pulsando sobre el botón de la sección o de la sección alternativa, esta se replegará.

Calcular diferencia

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá calcular las diferencias entre dos imágenes dadas.
Precondición	El usuario debe establecer las dos imágenes sobre las cuales quiere calcular la diferencia.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **Calculate differences** dentro de la sección desplegable **Make differences between images**.

Reiniciar

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá reiniciar la herramienta de cálculo de diferencias entre imágenes.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **RESET** situado en la sección desplegable **Make differences between images**.

Aplicar filtro monocromático

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá editar la diferencia calculada entre imágenes volviéndola de un solo color.
Precondición	El usuario debe haber obtenido el resultado de la herramienta de diferencia entre imágenes.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre los botones dispuestos en la parte inferior del resultado de la diferencia situado en la sección desplegable **Make differences between images**, pudiendo así seleccionar entre los diferentes colores *rojo, azul y verde*.

Editar imagen

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá acceder a la herramienta de edición de imágenes mediante filtros.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre la sección desplegable **Apply image filters**.
Flujo Alternativo	Pulsando sobre el botón de la sección o de la sección alternativa, ésta se replegará.

Añadir imágenes

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá añadir una imágen para poder aplicarle los filtros disponibles.
Precondición	*Ninguna*
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón de selección de ficheros situado en la parte inferior de la sección deplegable **Apply image filters**.

Guardar imagen

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá descargar la imagen que ha añadido y editado.
Precondición	El usuario debe haber añadido anteriormente una imagen.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre el botón **Save image** situado en la parte inferior de la sección desplegable **Apply image filters**.

Aplicar filtros a imagen

Actor	Usuario.
Descripción	A través de este caso de uso el usuario podrá aplicar los filtros disponibles a la imagen añadida.
Precondición	El usuario debe haber añadido anteriormente una imagen y tenerla seleccionada.
Flujo Normal	El usuario puede realizar esta acción pulsando sobre los filtros disponibles situados en la parte derecha de

	la sección desplegable **Apply image filters**.

5.2 Diseño

El diseño es la siguiente fase después del análisis del sistema. En esta etapa se pretende mostrar en detalle cómo está definido el sistema. Para lograr esto primero diseñaremos la arquitectura del sistema, a continuación especificaremos el modelo de clases.

5.2.1. Arquitectura del sistema

El modelo de arquitectura en el que nos basaremos para el diseño del sistema será el de cliente/servidor. Tendremos una aplicación móvil que actuará de servidor y esta dará servicio al cliente que se conecte a ella, para poder acceder a la interfaz web y poder realizar una petición para visualizar el streaming. A modo de idea general mostramos en la siguiente ilustración el modelo de la arquitectura del sistema.

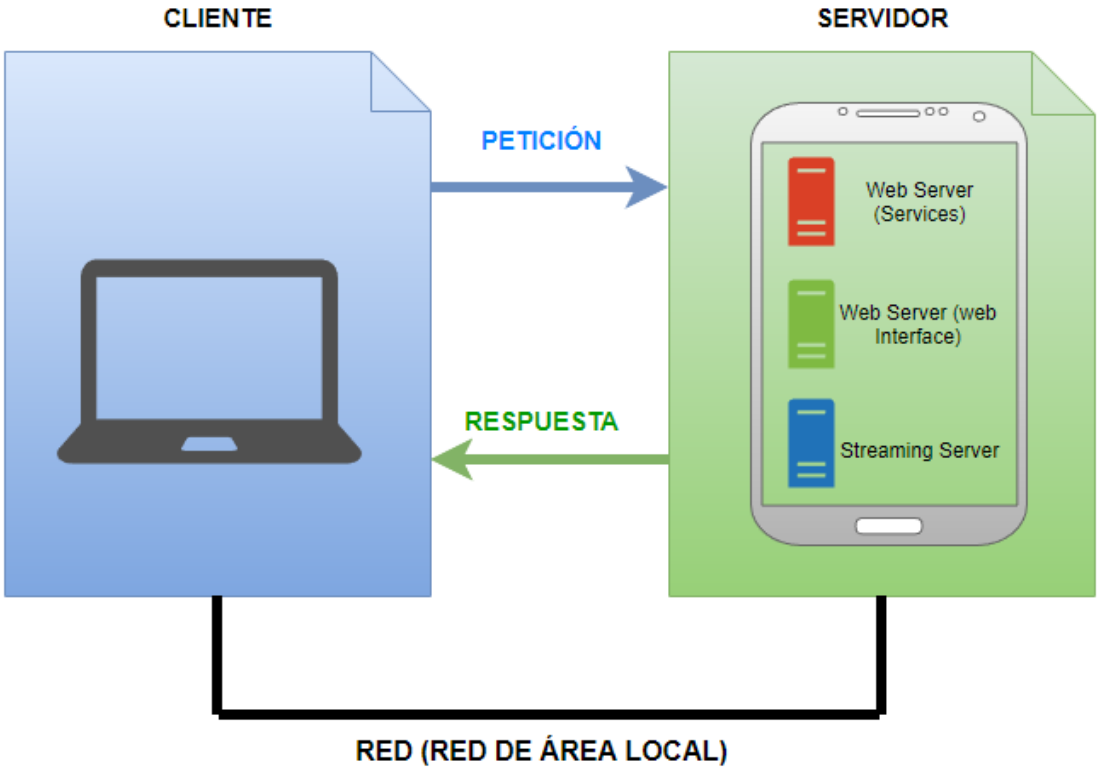


Ilustración 29

Podemos observar mediante la imagen, que nos encontramos frente a una arquitectura de dos capas. Ésta topología nos permite distribuir la carga entre dos máquinas diferentes.

La primera capa comprenderá la *lógica de presentación* y por otra parte *la lógica de negocio*. La *lógica de presentación* se encargará de obtener información del usuario, enviarla a la *lógica de negocio* para procesarla, recibir los datos que provengan de la *lógica de negocio* y presentar los resultados obtenidos al usuario, por tanto servirá la interfaz web de la aplicación mediante una página, actuando por tanto los lenguajes *HTML , CSS y JavaScript*. Por otra parte la *lógica de negocio* actúa de puente entre el usuario y los datos, su función es el procesamiento de los mismos y agrupar las herramientas diseñadas para el tratamiento de los datos y que tiene que cumplir la aplicación, tras la recogida de los datos y su procesamiento su tarea final es enviar el resultado del procesamiento al nivel de presentación, esta capa utilizará lenguaje de *scripting* por lo que las herramientas estarán desarrolladas en el lenguaje de programación *JavaScript*.

La segunda capa se corresponde con la *lógica de datos*. Esta capa se encargará de generar los datos y gestionarlos de forma adecuada, de forma añadida será la encargada de activar los servidores, por lo que parte de la lógica de negocio recae sobre ella, esta capa se servirá del lenguaje de programación *Java* para realizar la generación de datos y su posterior gestión, adicionalmente será empleado el lenguaje *XML* para fabricar los *layouts* de las actividades que componen la aplicación y que servirán al usuario para interactuar con la misma.

Teniendo en cuenta las capas anteriormente definidas nos encontraríamos pues, frente a una arquitectura **Presentación + Lógica / Lógica + Datos**, por lo tanto, frente a una aplicación distribuida. Este modelo nos otorga flexibilidad puesto que nos permite tanto al servidor como al cliente mantener la lógica de negocio realizando cada uno las funciones que le sean más adecuadas.

Como podemos comprobar en la siguiente mostramos el esquema de una aplicación distribuida.

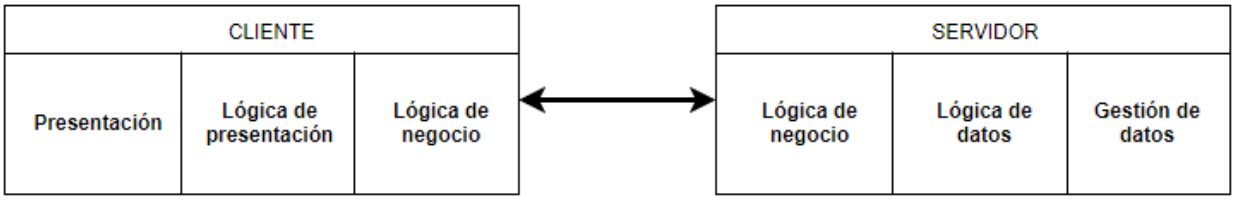


Ilustración 30

Y finalmente se muestra a continuación a través de la siguiente figura una representación esquemática de la arquitectura.

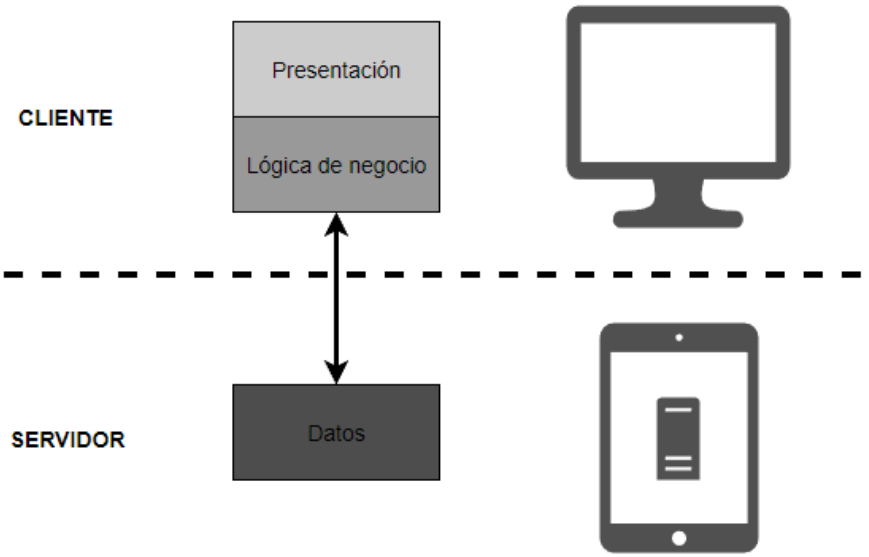


Ilustración 31

5.3 Diagrama de clases de la aplicación Android

Para más tarde profundizar en la implementación nos parece oportuno realizar el diseño de clases correspondiente a la parte correspondiente de la aplicación android a través de un diagrama de clases.

La siguiente figura ilustra el diagrama resultante:

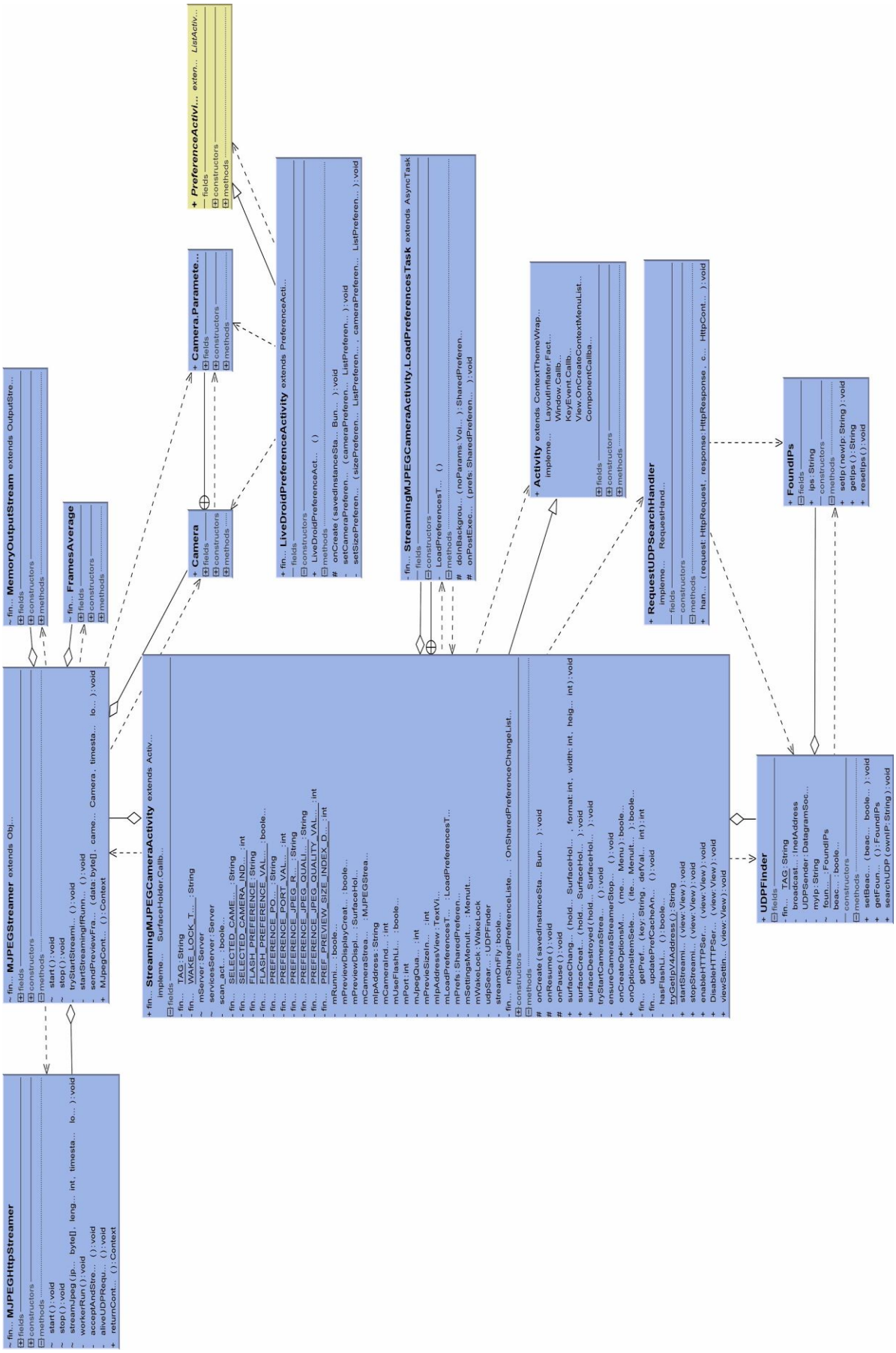


Ilustración 32

5.4 Implementación

Como hemos aclarado en el apartado 5.1 con el diagrama de casos de uso del proyecto y el diagrama de clases de la aplicación en el anterior apartado 5.3, procederemos a continuación a analizar detalladamente el proyecto en dos grandes apartados. Por una parte analizaremos las clases *Java* pertenecientes a la aplicación *Android* y su funcionalidad dentro del proyecto. Por otra parte analizaremos la interfaz web que será servida al usuario a través del servidor web y que le dotará de ciertas herramientas para conseguir la interactividad deseada en el proyecto.

5.4.1 Implementación de la aplicación android

Igual que hemos mencionado en el encabezado de la sección, procederemos a analizar las diferentes clases *Java* que componen la aplicación *Android*, de esta forma conectaremos dichas clases a la funcionalidad a la que están destinadas dentro del proyecto, para ello dividiremos individualmente en fragmentos las acciones que realiza la aplicación *Android* del proyecto, e incluiremos en cada una de ellos las clases *java* que participan.

❑ Actividad principal o StreamingMJPEGCameraActivity.java

Esta será la actividad principal una vez se lance la aplicación móvil, es decir, será lo primero que visualizará el usuario una vez ejecute la aplicación en el terminal. Este primer fragmento comprende el marco a partir del cual el usuario es capaz de diferentes acciones, de entre las cuales destacamos las de mayor relevancia:

- ➔ La inicialización por defecto de las preferencias elegibles por el usuario: cámara elegida, puerto del stream, activación del flash, la resolución de las imágenes transmitidas, o la calidad de imagen de las mismas.
- ➔ Se resolverá la dirección ip de red del dispositivo a través del método *tryGetIpV4Address()*, que iterará sobre las diferentes interfaces de red del dispositivo, descartando las direcciones de loopback y devolviendo como resultado la ip que coincida con una del tipo ipv4. Posteriormente el resultado será mostrado en la actividad.
- ➔ *startStreaming()* es el método que realiza la interacción entre el botón *play* del layout de la actividad con el inicio del streaming del dispositivo a través del uso del método *tryStartCameraStreamer()* bloqueando por seguridad la cámara del dispositivo, e incluye la modificación sobre el layout para ocultar el botón *play* y mostrar el botón *stop*, se mostrará un diálogo en el dispositivo para indicar que la acción se ha llevado a cabo, por último el valor booleano *streamOnFly* se tornará en *true* ya que este es indicador del estado del stream.

- ➔ *stopStreaming()* es el método que realiza la interacción entre el botón *stop* del layout de la actividad con la finalización del streaming del dispositivo a través del método *ensureCameraStreamerStopped()* que detendrá el streaming y liberará la cámara del dispositivo para su uso posterior por otra aplicación si se requiere. Incluye la modificación sobre el layout de la actividad para ocultar el botón *stop* y mostrar el botón *play*, también se mostrará un diálogo en el dispositivo para indicar al usuario la acción que ha llevado a cabo, por último el valor booleano *streamOnFly* que denota el estado del stream se tornará al valor *false*.
- ➔ *enableHTTPServer()* proporcionará la capacidad al usuario de ejecutar los servidores web de la aplicación que ofrecerán la interfaz web y los servicios web. Se activará pulsando sobre el botón de encendido gris del layout de la actividad, se notificará al usuario mediante un diálogo que ha activado los servidores y la dirección a la cual debe acudir para acceder a la interfaz web a través del navegador. Previa a la ejecución de los servidores se comprobará si el dispositivo tiene la ip previamente asignada de la anterior búsqueda. Este método también realizará una modificación sobre el layout de la actividad para ocultar el botón de encendido gris y mostrar el botón de encendido verde.
- ➔ *DisableHTTPServer()* dará la capacidad al usuario para detener los servidores web ejecutados a través de *enableHTTPServer()*, se ejecutará presionando sobre el botón de encendido verde del layout de la actividad, se notificará al usuario que ha detenido los servidores mediante un diálogo y realizará una modificación sobre el layout de la actividad para ocultar el botón de encendido verde y mostrar el botón de encendido gris.
- ➔ *viewSettings()* permite al usuario acceder a la actividad de preferencias, realizando una instancia de dicha actividad y posteriormente iniciándola, de esta forma la actividad principal queda en pausa mientras nos encontramos en la de preferencias. Puede ejecutarse pulsando sobre el botón de *settings* con símbolo de un engranaje.

Para realizar el inicio y la detención del streaming de imágenes es necesario la creación de un objeto de la clase *MJPEGStreamer()* el cual poseerá los métodos *start()* y *stop()* que iniciarán y detendrán el stream respectivamente y que veremos más adelante en el fragmento centrado en explicar la generación dicho stream de imágenes. Por otra parte, para la ejecución de los servidores web, es necesario también, realizar una pequeña configuración antes de poder iniciarlos, también entraremos más adelante en detalle, en el fragmento que define la configuración de dichos servidores, así como su inicialización y funcionamientos básicos.

❑ Actividad de referencias o LiveDroidPreferenceActivity.java

En este fragmento nos centraremos en detallar las funciones que alcanzará la actividad de preferencias de la aplicación android.

Durante la creación de esta actividad se realizará un listado de las cámaras disponibles en el dispositivo que ejecute la aplicación, primero se creará una referencia al apartado correspondiente de la actividad, este es el denominado *camera* de tipo *Listpreference* y se utilizará como argumento para el método *setCameraPreferences()* a fin de rellenar dicho apartado. Para realizar esto, se referencia el objeto global *Camera* de android, que contiene todas las herramientas para poder manejar o conocer los aspectos del elemento hardware Cámara del dispositivo. Se obtiene el número de cámaras disponibles utilizando el método *getNumberOfCameras()* sobre dicho objeto, una vez disponemos del número de cámaras disponibles se obtiene el objeto *cameraInfo* que guardará las características de las mismas más adelante. Mediante un bucle limitado por el número de cámaras que se han encontrado, obtenemos la información de cada una de ellas mediante *getCameraInfo()* que requiere del índice de cámara y del objeto *cameraInfo* para almacenar la información de la misma, por lo tanto utilizaremos el índice de iteración del bucle y el anterior objeto *cameraInfo* creado como argumentos para poder referenciar una cámara en concreto. Tras ir obteniendo los resultados de esa operación, observaremos al parámetro *facing* del objeto *cameraInfo* que actualmente poseamos, para determinar si la cámara actualmente referenciada es la trasera o la delantera, y por tanto clasificarlas como tal en la sección *camera* de la actividad de preferencias.

De forma análoga durante la creación de la actividad y una vez realizado el listado de cámaras, se realiza el listado de resoluciones disponibles para dichas cámaras. De esta forma, a partir de los índices calculados anteriormente, estos son utilizados para en un primer paso abrir la cámara mediante el método *open()*, con ello son recuperados los parámetros de la misma mediante el método *getParameters()*. Recuperaremos todos los tamaños posibles en una lista de tipo *Size* y en la cual encontraremos la anchura (height) y la altura (width) correspondientes, todo ello utilizando *getSupportedPreviewSizes()* sobre los parámetros que anteriormente nos han sido devueltos. Realizaremos el mismo planteamiento que hemos seguido anteriormente, utilizaremos un bucle utilizando la longitud de la lista obtenida para poder ir iterando entre los elementos de la lista *Size*, a partir de cada uno de ellos utilizaremos su índice correspondiente el cual usaremos como argumento para el método *get()* de la lista, lo que nos otorgará individualmente un elemento *Size* y a partir del cual podremos obtener su atributo width y su atributo height. Tras ello los almacenamos en su sección correspondiente de las preferencias *JPEG Size*.

Adicionalmente hemos añadido un evento clic en ambas preferencias para realizar estas acciones descritas, siendo por tanto, *setCameraPreferences()* el método ejecutado cuando se detecta un clic en la sección *camera* de preferencias, y el método *setSizePreferences()* siendo ejecutado cuando es detectado un clic en la sección *JPEG Size* de preferencias.

El resto de elementos de la actividad, son simples objetos *EditTextPreference*, los cuales en sus eventos clic permiten ser editados y posteriormente guardados, por lo que su lógica es relativamente simple y su definición ha sido únicamente necesaria dentro del layout que usará la propia actividad.

❑ Stream de imágenes o flujo de imágenes

Este fragmento analiza una de las partes más importantes del proyecto, la encargada de generar el stream o flujo de imágenes a partir de la cámara del dispositivo y que posteriormente visualizará el usuario utilizando la interfaz web servida para el navegador del cliente o simplemente utilizando el propio navegador. Analizaremos la toma y creación de las imágenes, que elementos intervienen y de que forma estas son enviadas para su recepción por un cliente, además de observar de que forma son enviadas para asegurar la codificación *MJPEG* anteriormente mencionada.

Las clases que se estudiarán durante este fragmento y que serán responsables de generar el stream o flujo de imágenes serán: *MJPEGStreamer.java*, *MJPEGHttpStreamer.java*, *MemoryOutputStream.java* y *FramesAverage.java*.

❑ *MJEGStreamer.java*

Esta clase actuará como objeto intermediario entre la actividad principal de la aplicación y el resto de clases expuestas en este fragmento. Tiene como objetivos la toma de imágenes a través del callback *onPreviewFrame* de la cámara, su captura a formato *Yuv* nativo de android y su posterior compresión a formato *jpeg*, además ofrecerá la información obtenida hacia la clase *MJPEGHttpStreamer.java* para su posterior transmisión.

Esta clase será instanciada en la actividad principal de la aplicación, de forma que recibe como parámetros en su constructor las preferencias por defecto o que a seleccionado el usuario, tales como el índice de la cámara que corresponde con la cámara frontal o trasera, el uso o no del flash de la cámara especificado con un booleano, el puerto seleccionado para el stream o flujo, el índice correspondiente al tamaño o resolución seleccionado para las imágenes (*previewSize*), la calidad de las imágenes y por último la referencia al elemento *SurfaceView* del layout que nos permitirá pintar las imágenes recogidas por la cámara.

El objeto construido con esta clase ofrecerá principalmente dos métodos, *start()* y *stop()* para su uso externo a la clase.

Cuando el método *start()* es llamado es creado un hilo del tipo *HandlerThread* que nos permitirá crear una cola mensajes e ir realizando las tareas disponibles en un *Handler* instanciado a partir del *HandlerThread* según el mensaje que ha sido recibido, por ello poseemos dos tipos de mensajes *TRY_TO_START_MESSAGE* y *SEND_PREVIEW_DATA_IMAGE*. En el caso actual el mensaje enviado corresponderá con *TRY_TO_START_MESSAGE* e implicará la ejecución del método *tryStartStreaming()* definido como tarea de dicho mensaje.

tryStartStreaming() tratará de capturar las excepciones *RuntimeException* para detectar si el acceso a la cámara está disponible, si es recibida esta excepción, tras un tiempo definido anteriormente en la variable open_camera_poll_interval_ms se

probará de nuevo al intento de acceso a la cámara del dispositivo. Cuando se consigue el acierto, se accederá al método *startStreamingIfRunning()*.

Durante la ejecución de *startStreamingIfRunning()* se procederá a preparar la cámara para la posterior captura de imágenes. Primero se configurará el tamaño en el cual se capturarán las imágenes obteniendo la lista de los mismos mediante *getSupportedPreviewSizes()* y seleccionando el correspondiente de acuerdo al índice recibido en el constructor de la propia clase, a continuación guardaremos este parámetro mediante *setPreviewSize()* para establecer finalmente en la cámara a que tamaño o resolución serán realizadas. El siguiente parámetro a configurar será la activación o no del flash de la cámara según como lo haya establecido el usuario en las preferencias, accediendo a él a través del uso de *setFlashMode()* con el cual podremos establecer el flash de la cámara en modo antorcha. Finalmente configuraremos el rango de imágenes por segundo que alcanzará la toma de imágenes que será visualizada en el *SurfaceView*, para ello recuperaremos los posibles valores mediante *getSupportedPreviewFpsRange()* y del cual seleccionaremos el primer y último elemento, a continuación guardaremos este parámetro en la cámara mediante *setPreviewFpsRange()*. Para poder producir un stream o flujo de imágenes será necesaria la creación de un buffer de datos y de esta forma antes de ello debemos calcular el tamaño que necesitará, siempre basado en que este buffer debe contener un frame o imagen, su tamaño según la documentación, debe ser calculado como [*width x height x bytesPerPixel*] sin embargo para este proyecto se quedará corto ya que son recibidos diversos errores de espacio insuficiente por lo que este, una vez recalculado, debe ser 2.5 veces más grande que como se especifica, quedando por tanto [*width x height x bytesPerPixel x (3/2+1)*], mediante la clase *MemoryOutputStream.java* será creado el buffer necesario. Posteriormente será creado el objeto MJPEGHttpStreamer y se iniciará dicho objeto, finalmente será iniciada también el *preview* de la cámara la cual comenzará a disparar el evento que hemos sobreescrito, *onPreviewFrame*.

Una vez iniciado el *preview* de la cámara, por cada imágen o frame capturado por la misma el evento *onPreviewFrame* será disparado y por tanto, el código que hemos reescrito para el evento. La acción que se realizará en este evento será el envío del mensaje *SEND_PREVIEW_DATA_IMAGE* al *Handler* anteriormente creado. La tarea que realizará el *Handler* al recibir dicho mensaje será la ejecución del método *sendPreviewFrame* que realizará el cálculo de imágenes por segundo utilizando la cantidad actual de imágenes que se han disparado, ya que con cada ejecución de este código puede traducirse a la llegada de una nueva imagen y el tiempo actual del dispositivo, consiguiéndolo a través de este método *SystemClock.elapsedRealtime()*. Posteriormente se realizará el guardado de la imagen del *preview* de la cámara generándola en formato *Yuv*, que requiere de la información recogida en el evento *onPreviewFrame*, el formato de la cámara obtenido a través de los parámetros de la cámara y la altura y anchura establecidos para la imagen. Una vez obtenida la imagen en formato *Yuv* se procede a su compresión a formato *jpeg* a través del método *compressToJpeg*, ésta es almacenada en el buffer anteriormente creado mediante *MemoryOutputStream* durante su compresión, y posteriormente dicho buffer es transmitido al objeto *MJPEGHttpStreamer* mediante su método interno *streamJpeg* que requerirá del buffer y de la longitud del mismo. Por último es limpiado el buffer

utilizado para la recepción de la siguiente imagen para el siguiente evento *onPreviewFrame*, tras ello devolvemos el buffer al servicio de la cámara.

Por otra parte el método *stop()*, realizará la ejecución del método *stop()* sobre el objeto *MJPEGHttpStreamer* una vez hemos comprobado por seguridad que este no se encuentra como *null*, por último la cámara es liberada para otra aplicación si se requiere y el *HandlerThread* es detenido.

- ❏ *MJPEGHttpStreamer.java*

Esta clase se encargará de los mecanismos necesarios para servir las imágenes a través de un stream o flujo de datos, y de realizarlo para conformarlo con formato *MJPEG*.

En la definición de los atributos de la clase se encuentran las cabeceras *HTTP* que portarán las imágenes al ser enviadas y que se definirán dentro de la variable *HTTP_HEADER* por otra parte se encuentra el contenido de la variable *BOUNDARY* permitirá al cliente la identificación de las diferentes imágenes durante el stream o flujo de datos. El constructor de esta clase, requiere del puerto seleccionado en las preferencias de la aplicación y el tamaño necesario para el buffer calculado anteriormente durante la ejecución de la clase *MJPEGStreamer.java*.

La clase *MJPEGHttpStreamer.java* posee el método *start()* que ejecutará el bloque de código indicado para un hilo anteriormente creado en los atributos de la clase. Dicho código ejecutará el método *acceptAndStream()* que creará un *ServerSocket* y un *Socket* para hacer posible la conexión de un cliente al mismo. Los datos que contienen los búferes creados en los atributos de la clase y que contendrán las imágenes en formato *jpeg*, serán volcados sobre el *DatagramOutputStream* creado para la comunicación con el *Socket*, añadiendo además las cabeceras necesarias para su transporte por el protocolo *HTTP* incluyendo los segmentos de la variable *BOUNDARY* que permitirán secuenciar las imágenes *jpeg* enviadas y por tanto crear el formato *MJPEG* en el stream o flujo de datos que percibirá el cliente. El buffer es rellenado y enviado de nuevo mientras el valor booleano condicionante lo permite, ya que esta acción se encuentra encerrada en un bucle *while*. Cuando esta condición deja de darse el socket es cerrado.

El método *stop()* de esta clases simplemente terminará la actividad que realiza el hilo iniciado en el método *start()*.

Por último, *MJPEGHttpStreamer* posee el método *streamJpeg* utilizado como hemos mencionado anteriormente por *MJPEGStreamer*. Requiere de un búfer de datos, su longitud y del tiempo anteriormente calculado en *MJPEGStreamer* del reloj del sistema, como argumentos. Su trabajo es la copia del búfer de datos obtenido a uno de los creados internamente en la clase para su posterior volcado en el *DatagramOutputStream* creado para la comunicación del *Socket*, el tiempo del sistema es utilizado para una de las cabeceras *HTTP* que contendrá cada una de las imágenes y que simplemente indicará su fecha de creación.

- *MemoryOutputStream.java*

Esta clase proporcionará las herramientas para la creación del buffer en la clase *MJPEGStreamer* que contendrá las imágenes obtenidas por la cámara en el evento *onPreviewFrame*. Extenderá de la clase *OutoutStream* e implementará diferentes métodos de escritura en el buffer según la necesidad. Entre ellos la utilización del método *write* para la escritura de un solo byte, la escritura de un array de bytes[] o la escritura de un array de tipo byte[] incluyendo un offset para su posición exacta en la escritura y un contador.

Por otra parte también es posible modificar la longitud del array de bytes[] denominado buffer internamente, mediante el método *seek* el cual es usado durante el método *sendPreviewFrame* en la clase *MJPEGStreamer*.

- *FramesAverage.java*

FramesAverage.java es utilizado en la clase *MJPEGStreamer* para el cálculo de los Fps o imágenes por segundo que se producen durante la visualización de las imágenes mediante el *preview* de la cámara. Se compone de dos sencillos métodos, el primero, denominado *update()* realizará una actualización del valor dado inicialmente durante su instanciación, para retener un valor actualizado de las imágenes por segundo que se están produciendo. El segundo método es *getAverage()* que permite la recuperación de las imágenes por segundo calculada internamente en la clase.

- **Búsqueda a través de broadcast UDP**

Una de las funcionalidades que incorporará el proyecto será un sistema que permita realizar la búsqueda de las *ips* correspondientes a los dispositivos que se encuentran en ese momento transmitiendo las imágenes de sus respectivas cámaras. Para ello se han dedicado tres clases *java* dentro del proyecto que otorgarán a la aplicación la capacidad de recuperarlas y de esta forma no obligar al usuario a recordarlas, apuntarlas o volver a consultarlas en los dispositivos, dichas clases corresponderán con *UDPFinder.java*, *FoundIps.java*, *RequestUDPSearchHandler.java* y el método *aliveUDPRequest()* contenido en la clase *MJPEGHttpStreamer.java*. A continuación analizaremos con más detalle las clases anteriormente mencionadas.

- *FoundIps.java*

Esta clase java tiene un uso muy simple, será el objeto que será recuperado y configurado por *UDPFinder.java* y *RequestUDPSearchHandler.java* ya que su objetivo es almacenar las *Ips* que se hayan encontrado mediante la búsqueda UDP.

Consta de un atributo *String* encargado de contener las *Ip* y el cual finalmente será recuperado para su lectura y por consiguiente, el conocimiento del resultado de la búsqueda. Llevará consigo tres métodos: *setIp()*, *getIps()* y *resetIps()*.

setIp() será el método encargado de añadir una dirección *ip* al atributo *ips* de la clase, por ello, necesitará como argumento una cadena *String* que representará una *ip*. Éste método comprobará que el argumento recibido no exista en la cadena, de esta forma se elude la repetición de direcciones *ip* dentro del atributo *ips*.

getIps() devolverá el atributo *ips* de la clase *FoundIps.java*.

resetIps() vaciará el atributo *ips* de la clase *FoundIps.java*, es decir, tornaremos el atributo como cadena vacía para reiniciarlo.

- Método *aliveUDPRequest()*

La clase *MJPEGHttpStreamer.java* ha sido mencionada anteriormente durante el análisis que se concentra en explicar cómo se genera el stream o flujo de imágenes. En el actual fragmento nos centramos en cómo se realiza la búsqueda de las *ips* de los dispositivos en streaming a través de un *broadcast UDP*, y concretamente, la función que posee este método para ello.

Este método es ejecutado en el propio método *start()* de la clase *MJPEGHttpStreamer.java* de forma que es utilizado junto a *acceptAndStream()* anteriormente mencionado.

Está compuesto internamente como un *AsyncTask* que nos permitirá ejecutarlo en segundo plano, fuera del hilo principal y del hilo de la interfaz de usuario. Las primeras acciones que emprende este método son la creación de un *DatagramPacket* y un *DatagramSocket* para la comunicación. Es utilizado un bucle *while* junto a una variable booleana, que será configurada a *true* o *false* durante los métodos de la clase *MJPEGHttpStramer.java start()* y *stop()* respectivamente. Mientras el bucle sigue activo, el socket permanece bloqueado a la espera de la llegada de un mensaje durante dos segundos, especificados anteriormente para no paralizar totalmente el socket en caso de que un mensaje haya sido enviado con anterioridad al modo escucha del socket abierto. Si un mensaje es recibido por el socket, es preparado el mensaje de respuesta y enviado a la dirección *ip* obtenida del mensaje recibido. Si el bucle sigue activo repetirá esta acción indefinidamente en segundo plano.

En caso contrario, si el stream o flujo del dispositivo es detenido, entonces el valor booleano de la variable cambia a *false*, y el bucle es terminado siendo por tanto la última acción el cierre del *DatagramSocket* que al inicio hemos creado.

- *UDPFinder.java*

La clase *UDPFinder.java* es utilizada para realizar el propio *broadcast UDP* y recoger las respuestas que se reciban a raíz de este *broadcast*. Implementa tres métodos diferentes para realizar su función, estos son: *setBeacon()*, *getFoundIP()* y *searchUDP()*.

Para poder mantener una lista actualizada de los dispositivos que estén a la espera de recibir un paquete UDP y poder contestar al origen, se ha diseñado el broadcast de forma que es ejecutado dentro de un objeto *Timer* de *java-Android* , gracias al cual

podremos ejecutar una secuencia de código con cierto *delay* y cada cierto *intervalo* de tiempo. Por ello, es diseñado el método *setBeacon()* que da la capacidad de cambiar el valor booleano del atributo *beacon*, que permitirá detener el *Timer* si el mismo tiene valor *false* y por tanto el bucle que produce, parando así el *broadcast UDP*.

UDPFinder.java hace uso de la clase *FoundIps.java* para poder almacenar correctamente las *ips* encontradas, y por tanto poder devolverlas adecuadamente, ya que de otra forma no sería posible la devolución del valor de las mismas si no es a través de un objeto. Por ello el método *getFoundIP()* devolverá el objeto *FoundIps.java* creado en esta clase, para la consulta de las *ips* que han sido halladas y almacenadas en el mismo.

searchUDP() es el método que realizará la búsqueda de *ips* a través de un *broadcast UDP*, requerirá de un argumento, que deberá ser un *String*, el mismo representará la dirección *ip* del dispositivo, la cual será empleada para obtener la *ip* de difusión de la red de área local. Tras la obtención de la *ip* de difusión es abierto un *DatagramSocket* para la comunicación, es configurado un *DatagramPacket* para que sea dirigido hacia la *ip* de difusión y sea recibido por todos los dispositivos. Tras el envío del paquete, el socket de comunicación se queda bloqueado a la escucha de recibir las respuestas del envío anteriormente realizado. Este será incluido dentro de un bucle *while* que permanecerá activo durante quince segundos, realizando la comprobación del tiempo registrado al inicio y el actual del sistema, durante el bucle si un paquete es recibido es incluida la *ip* de su origen al objeto *FoundIps*. Una vez acabado el tiempo límite en el cual es ejecutado el *while*, el socket es cerrado. Excluyendo la creación de las variables en la clase, el resto del código es implementado dentro de un objeto *Timer* que es ejecutado continuamente tras un *delay* e *intervalo* de tiempo especificado, por lo que el objeto *FoundIps* es reiniciado, es vuelto a crear el *DatagramSocket* y enviado un nuevo paquete a la *ip* de difusión de la red para posteriormente volver a la escucha de los mensajes que lleguen de los destinos que contesten a dicha difusión. El atributo booleano *beacon* permitirá la detención de dicho *Timer* que puede ser modificado a través del ya mencionado método *setBeacon()*.

- *RequestUDPSearchHandler.java*

Para poder recuperar los resultados de la búsqueda por *broadcast UDP* a través de la interfaz web, es necesario un mecanismo que permita recuperar por tanto, el objeto *FoundIps* que contiene los resultados mediante una petición al servidor web iniciado. El servidor web escogido para el proyecto es *AndServer* como hemos visto anteriormente. Este servidor ofrece la posibilidad de construir un *Api Http* a través de los *RequestHandler* en *Android*, por lo que simplemente debemos registrar con una *uri* al *RequestHandler* que emplearemos, en este caso utilizaremos *RequestUDPSearchHandler.java*. Esta clase simplemente creará una referencia al objeto *UDPFinder* y accederá a su objeto *FoundIps* mediante el método *getFoundIp()* del mismo, tras recuperar dicho objeto son recuperadas las direcciones *ips* empleando sobre ese objeto el método *getIps()* que serán devueltas a la petición que originalmente se ha realizado al servidor.

❑ Servidores web: configuración e inicialización

Para poder servir la interfaz web y el servicio de búsqueda de las *ips* de los dispositivos es necesaria la inicialización de dos servidores web, el primero servirá la interfaz web para el usuario y el segundo será utilizado para acceder al servicio. Dado que estos servidores son aportaciones incluidas al proyecto mencionaremos de que forma realizamos su configuración y en que momento pueden ser ejecutados por el usuario.

La decisión de realizar diversas instancias del servidor para darle un uso diferente a cada una, es para repartir los roles, de igual forma las instancias para los servidores no requieren de gran capacidad para ser ejecutadas y mantenidas por lo que es totalmente viable tener ambas debido a su ligereza. De este modo una de las instancias es dedicada a servir únicamente la interfaz web para el usuario y la otra para servir únicamente los servicios, en este caso el servicio de búsqueda. De esta forma reducimos al usuario la visibilidad del uso de estos servicios.

Ambos servidores son configurados y posteriormente inicializados durante la ejecución del método *enableHTTPServer()* situado en la actividad principal de la aplicación. Estos servidores a su vez serán detenidos durante la ejecución del método *disableHTTPServer()* también situado en la actividad principal de la aplicación. La ejecución de estos métodos es realizado en el evento clic del botón de encendido gris para el caso de *enableHTTPServer()* y del botón de encendido verde para el caso de *disableHTTPServer().*

Para inicializarlos es necesaria la creación de la referencia al objeto *AssetManager* de android a través del cual podremos obtener la ruta en el dispositivo de los *assets* de la aplicación, indicandole pues a la configuración del servidor donde se sitúan los ficheros que conformarán el *website* que se servirá tras su ejecución y acceso, por otra parte se configurará el puerto deseado para su ejecución y acceso.

De esta forma y tal y como se indica en el sitio del autor de la aportación, las directrices para su configuración son las siguientes:

Para el caso del servidor web que servirá la interfaz lo configuraremos de la siguiente forma:

```
AssetManager mAssetManager = getAssets();

WebSite website = new AssetsWebsite(mAssetManager, "");

AndServer andServer = new AndServer.Build()

        .website(website)

        .port(8081)

        .build();
```

De esta primera forma accedemos a la raíz del directorio *assets* del proyecto y el propio servidor identifica los ficheros de la misma forma que lo realiza un servidor *apache* jerárquicamente, localizando el fichero *index.html* como la página principal y teniendo en cuenta los directorios *img*, y *js* como se haría normalmente. Observamos también que el puerto seleccionado para su ejecución es el *8081*. Con la directriz *.build()* finalmente completamos la configuración del servidor.

Para el caso del servidor web que servirá el servicio de búsqueda su configuración se realizará de la siguiente manera:

```
WebSite serviceWebsite = new AssetsWebsite(mAssetManager, "services");

AndServer serviceServer = new AndServer.Build()

        .website(serviceWebsite)

        .port(8777)

        .registerHandler("findStreamers", new RequestUDPSearchHandler())

        .build();
```

En este caso la ruta que le ofrecemos al objeto *AssetManager* es la de *services*. El puerto configurado también es diferente pues es una instancia diferente del servidor siendo este el *8777*. También es incluida la directriz *.registerHandler* que nos permite registrar la *uri /findStreamers* de una petición a este servidor, con la ejecución de la clase *RequestUDPSearchHandler.java* que como anteriormente hemos explicado devolverá los resultados de la búsqueda.

Por último restaría inicializar ambos servidores, esto es posible ejecutando sobre ellos el método *start()* en ambos, del mismo modo el método *stop()* aplicado en ambos, los detendrá.

5.4.2 Implementación de la interfaz web

El otro gran bloque del proyecto es la interfaz web. Su objetivo es conformar una interfaz en la cual el usuario pueda cómodamente agrupar aquellos streamings o flujos que desee, y además, incluir diversas herramientas para el usuario.

La interfaz será implementada en lenguajes web, entre los cuales utilizaremos *HTML, CSS* y *Javascript*. Toda la funcionalidad de la interfaz será implementada con el lenguaje *Javascript*, el cuerpo y los elementos de la interfaz (o página) serán diseñados con el lenguaje de etiquetas *HTML* por último, la estilización de la misma será conseguida gracias al lenguaje *CSS*.

La interfaz poseerá cualidades provistas por librerías externas basadas en *javascript* que nos permitirán incluir funcionalidades a la página sin tener que partir de cero para construirlas, estas funcionalidades se resumen en las siguientes:

- La interfaz implementará un sistema de notificaciones flotantes para diferentes acciones que realiza el usuario en la misma. El sistema de notificaciones vendrá simplificado por la aportación de la librería *Noty.js*.
- Hay disponible un objeto *TimePicker* para la selección de horas y minutos, que resulta parte para la herramienta que permite realizar capturas a un streaming o flujo, en una determinada hora. Para ello haremos uso de la librería *timepicker.js* que simplificará la creación, utilización y creación de este objeto.
- La interfaz para la herramienta de edición de una imagen a través de filtros, incluye la opción de poder guardar el fichero como una imagen, para reducir el coste de trabajo y tiempo, se recurre a una librería que aporta el código de una manera simple y eficiente. *FileSaver.js* será la librería que aporte la solución para el guardado de ficheros y de esta forma, transportar su funcionalidad a un simple botón.
- La herramienta de edición de imágenes a través de filtros, se apoya en la funcionalidad dada por la librería *fabric.js*, diseñada y dedicada para la manipulación de imágenes, que resulta una gran base sobre la que basarse para desarrollar la herramienta.
- Cuando la búsqueda de ips es utilizada en la interfaz y es recuperado el resultado, dichas ip son un texto copiable, para darle más usabilidad a la interfaz, se implementa un botón que permita directamente copiar la ip seleccionada, simplificando así el proceso. Para ello se hace uso de la librería *clipboard.js* que permite el uso del clipboard del ordenador para la copia de textos a través de diversos elementos.
- Por último haremos un uso genérico de la librería *jQuery*, este *framework* nos permite simplificar el uso de *javascript* ya sea a la hora de manejar los elementos *HTML* o de manipular sencillamente los eventos, animaciones y peticiones *Ajax* que se produzcan. Su objetivo principal, será la simplificación del código usado y su entendimiento una vez escrito.

Un aspecto importante de la interfaz web, será su estilización ya que será en gran parte responsable de realizar un diseño amigable y simple para el usuario. El lenguaje web utilizado para realizarla es *CSS* y a través de él podremos manipular las diferentes propiedades de los elementos *HTML* que existan en la página, estos pueden ser las dimensiones del objeto, su color, fuentes de texto, situación dentro del documento *HTLM*, márgenes, paddings y un sin fin de opciones entre las que podemos escoger.

Para trabajar juntamente con el lenguaje *CSS* incluiremos las librería *bootstrap.css* y *bootstrap.js* en su versión 4, siendo ambas potentes librerías para el diseño y estilización web, que sobretodo trabajan con la aplicación de clases sobre los objetos *HTML* entre otras grandes utilidades como el uso de elementos ya prediseñados y estilizados como por ejemplo barras de progreso o gráficos, considerados como *jQuery Plugins*. *Bootstrap* permitirá implementar el diseño de cuadrícula para el documento *HTML*, de forma que el espacio puede ser seccionado en diferentes columnas y por tanto, realizar diseños para el resto de tamaños de pantalla proporcionando así un diseño *responsive* de fácil implementación, que como ya se ha indicado podrá verse para diferentes tamaños de pantalla de forma que el espacio utilizado utilizado por la interfaz se adapta al tamaño de pantalla en el cual es visualizado.

Por último, es necesario mencionar la lógica de negocio de la interfaz que estará escrita en lenguaje *javascript*, y a través de ella el usuario podrá interactuar con la interfaz web utilizando las diferentes herramientas que ofrece, por ello, se está modelando en el lado del cliente conformando así un *FatClient*. Aunque anteriormente ya hemos mencionado las librerías que han dado apoyo a diversas funciones, también es necesario analizar el resto de documentos *javascript* creados e incluidos y que conformarán dicha lógica de negocio. Por tanto, a continuación detallamos de que documentos se tratan, y que funciones poseen.

Los documentos que conformarán la lógica de negocio son: *animations.js*, *camGenerator.js*, *canvasDiffTool.js*, *captureFrameInTime.js*, *fabricCanvasTools.js*, *lastFrameCapture.js*, *logGenerator.js*, *mjpeg.js*, *myNotyMessages.js*, *reachable.js*, *streaming.js* y *temporizedControls.js*. Se analizarán individualmente para observar que funciones componen cada fichero, y que trabajo desempeña cada función.

- ***animations.js***

Este fichero *javascript* tiene como objetivo aplicar animaciones para la apertura y cierre de los menús situados en el lateral izquierdo de la interfaz web, la función que implementa es *jQueryResize()*.

➔ *jQueryResize()* - Esta función realiza el cambio de las animaciones de los menús, dependiendo del tamaño de pantalla del cliente, ya que para ciertos tamaños los valores de las animaciones son diferentes. Esta función es utilizada dentro del evento *resize* del navegador y en la carga de la interfaz (o página) web.

- ***camGenerator.js***

camGenerator.js tiene como objetivo la creación de los contenedores para visualizar los streams o flujos de imágenes, además implementa diversas herramientas para dichos contenedores. Las funciones que contiene son *camFieldGenerator()*, *deleteCameraField()*, *renameField()*, *canvasRefresh()* y *removeAllCameras()*.

➔ *camFieldGenerator()* - Esta función crea un contenedor para poder visualizar un stream o flujo de imágenes, y con él todos los campos necesarios para introducir la dirección ip, renombrarlo, eliminarlo o realizar una captura, su ejecución está ligada al evento clic del botón *add camera* situado en la interfaz web.
➔ *deleteCameraField()* - Es utilizado cuando se pulsa sobre el botón *delete* de unos de los contenedores y su función es eliminar el contenedor al cual pertenece el botón.
➔ *renameField()* - Esta función renombrará el label que posee el contenedor. Es ejecutada pulsando sobre el botón *rename* del contenedor seleccionado.
➔ *canvasRefresh()* - El contenedor contendrá un elemento *canvas* a través del cual se visualizarán las imágenes del stream o flujo de

imágenes y esta función simplemente limpiará el canvas perteneciente al contenedor en el cual se haya detenido dicho stream o flujo.

➔ *removeAllCameras()* - Permitirá la eliminación de todos los contenedores existentes creados, previamente se le mostrará un diálogo al usuario preguntando si realmente quiere llevar a cabo la acción. Si la acción es realizada será mostrada una notificación al usuario despues de borrar todos los contenedores.

- ***canvasDiffTool.js***

Este fichero implementa la herramienta de diferencia entre imágenes disponible en la interfaz. Se compone de las funciones *handleDragEnter(), handleDragOver(), handleDrop(), handleFiles(), canvasDifferenceTool(), resetDiffTool(), restoreOriginalDiff(), diffColorChange()* y *thumbnailer()*.

➔ El conjunto de funciones *handleDragEnter, handleDragOver, handleDrop y handleFiles* es utilizado para poder utilizar el elemento *FileReader* para adjuntar las imágenes a la herramienta mediante el evento *drag and drop*.

➔ *canvasDifferenceTool()* - Es la función que calculará la diferencia entre las imágenes dadas y posteriormente dibujará el resultado en un nuevo canvas. Deshabilitará el botón de cálculo de la diferencia puesto que ya se ha realizado e incluirá una serie de opciones para modificar el resultado obtenido.

➔ *resetDiffTool()* - Devolverá la herramienta de diferencia entre imágenes a su estado original, es decir, reiniciará la herramienta para un nuevo uso.

➔ *diffColorChange()* - Permitirá aplicar sobre el resultado obtenido de la diferencia un filtro de color, tornando de esta manera el resultado a modo monocromático.

➔ *restoreOriginalDiff()* - Recalcula la diferencia original para el caso en que se haya aplicado un filtro monocromático sobre el resultado de la diferencia.

➔ *thumbnailer()* - Esta función permite a la imagen que es insertada mediante el evento *drag and drop* que ocupe el el tamaño del canvas sin modificar la escala a la que se encuentra la imagen para mantener la calidad de la misma.

- ***captureFrameInTime.js***

Con este fichero se implementan las capturas a los streams o flujos de imágenes de forma temporizada, es decir, tras la llegada de una hora seleccionada por el usuario, la funciones que actúan en este fichero son *miliseconds(), timedFrames(), waitForTime() y executeTimedSnapshots()*.

➔ *miliseconds()* - Simplemente realizará la conversión de horas, minutos y segundos dados a milisegundos.

➔ *timedFrames()* - Esta función se centra en la comprobación de estado de los diferentes elementos en caso de que alguno falte o falle, de este

modo si alguna configuración necesaria falta, se le avisa al usuario mediante una notificación. En caso de que todo sea correcto entonces se procede a inicializar el método *waitForTime* que esperará el tiempo indicado por el usuario.

- ➔ *waitForTime()* - Esta función se compone de un *timeOut* que esperará a lo largo de cinco segundos, pasado ese tiempo se reducirán esos segundos al tiempo establecido por el usuario, de forma que hasta que este no es cero o menor no es ejecutada la función que finalmente realizará las capturas.
- ➔ *executeTimedSnapshots()* - Esta función es la encargada de realizar las capturas del stream o flujo de imágenes si todo ha salido bien y el tiempo establecido por el usuario ha llegado a su fin, de este modo se realiza una copia del canvas donde se visualiza el stream o flujo de imágenes a un nuevo *canvas* creado, a su vez este es añadido en diversos menús que actuarán de galerías para la posterior visualización por el usuario si este lo desea.

- ***fabricCanvasTools.js***

Este fichero incluye las utilidades que implementan la herramienta de edición de imagen con filtros en la interfaz web. Incluirán las líneas necesarias para interactuar con la librería *fabric.js* y de este modo aprovechar los recursos que ésta ofrece como los filtros utilizados. Analizamos de este fichero únicamente la función *fCanvasSaveImage()* puesto que el resto son funciones aportadas también por la documentación de la propia página de la librería y que son utilizadas para la utilización de los propios filtros, así como los eventos recogidos durante todo el fichero, que son la descripción de la interactividad entre los selectores *html* de la interfaz web y los filtros implementados en *javascript*.

- ➔ *fCanvasSaveImage()* - Permitirá al botón *save image* incluido en la herramienta la capacidad de descargar la imagen resultante en el equipo del usuario si así lo decide.

- ***lastFrameCapture.js***

Este fichero *javascript* implementa la funcionalidad del botón *capture* situado en cada contenedor creado, y su objetivo es realizar una instantánea del stream o flujo de imágenes que se está visualizando en el contenedor al que pertenece. Posee una sola función la cual es *lastFrame()*.

- ➔ *lastFrame()* - Fabricará un *canvas* como un objeto *javascript* en el cual se realizará una copia del *cavas* del contenedor en el que se esté visualizando el stream o flujo de datos, posteriormente el *canvas* creado anteriormente es añadido a los menús situados en el lateral izquierdo ya que estos actuarán de galería si el usuario decide consultar las capturas realizadas.

- ***logGenerator.js***

Implementa las funciones necesarias para interactuar con el log de eventos incluido en la interfaz web, consta de dos funciones: *logPrint()* y *logClean()*.

➔ *logPrint()* - Dado un texto, el objetivo de esta función es pintar dicho texto dentro del elemento prediseñado para actuar como log en el documento *HTML*, incluyendo horas y minutos actuales.

➔ *logClean()* - Esta función permitirá limpiar el log de eventos de la interfaz web y posteriormente se avisará al usuario mediante una notificación.

- ***mjpeg.js***

Este documento no contiene funciones ya que sus dos elementos consisten en dos módulos diferentes, los cuales son *stream y* player. El módulo *stream* proporcionará el stream o flujo de imágenes de forma que asigna la dirección indicada al contenedor, a un elemento imagen en *javascript*, esto es realizado continuamente por lo que los datos llevados al elemento imagen son los distintos frames que ofrece el dispositivo a través de la aplicación. Los diferentes elementos imagen de *javascript* son utilizados por el módulo *player* que pinta continuamente estos elementos sobre el canvas contenido en el contenedor al cual pertenece y por tanto generando la apariencia de vídeo, siendo en realidad una secuencia de imágenes pintadas rápidamente una tras otra.

- ***myNotyMessages.js***

Como anteriormente hemos visto se ha incluido una librería denominada *Noty.js* que nos aporta una serie de notificaciones para su visualización por el usuario en la interfaz web, este fichero *javascript* simplemente actúa como una interfaz para fácilmente fabricar dichas notificaciones de manera más simple, su única función es *notifyMe().*

➔ *notifyMe()* - Esta función tiene como objetivo la simplificación de la creación de una notificación de la librería *Noty.js*. Esta función recoge como argumentos, el mensaje deseado para la notificación, el tipo de notificación, su posición en el documento y su duración de visualización, después de recoger esta información internamente posee el constructor de notificaciones de la librería y únicamente son utilizados dichos argumentos dentro del constructor.

- ***reachable.js***

Este fichero *javascript* realizará la comunicación por *ajax* con el servidor de servicios iniciado en el dispositivo móvil y recuperar el resultado de la búsqueda de *ips* realizado a través de un *broadcast UDP*. Implementa la función *UDPBroadcastSearch()*.

➔ *UDPBroadcastSearch()* - Esta función realiza una petición *GET* mediante *ajax* a la uri especificada */findStreamers*. Si la operación es

exitosa y es devuelto un resultado, estos son separados e incluidos dentro del recuadro inferior al botón *find ips*. En caso contrario se asumirá un fallo de comunicación con el servidor y se le informará al usuario mediante una notificación. Si el resultado es exitoso o si este no contiene nada puesto que no hay resultados de búsqueda, también es informado el usuario mediante una notificación. Esta función está ligada al evento clic de botón *found ips* anteriormente mencionado, este botón es deshabilitado durante la espera del resultado de la operación para evitar de esta forma un *spam* de peticiones al servidor.

- ***streaming.js***

Este fichero *javascript* actúa de intermediario entre la interfaz y el fichero *mjpeg.js*, incluye dos funciones: *startStream()* y *verifyIP()*.

- ➔ *startStream()* - Es ejecutado en el evento clic del objeto *canvas* de un contenedor, crea una instancia del módulo *player* del fichero *mjpeg.js* y le pasa la referencia del *canvas* contenido en el contenedor, ya que sobre este será visualizado el stream o flujo de imágenes, es verificada la dirección del dispositivo indicada en el contenedor, que también será el segundo argumento requerido para el módulo *player* ya que de esta dirección serán obtenidas las imágenes. Si el usuario inserta incorrectamente la dirección del dispositivo el sistema informa al usuario mediante una notificación, también si el dispositivo seleccionado supera el tiempo de espera y no se ha recibido respuesta del mismo.
- ➔ *verifyIP()* - El objetivo de esta función es comprobar que la IP indicada del usuario es correcta, es decir, que no corresponde a direcciones especiales incluida la de *loopback* o a direcciones erróneas.

- ***temporizedControls.js***

Este documento *javascript* pretende realizar un control sobre los parámetros de entrada para la herramienta de capturas en el tiempo. Es inicializada en el evento *ready* del documento debido a que acudimos a diferentes eventos de ciertos elementos *HTML* que en el momento del procesamiento del código pueden no haber sido creados, y por tanto dar un error de referencia no existente. No posee funciones, y es inicializado el objeto *TimePicker* para su uso. Es comprobado el valor de los elementos que controlan la cantidad de capturas que se van a realizar y el intervalo entre estas, no permitiendo de esta forma que se coloque un valor inferior o superior al permitido.

6. Pruebas de ejecución

Para probar nuestra aplicación y comprobar que se cumplen los requisitos de la misma, se realizan una serie de pruebas donde se demuestra el correcto funcionamiento de la misma, que la obtención de la dirección ip del dispositivo es correcta, que el streaming o flujo de imágenes se ejecuta correctamente, que los servidores web son inicializados correctamente para servir la interfaz web y el servicio y por último las funcionalidades de la propia interfaz web, donde con todo ello se observa el correcto funcionamiento de la aplicación.

Se tratará de realizar capturas de pantalla a las diferentes secciones para confirmar todo lo anterior mencionado e incluiremos un pantallazo de la captura del tráfico de la red para poder observar los paquetes que corresponden con las imágenes enviadas, siendo esta prueba la que efectivamente acaba por comprobar que todo es realizado correctamente.

6.1 Observación del funcionamiento de la aplicación android y la interfaz web.

A continuación observaremos una serie de capturas referentes a la aplicación android y la interfaz web. A través de ellas podremos vislumbrar el funcionamiento de la misma y comprobaremos que todo es realizado correctamente.

6.1.1 Imágenes funcionales de la aplicación android

La siguiente captura corresponde con la inicialización de la aplicación donde podremos comprobar que la dirección ip del dispositivo es obtenida y visualizada correctamente:

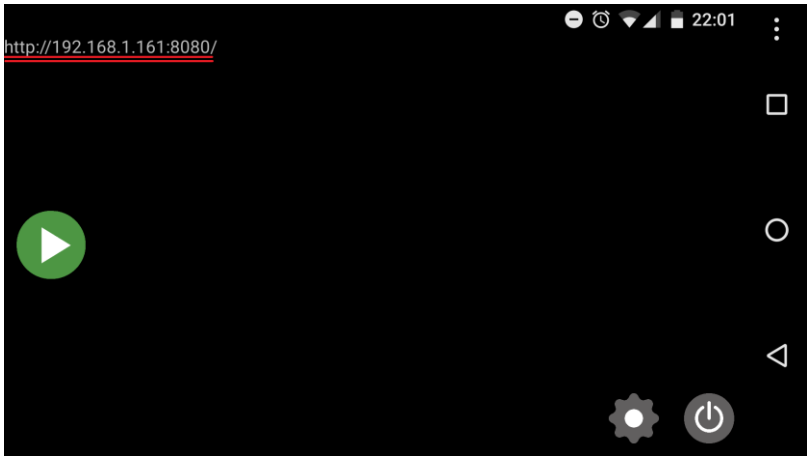


Ilustración 33

Como se aprecia en la siguiente imagen la ip que ofrece la aplicación correspondiente al dispositivo es idéntica a la que podemos consultar en la información del propio dispositivo.

Ilustración 34

Seguidamente observaremos que los servidores web son instanciados a través del botón de encendido gris, tornando el botón de color verde indicando que estos están activos, y mostrando un diálogo que remite la dirección a través de la cual el usuario puede acceder a la interfaz web. El servidor que ofrece el servicio no es sugerido al usuario puesto que como anteriormente se ha mencionado se pretende que posea cierta invisibilidad hacia el usuario.

Ilustración 35

A través de un cliente, como el navegador web del usuario, podremos acceder a la interfaz web insertando la dirección mostrada en el diálogo de la aplicación en la barra de direcciones del navegador como es mostrado a continuación:

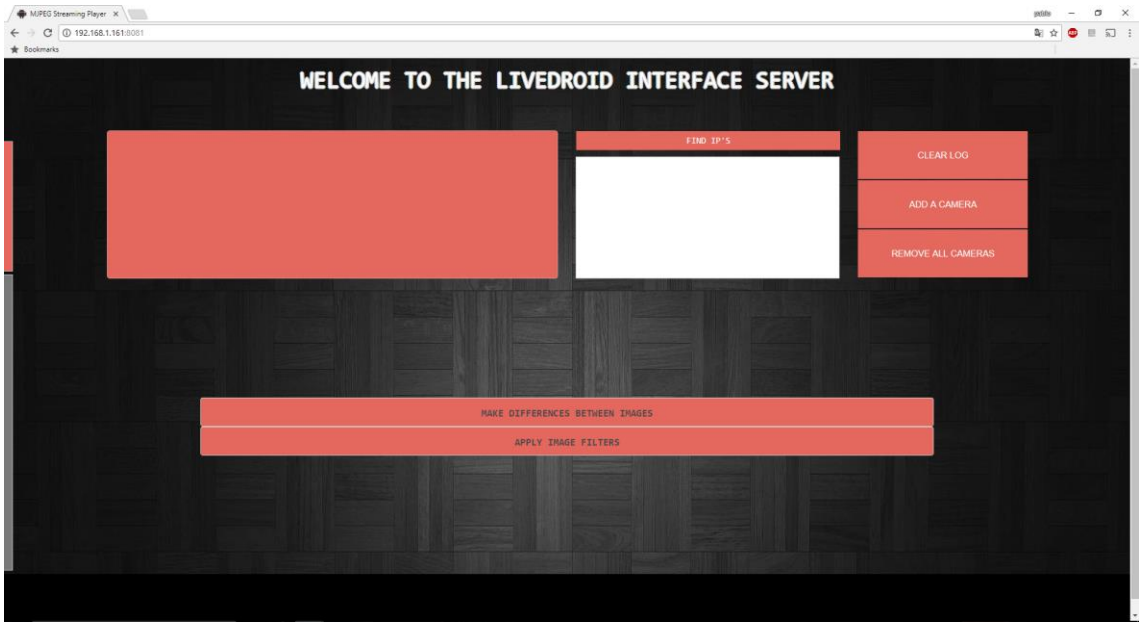


Ilustración 36

Cuando el servidor es desactivado mediante el botón de encendido verde, es notificado al usuario mediante un diálogo la acción realizada, quedando el acceso evidentemente inviable:

Ilustración 37

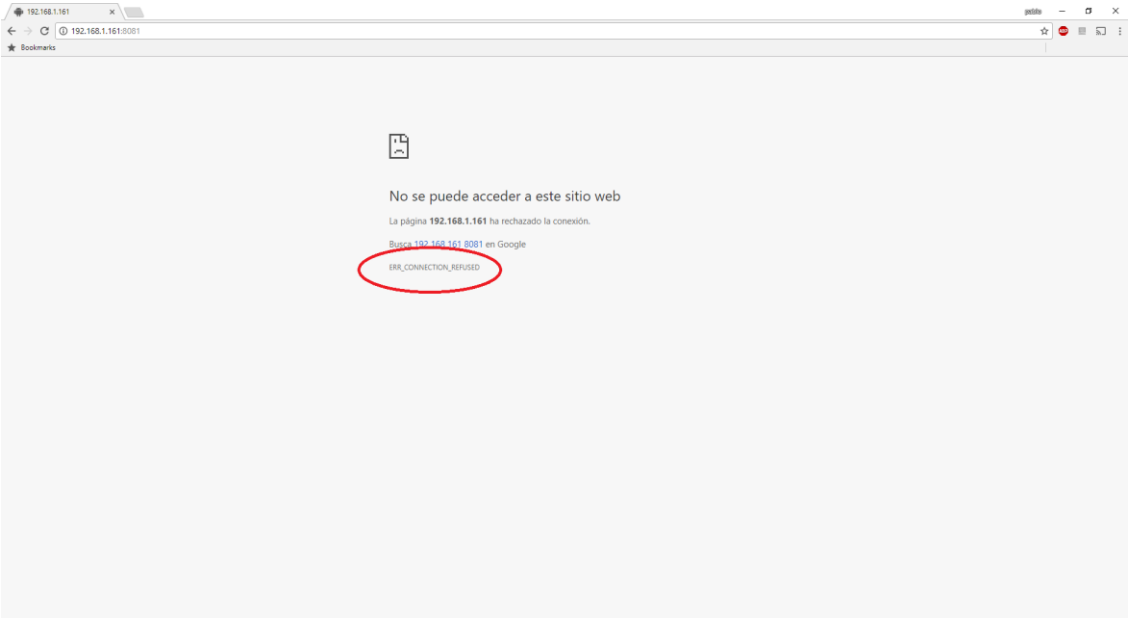


Ilustración 38

Cuando es iniciado el streaming o flujo de imágenes de la aplicación, el elemento *SurfaceView* incorporado en el layout de la actividad principal, mostrará las imágenes de la cámara creando así un *preview* de la cámara, es decir, las imágenes previas que se producirían durante una fotografía o la grabación de un vídeo, pero en este caso, corresponden con las imágenes que ya se encuentran en emisión a través de la red de área local. El ejemplo es ilustrado en la siguiente imagen:

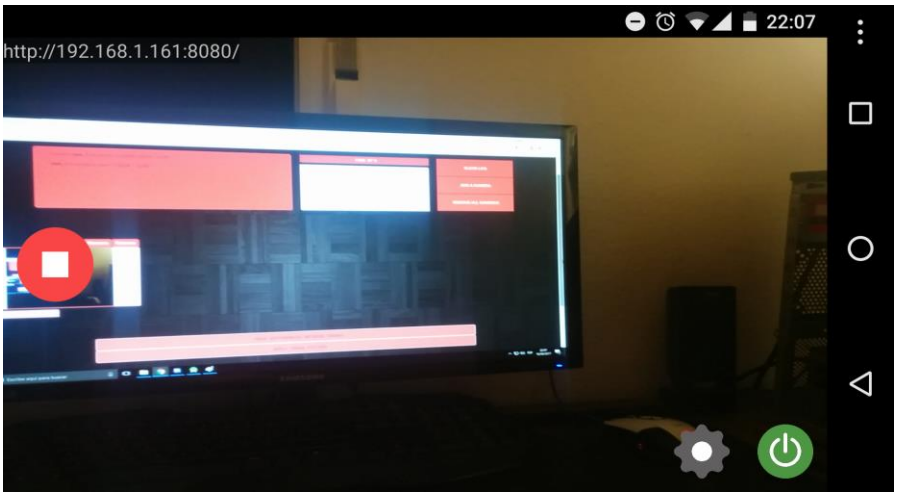


Ilustración 39

El estado de la imagen anterior es de la cámara del dispositivo realizando el streaming o lujo de imágenes por lo que mediante la interfaz web podríamos visualizarlo accediendo a su dirección ip como se indica en la siguiente ilustración:

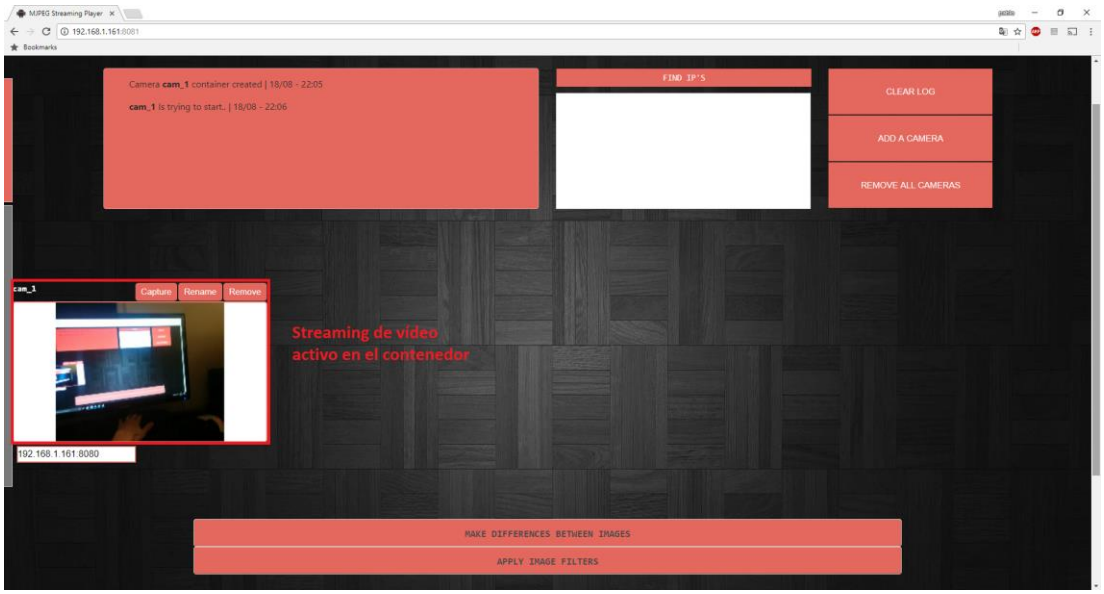


Ilustración 40

6.1.2 Imágenes funcionales de la interfaz web

A continuación en este apartado incluiremos capturas sobre diferentes secciones de la interfaz web, utilizando las opciones más remarcables que ofrece echando una vista sobre el menú que actúa de galería para las capturas, la herramienta de capturas temporizadas y las herramientas para la diferencia y edición de imágenes.

En primer lugar visualizamos en la siguiente imagen el menú para ejecutar la herramienta de las capturas temporizadas, como podemos comprobar hay una serie de selectores y secciones de entrada para configurar dicha herramienta, una vez es todo seleccionado correctamente, la herramienta es puesta en marcha:

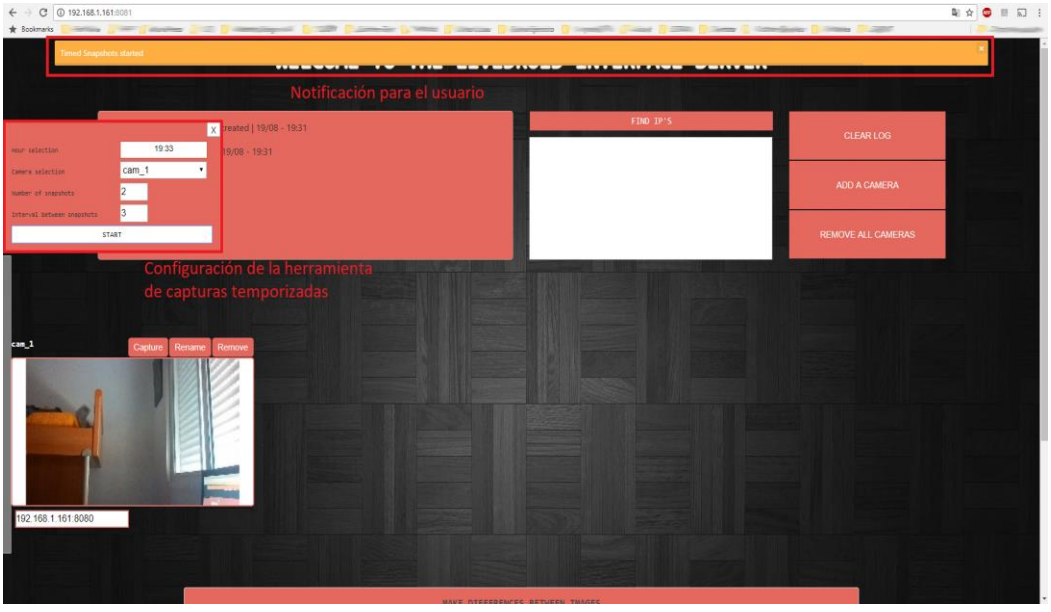


Ilustración 41

También podemos comprobar cómo a través de la consola de *javascript* del navegador es obtenido el tiempo restante hasta poder realizar la captura:

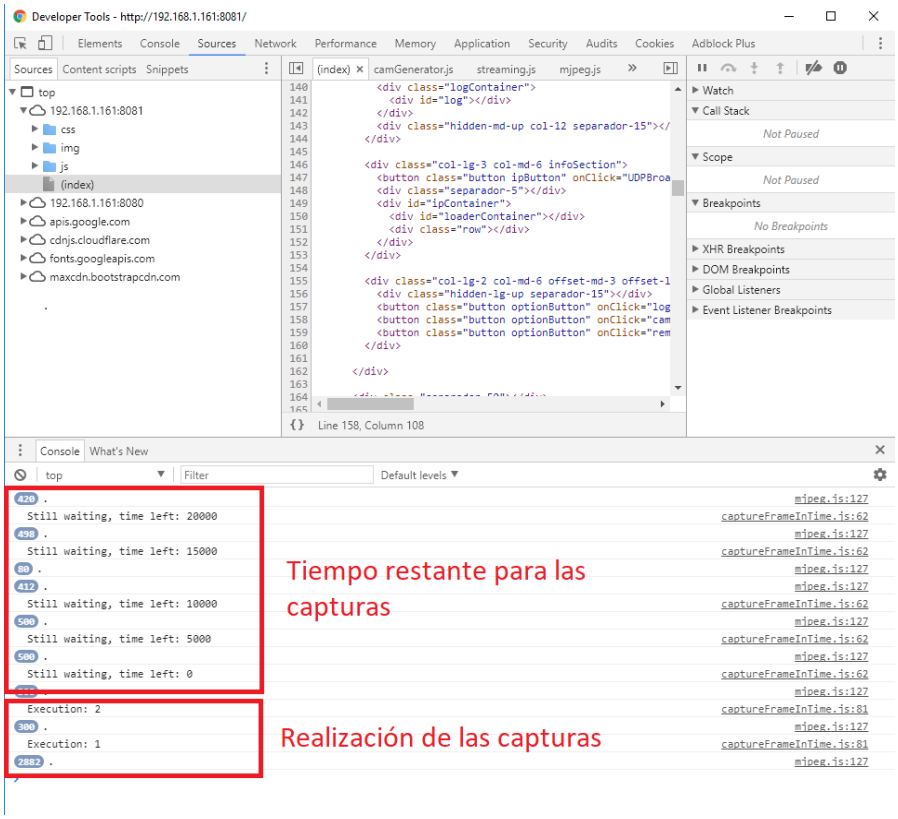


Ilustración 42

Cuando se realizan capturas, todas ellas son almacenadas en el menú en el lateral izquierdo, de forma que estas quedan allí para una posterior consulta por parte del usuario:

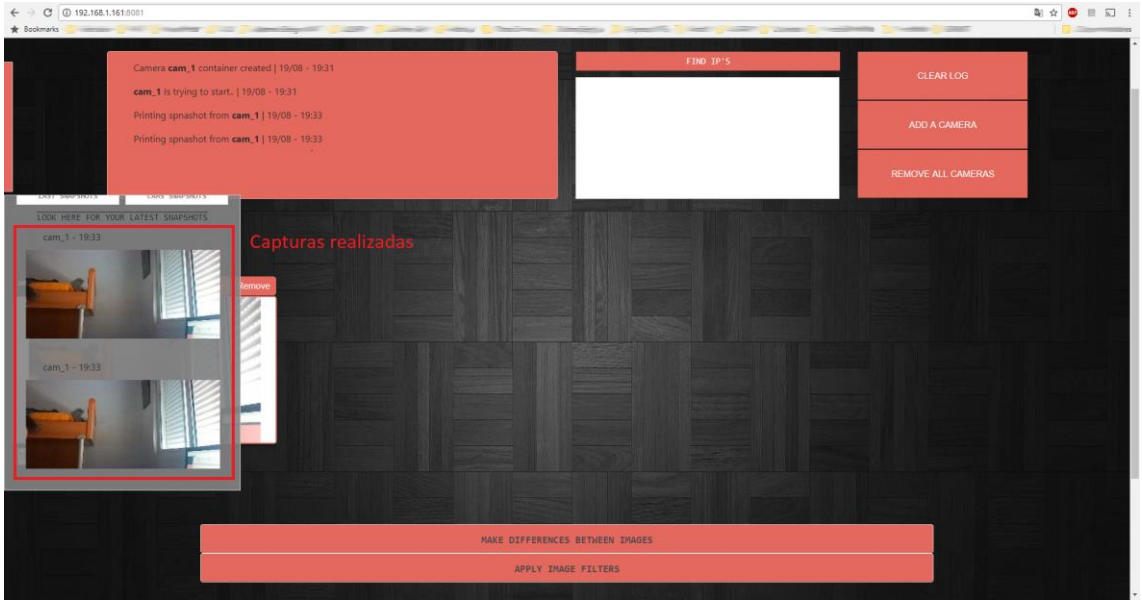


Ilustración 43

A continuación observamos la herramienta para obtener las diferencias entre dos imágenes dadas, las imágenes utilizadas serán ambas capturas obtenidas a partir de un stream o flujo de imágenes activo y posteriormente descargadas:

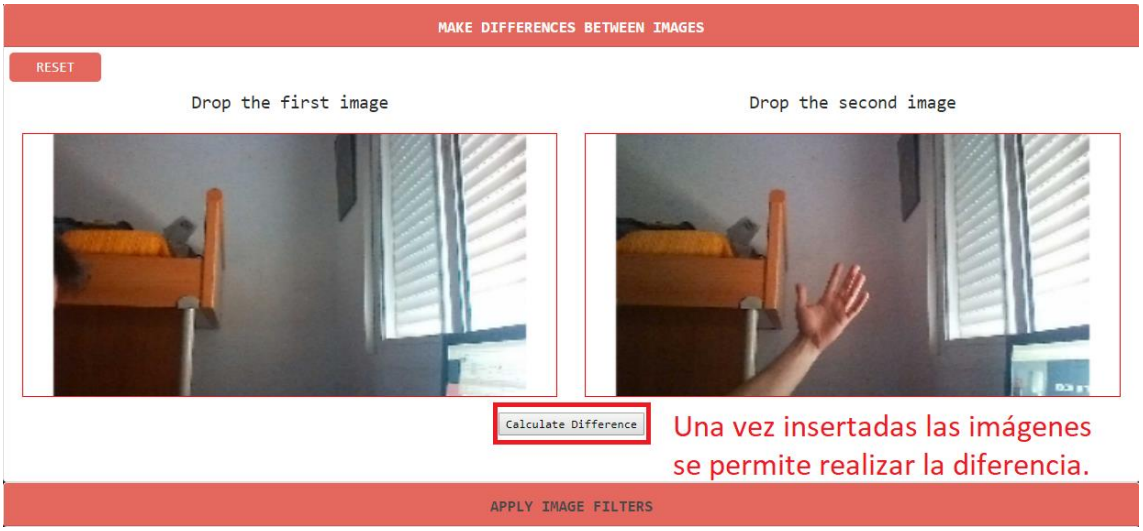


Ilustración 44

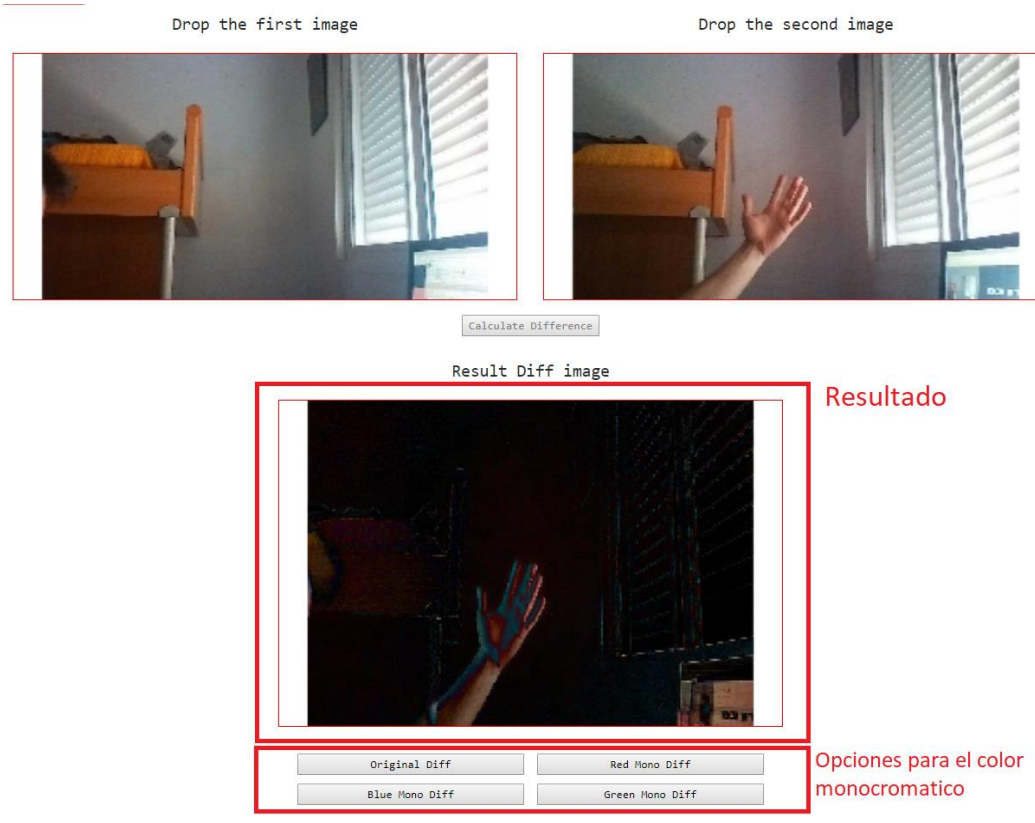


Ilustración 45

Por último se muestra en la siguiente imagen la herramienta de edición de imagen mediante filtros, aplicando en este caso un sencillo filtro de inversión del color:

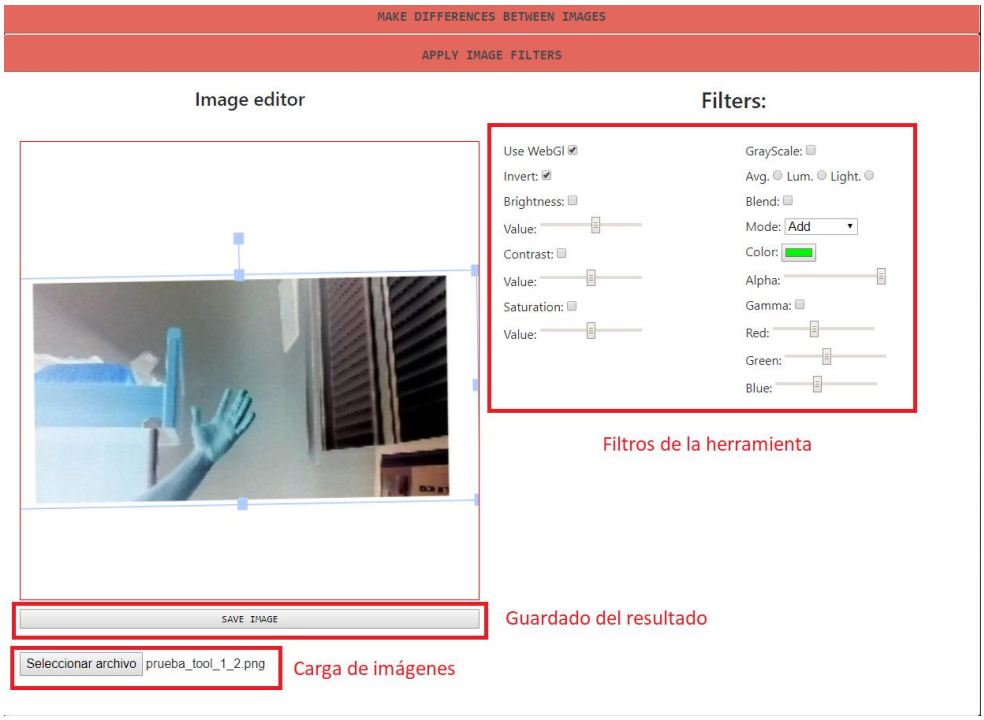


Ilustración 46

6.2 Prueba de tráfico en la red de área local

Realizaremos una captura del tráfico de la red mediante la herramienta software gratuita *WireShark*, y realizaremos un filtro de los paquetes recibidos para poder claramente los paquetes que provienen del dispositivo y que se corresponden con las imágenes enviadas de su cámara, se puede observar con la ilustración de la siguiente página:

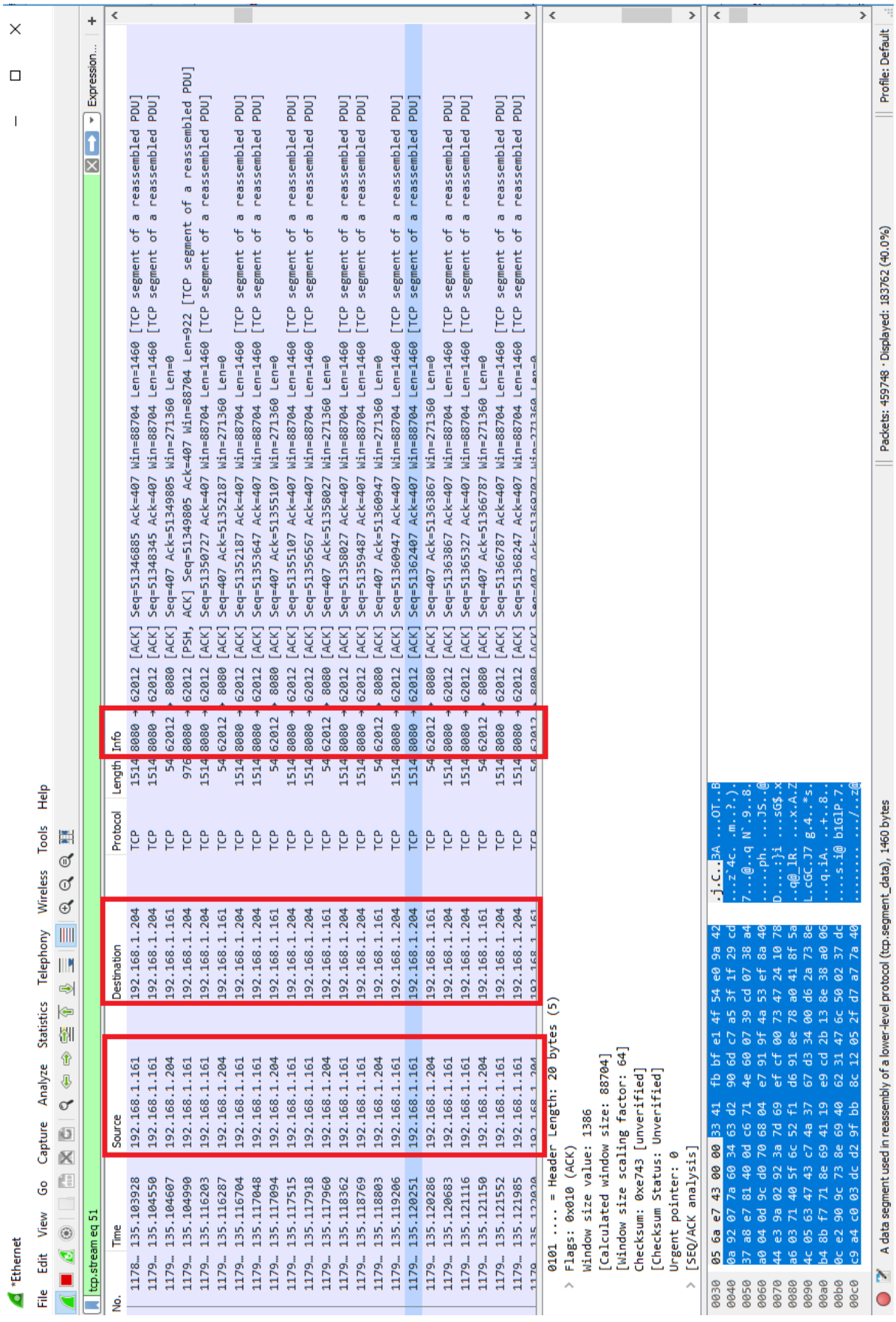


Ilustración 47

Por último mostramos ilustramos las cabeceras *HTTP* de los paquetes recibidos por la dirección 192.168.1.161, es decir, el dispositivo móvil, realizando un seguimiento sobre uno de los paquetes enviados por el mismo:

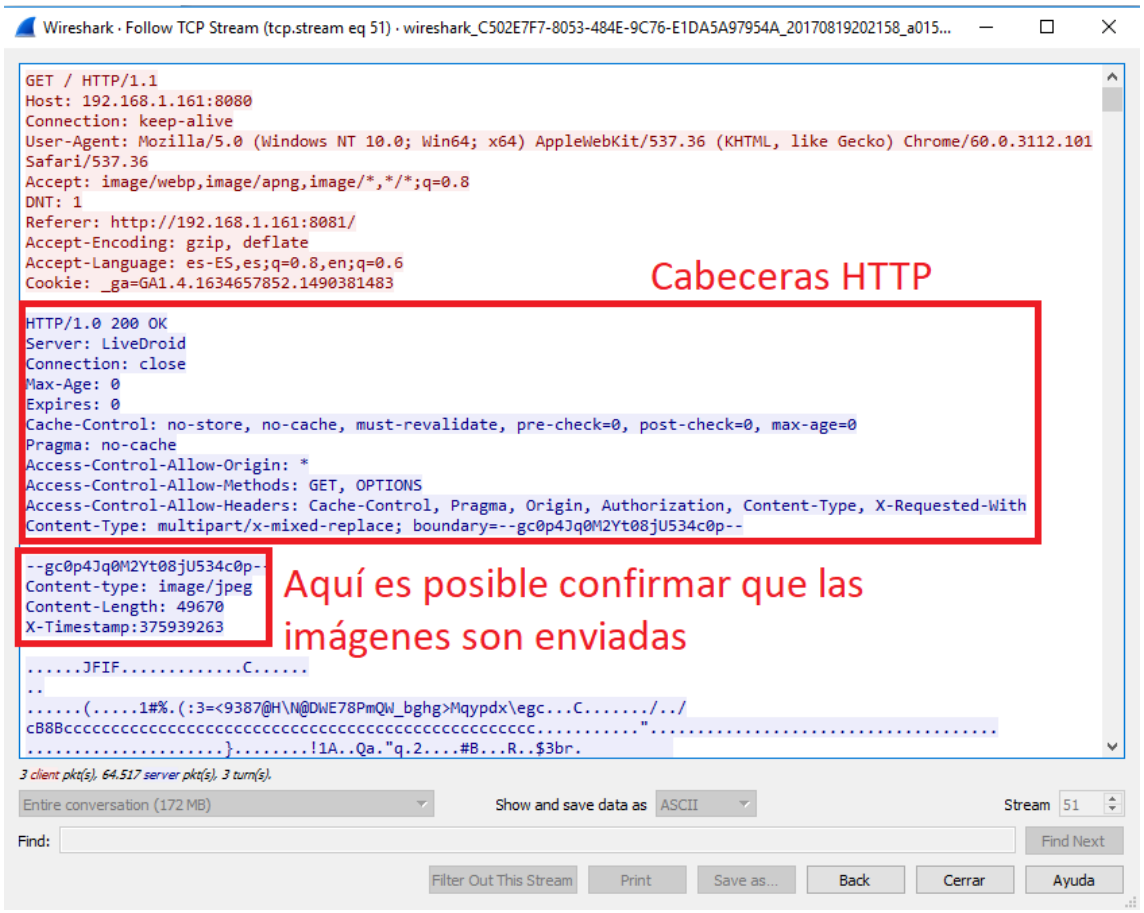


Ilustración 48

7. Conclusiones y trabajo futuro

El proyecto realizado ha sido la implementación de una aplicación android que permite la transmisión de imágenes de la cámara del dispositivo móvil, conformando de esta manera un stream o flujo de imágenes con formato *MJPEG*, es decir, la reproducción de la secuencia de imágenes enviadas por el dispositivo, todo ello junto a una interfaz web, desarrollada en lenguajes web y hosteada en una instancia de un servidor web alojada en la propia aplicación, es decir, en el dispositivo móvil, para ser abierta en un navegador y que ofrece herramientas para visualizar un número indeterminado de streams o flujos de imágenes de los diferentes dispositivos en los que se ha instalado la aplicación y se ha iniciado dicha retransmisión así como una serie de herramientas para interactuar con dichos flujos de imágenes.

Haciendo una valoración del proyecto realizado, es posible concluir el trabajo que se ha realizado bien y aquél que es mejorable (desde un propio punto de vista). Para todas aquellas posibles mejoras se ha decidido incluir un apartado llamado *Trabajo futuro* que recogerá las posibles mejoras en el proyecto. Para las conclusiones restantes, se dedican el resto de apartados de este punto.

7.1 Conclusiones

Dadas las revisiones realizadas en el punto anterior, es posible decir que el proyecto se encuentra en posición de afirmar que cumple con las especificaciones y requerimientos iniciales. Y por otra parte los objetivos personales marcados que son el aprendizaje en el desarrollo de aplicaciones móviles y aumentar la profundidad en el conocimiento de los lenguajes web, se dedican los siguientes apartados.

7.1.1 Aprendizaje durante el desarrollo del proyecto

El caso personal es el de un alumno cursando el grado en ingeniería informática que nunca había cursado asignaturas que introduzcan a la programación móvil, es decir, al desarrollo de aplicaciones móviles, por tanto el punto de partida ha sido un nivel de escasos conocimientos dentro de este entorno de programación, básicamente el nivel más principiante dado que el lenguaje empleado para la programación es basado en *Java*.

Dada la elección del entorno de desarrollo de *Android* el desarrollo del proyecto se ha facilitado notablemente ya que como hemos comentado anteriormente el lenguaje base de este entorno es *Java* el cual ya se aprendió en previos cursos durante los años del grado en ingeniería informática.

La evolución de los conocimientos adquiridos se concentran desde el desarrollo de aplicación más básica y típica como el *Hola mundo* hasta el proyecto que se ha ido describiendo durante la memoria y que se ha desarrollado. Esta aglomeración de conocimientos han sido adquiridos desde diversas fuentes como libros especializados, foros especializados, páginas web, etc., llegando al punto en el que se asume que se poseen las bases necesarias para poder desarrollar aplicaciones de nivel medio.

Por otra parte también se ha realizado una ampliación sobre los lenguajes web. En este punto si se poseían conocimientos de estos lenguajes gracias a algunas asignaturas cursadas en cursos anteriores del grado en ingeniería informática, y sobretodo gracias a las diferentes prácticas de empresa realizadas durante diversos años, asistiendo a estas como trabajador frontend de una plataforma web. Se han conseguido ampliar los conocimientos gracias a la implementación de librerías de las que no se tenía conocimiento como *fabric* o *noty*, además del uso de la última versión de *bootstrap* que incluye novedosos e interesantes cambios frente a sus predecesores. También se ha atesorado conocimiento sobre el elemento *canvas* y muchas de sus utilidades y funciones.

7.1.2 Entornos de plataformas y aplicaciones móviles

Para poder realizar una elección sobre la plataforma y las tecnologías empleadas para el proyecto, ha sido necesario emplear un tiempo para investigar la combinación entre las tendencia actual y la situación para seleccionar la más apropiada.

Las decisiones seleccionadas el argumento de mayor importancia siempre ha sido las opciones más populares, gratuitas y sobretodo funcionales en el mayor número de dispositivos. Conociendo la tendencia de los últimos años en el mercado de los smartphones que se corresponde con el cambio de terminal cada cierto número de años generalmente, es de lógica pensar que la gran mayoría de usuarios poseerán terminales que han caído en desuso y que probablemente todavía conserven. Respecto a la selección de la plataforma, se ha decantado por Android ya que es la que tiene un mayor número de usuarios, y es la plataforma en la que la fragmentación de forma aislada va a ser beneficiosa, ya que se encontrarán en ella una gran variedad de dispositivos económicos y en posesión de una gran cantidad de usuarios, permitiendo desarrollar el potencial de la aplicación debido al aumento de dispositivos en los que se puede instalar y siendo por tanto, los mismos, con utilidad ya que cada uno actúa como nodo de emisión de imágenes.

Por último otra de las conclusiones que se trata de mostrar en el apartado es la importancia que ha tenido el tiempo utilizado, ya que ha permitido actualizar los conceptos referentes al proyecto dentro del constantemente cambiante ámbito como es de la informática.

7.2 Trabajo futuro

El trabajo desarrollado en el proyecto es valorado positivamente, no obstante existen ciertos aspectos mejorables e incluso ampliables dentro del mismo dados a mejorar o ampliar la funcionalidad. A continuación se describen una serie de ideas que pueden resultar propicias para realizar mejoras al proyecto:

- **Desarrollo multiplataforma.** Ya que se ha tenido en cuenta que el desarrollo de la interfaz web ha estado basado para el navegador google chrome, cabría la seguridad de expandir el desarrollo de la misma hacia otros navegadores de gran uso entre los usuarios como el *mozilla firefox* o las últimas versiones del *internet explorer* incluido *microsoft edge*.
- **Habilitar la detección de movimiento en tiempo real.** Debido a que se partía desde un nivel muy básico, es difícil completar una política que respete el *cross-origin* que se produce cuando las imágenes son transmitidas desde el terminal hasta el cliente, necesario para poder utilizar el recurso de la imágen de forma ilimitada ya que sin este proceso las opciones para el tratamiento de dichas imágenes son muy limitadas. Respetando este proceso el sistema sería capaz de obtener cualquier información de las imágenes e implementar un sistema de detección en tiempo real en el lado del cliente, es decir, en el navegador para evitar esa pesada carga al terminal y perjudicar la eficacia del stream o flujo de datos, de la misma forma se obtendría un sistema que abriría las puertas a una gran cantidad de opciones y novedades interesantes.
- **Mejorar la interfaz para poseer información persistente.** La interfaz web no posee mecanismos para mantener un estado de la información, sino que cualquier objeto creado en la misma es volátil si el usuario decide recargar la interfaz web, es decir, la página servida. Sería interesante introducir algún sistema de bases de datos, siendo a su vez interesante una solución para utilizar el *LocalStorage* de los navegadores sin tener que incluir una capa más para el almacenamiento de datos, sino una lógica de negocio que pueda implementar el comportamiento deseado.
- **Capacidad de grabación.** Puesto que el sistema solo permite ver un stream o flujo y realizar capturas sobre el mismo, sería interesante establecer un sistema que permita realizar una grabación del mismo y posteriormente dar la posibilidad al usuario de poder descargarlo en su equipo.

Por lo general el desarrollo realizado en este proyecto de final de grado a resultado satisfactorio y ha aportado una experiencia muy enriquecedora, ya que ha sido muy útil para el aprendizaje y la puesta en práctica de diversas tecnologías de programación con las que anteriormente no había habido contacto.

8. Referencias

Durante diversos puntos en la memoria se han mostrado claves a modo de referencias para diversos conceptos. En este punto se definirán dichas referencias y se añadirán las fuentes de las mismas para su consulta.

[ANDRO4ALL] Cifras generales sobre android (descargas y uso) play store

https://andro4all.com/2017/01/cifras-datos-curiosos-google-play-store-2016

[STATISTA] Número de aplicaciones en la play store de Google

https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[SOURCEANDROID] Sistema Operativo Android

https://source.android.com/

[ANDRO4ALLDIST] Distribución de las versiones del sistema operativo android

https://andro4all.com/2016/02/distribucion-android-enero-marshmallow

[iOS10] Lema de Apple iOS 10

https://www.apple.com/es/ios/ios-10

[RFC1889] Propuesta de estándar 1889 para protocolos de transporte en tiempo real

https://tools.ietf.org/html/rfc1889

[RFC3550] Propuesta de estándar 3550 para protocolos de transporte en tiempo real

https://tools.ietf.org/html/rfc3550

[RFC2326] Estándar 2326 para protocolos de transmisión en tiempo real

https://www.ietf.org/rfc/rfc2326

[FRExt] Extensiones del códec de compresión H.264

http://ip.hhi.de/imagecom_G1/assets/pdfs/ICIP05_H264_Fidelity.pdf

[PMBOKGANTT] Guía de los Fundamentos de la Dirección de Proyectos. Tercera Edición (Guía del PMBOK).

[BOESUELDO] Datos para la obtención del sueldo base.

https://www.boe.es/boe/dias/2016/06/28/pdfs/BOE-A-2016-6266.pdf

[COSTESEGSOC] Datos para la cotización de la seguridad social.

http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/%20Basesytiposdecotiza36537/index.htm#36538

9. Anexos

A continuación incorporamos en este punto el código completo implementado de algunas clases de las que consideramos puedan tener mayor relevancia e importancia en el proyecto.

9.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
   package="org.tfg.livedroid"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-feature android:name="android.hardware.camera" />
   <uses-feature android:name="android.hardware.camera.autofocus" />
   <uses-permission android:name="android.permission.CAMERA"
android:required="true" />
   <uses-permission android:name="android.permission.INTERNET"
android:required="true" />
   <uses-permission android:name="android.permission.WAKE_LOCK"
android:required="true"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
android:required="true" />
   <uses-permission android:name="android.permission.VIBRATE"
android:required="true"/>

   <application
       android:label="@string/app_name"
       android:allowBackup="true"
       android:icon="@drawable/ic_launcher_logo"
       android:name="org.tfg.livedroid.LiveDroidApplication"
       android:theme="@android:style/Theme.NoTitleBar" >
       <activity
           android:name="org.tfg.livedroid.StreamingMJPEGCameraActivity"
           android:label="@string/app_name"
           android:screenOrientation="landscape" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
       <activity
android:name="org.tfg.livedroid.LiveDroidPreferenceActivity" />
   </application>
</manifest>
```

9.2 MJPEGStreamer.java

```
package org.tfg.livedroid;
```

```
import java.io.IOException;
import java.util.List;

import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.Rect;
import android.graphics.YuvImage;
import android.hardware.Camera;
import android.os.HandlerThread;
import android.os.Looper;
import android.os.Handler;
import android.os.Message;
import android.os.Process;
import android.os.SystemClock;
import android.util.Log;
import android.view.SurfaceHolder;


final class MJPEGStreamer extends Object
{
   private static final String TAG = MJPEGStreamer.class.getSimpleName();

   private static final int TRY_TO_START_MESSAGE = 0;
   private static final int SEND_PREVIEW_DATA_MESSAGE = 1;

   private static final long OPEN_CAMERA_INTERVAL_MS = 1000L;

   private final Object mLock = new Object();
   private final FramesAverage mAverageSpf = new FramesAverage(50);

   private final int mCameraIndex;
   private final boolean mUseFlashLight;
   private final int mPort;
   private final int mPreviewSizeIndex;
   private final int mJpegQuality;
   private final SurfaceHolder mPreviewDisplay;

   private boolean mRunning = false;
   private Looper mLooper = null;
   private Handler mWorkHandler = null;
   private Camera mCamera = null;
   private int mPreviewFormat = Integer.MIN_VALUE;
   private int mPreviewWidth = Integer.MIN_VALUE;
   private int mPreviewHeight = Integer.MIN_VALUE;
   private Rect mPreviewRect = null;
   private int mPreviewBufferSize = Integer.MIN_VALUE;
   private MemoryOutputStream mJpegOutputStream = null;
   private MJPEGHttpStreamer mMJpegHttpStreamer = null;

   private long mNumFrames = 0L;
   private long mLastTimestamp = Long.MIN_VALUE;


   MJPEGStreamer(final int cameraIndex, final boolean useFlashLight, final
int port,
                 final int previewSizeIndex, final int jpegQuality, final
SurfaceHolder previewDisplay)
   {
       super();

       if (previewDisplay == null)
       {
           throw new IllegalArgumentException("previewDisplay must not be
null");
       } // if
```

```
        mCameraIndex = cameraIndex;
        mUseFlashLight = useFlashLight;
        mPort = port;
        mPreviewSizeIndex = previewSizeIndex;
        mJpegQuality = jpegQuality;
        mPreviewDisplay = previewDisplay;
    } // constructor()

    private final class WorkHandler extends Handler
    {
        private WorkHandler(final Looper looper)
        {
            super(looper);
        } // constructor()

        @Override
        public void handleMessage(final Message message)
        {
            switch (message.what)
            {
                case TRY_TO_START_MESSAGE:
                    tryStartStreaming();
                    break;
                case SEND_PREVIEW_DATA_MESSAGE:
                    final Object[] args = (Object[]) message.obj;
                    sendPreviewFrame((byte[]) args[0], (Camera) args[1],
(Long) args[2]);
                    break;
                default:
                    throw new IllegalArgumentException("cannot handle
message");
            } // switch
        } // handleMessage()
    } // class WorkHandler

    void start()
    {
        synchronized (mLock)
        {
            if (mRunning)
            {
                throw new IllegalStateException("CameraStreamer is already
running");
            } // if
            mRunning = true;
        } // synchronized

        final HandlerThread worker = new HandlerThread(TAG,
Process.THREAD_PRIORITY_MORE_FAVORABLE);
        worker.setDaemon(true);
        worker.start();
        mLooper = worker.getLooper();
        mWorkHandler = new WorkHandler(mLooper);
        mWorkHandler.obtainMessage(TRY_TO_START_MESSAGE).sendToTarget();
    } // start()

     // Para el streaming de imagenes. La cámara es liberada durante la
ejecución del Stop()
    void stop()
    {
        synchronized (mLock)
        {
            if (!mRunning)
            {
                throw new IllegalStateException("CameraStreamer is already
stopped");
            } // if
```


```
            mRunning = false;
            if (mMJpegHttpStreamer != null)
            {
                mMJpegHttpStreamer.stop();
            } // if
            if (mCamera != null)
            {
                mCamera.release();
                mCamera = null;
            } // if
        } // synchronized
        mLooper.quit();
    } // stop()

    private void tryStartStreaming()
    {
        try
        {
            while (true)
            {
                try
                {
                    startStreamingIfRunning();
                } //try
                catch (final RuntimeException openCameraFailed)
                {
                    Log.d(TAG, "Open camera failed, retying in " +
OPEN_CAMERA_INTERVAL_MS
                            + "ms", openCameraFailed);
                    Thread.sleep(OPEN_CAMERA_INTERVAL_MS);
                    continue;
                } // catch
               break;
            } // while
        } // try
        catch (final Exception startPreviewFailed)
        {
            // Captura las excepciones IOException y InterruptException de
startStreamingIfRunning
            Log.w(TAG, "Failed to start camera preview", startPreviewFailed);
        } // catch
    } // tryStartStreaming()

    private void startStreamingIfRunning() throws IOException
    {
        //Lanza RuntimeException si la cámara está abierta por otra
aplicación
        final Camera camera = Camera.open(mCameraIndex);
        final Camera.Parameters params = camera.getParameters();

        final List<Camera.Size> supportedPreviewSizes =
params.getSupportedPreviewSizes();
        final Camera.Size selectedPreviewSize =
supportedPreviewSizes.get(mPreviewSizeIndex);
        params.setPreviewSize(selectedPreviewSize.width,
selectedPreviewSize.height);

        if (mUseFlashLight)
        {
            params.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
        } // if

        //Configuramos el rango FPS del preview, el mayor valor es devuelto
primero
        final List<int[]> supportedPreviewFpsRanges =
params.getSupportedPreviewFpsRange();
```


```
        if (supportedPreviewFpsRanges != null)
        {
            final int[] range = supportedPreviewFpsRanges.get(0);

params.setPreviewFpsRange(range[Camera.Parameters.PREVIEW_FPS_MIN_INDEX],
                    range[Camera.Parameters.PREVIEW_FPS_MAX_INDEX]);
            camera.setParameters(params);
        } // if

        // Set up preview callback
        mPreviewFormat = params.getPreviewFormat();
        final Camera.Size previewSize = params.getPreviewSize();
        mPreviewWidth = previewSize.width;
        mPreviewHeight = previewSize.height;
        final int BITS_PER_BYTE = 8;
        final int bytesPerPixel = ImageFormat.getBitsPerPixel(mPreviewFormat)
/ BITS_PER_BYTE;
        // El buffer size puede ser calculado como width * height *
bytesPerPixel
        // sin embargo saltará error ya que necesita ser 1.5 veces mayor
        mPreviewBufferSize = mPreviewWidth * mPreviewHeight * bytesPerPixel *
3 / 2 + 1;
        camera.addCallbackBuffer(new byte[mPreviewBufferSize]);
        mPreviewRect = new Rect(0, 0, mPreviewWidth, mPreviewHeight);
        camera.setPreviewCallbackWithBuffer(mPreviewCallback);

        // Asumimos que la imagen compresa no es mayor que la original
        mJpegOutputStream = new MemoryOutputStream(mPreviewBufferSize);

        final MJPEGHttpStreamer streamer = new MJPEGHttpStreamer(mPort,
mPreviewBufferSize);
        streamer.start();

        synchronized (mLock)
        {
            if (!mRunning)
            {
                streamer.stop();
                camera.release();
                return;
            } // if

            try
            {
                camera.setPreviewDisplay(mPreviewDisplay);
            } // try
            catch (final IOException e)
            {
                streamer.stop();
                camera.release();
                throw e;
            } // catch

            mMJpegHttpStreamer = streamer;
            camera.startPreview();
            mCamera = camera;
        } // synchronized
    } // startStreamingIfRunning()

    private final Camera.PreviewCallback mPreviewCallback = new
Camera.PreviewCallback()
    {
        @Override
        public void onPreviewFrame(final byte[] data, final Camera camera)
        {
            final Long timestamp = SystemClock.elapsedRealtime();
            final Message message = mWorkHandler.obtainMessage();
```

```
            message.what = SEND_PREVIEW_DATA_MESSAGE;
            message.obj = new Object[]{ data, camera, timestamp };
            message.sendToTarget();

        } // onPreviewFrame(byte[], Camera)
    }; // mPreviewCallback

  private void sendPreviewFrame(final byte[] data, final Camera camera,
final long timestamp)
  {
        // Calcalute the timestamp
        final long MILLI_PER_SECOND = 1000L;
        final long timestampSeconds = timestamp / MILLI_PER_SECOND;

        // Update and log the frame rate
        final long LOGS_PER_FRAME = 10L;
        mNumFrames++;
        if (mLastTimestamp != Long.MIN_VALUE)
        {
            mAverageSpf.update(timestampSeconds - mLastTimestamp);
            if (mNumFrames % LOGS_PER_FRAME == LOGS_PER_FRAME - 1)
            {
                Log.d(TAG, "FPS: " + 1.0 / mAverageSpf.getAverage());
            } // if
        } // else

        mLastTimestamp = timestampSeconds;

        // Create JPEG
        final YuvImage image = new YuvImage(data, mPreviewFormat,
mPreviewWidth, mPreviewHeight,
                null);
        image.compressToJpeg(mPreviewRect, mJpegQuality, mJpegOutputStream);

        mMJpegHttpStreamer.streamJpeg(mJpegOutputStream.getBuffer(),
mJpegOutputStream.getLength(),
                timestamp);

        // Limpiar el buffer
        mJpegOutputStream.seek(0);
        camera.addCallbackBuffer(data);
  }// sendPreviewFrame()

    public Context MJpegContext(){

        return mMJpegHttpStreamer.returnContext();
    }

} // class CameraStreamer
```

9.3 UDPFinder.java

```
package org.tfg.livedroid;

import android.os.AsyncTask;
import android.util.Log;

import java.io.IOException;
import java.lang.reflect.Array;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
```


```
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.TimeUnit;

public class UDPFinder {

   private static final String TAG = UDPFinder.class.getSimpleName();
   private InetAddress broadcastIP;
   private DatagramSocket UDPSender;
   private String myIp;

   static FoundIPs foundIp = new FoundIPs();
   private boolean beacon = true;

   public void setBeacon(boolean beacon){
       this.beacon = beacon;
   }

   public FoundIPs getFoundIP(){
       return foundIp;
   }

   public void searchUDP(String ownIP){

               myIp = ownIP;

               //Prepare Broadcast IP
               String []splittedIP = myIp.split("\\.");
               String stringBroadcastIP = splittedIP[0] + "." +
splittedIP[1] + "." + splittedIP[2] + ".255";
               try{
                   broadcastIP = InetAddress.getByName(stringBroadcastIP);
               }
               catch(UnknownHostException e){Log.d(TAG, e.toString());}

               //TIMER CODE
               final Timer beaconWork = new Timer( );
               beaconWork.scheduleAtFixedRate(new TimerTask() {

                   @Override
                   public void run() {

                       foundIp.resetIps();

                       //Future response and time variables
                       long endingTime = System.currentTimeMillis() + 8000;
//8 seconds
                       Log.d("UDP", "Starting new Beacon search");

                       //Open a random port to send the package
                       try{
                           UDPSender = new DatagramSocket();
                           UDPSender.setBroadcast(true);
                           UDPSender.setSoTimeout(1500);
                       }
                       catch(SocketException e){Log.d(TAG, e.toString());}

                       byte[] sendData = "ARE YOU ALIVE?".getBytes();

                       //Sending broadcast packet
                       try{
                           DatagramPacket broadcastPacket = new
DatagramPacket(sendData,sendData.length,broadcastIP,8889);
```

```
                            UDPSender.send(broadcastPacket);
                            Log.d(TAG, "Beacon signal sent");
                        }
                        catch (Exception e){Log.d(TAG, e.toString());}

                        //Sent!! Now wait to responses
                        byte[] UDPreceiverBuff = new byte[15000];
                        DatagramPacket UDPreceivePacket = new
DatagramPacket(UDPreceiverBuff,UDPreceiverBuff.length);

                        while( System.currentTimeMillis() < endingTime ) {

                            try{
                                Log.d(TAG, "Beacon waiting for response");
                                UDPSender.receive(UDPreceivePacket);
                            }
                            catch(IOException e){}


                            if(UDPreceivePacket.getAddress() != null){
                                String ipToInclude =
UDPreceivePacket.getAddress().toString().replaceAll("\\/","");
                                foundIp.setIp(ipToInclude);
                            }
                        }

                        UDPSender.close();
                        UDPSender = null;

                        if(!beacon) beaconWork.cancel();

                    }
                }, 0,1000); //0 second delay every 1 seconds

                //END TIMER CODE

   }//SearchUDP
} //Class
```

10. Bibliografía

- Jesús Tomás Gironés, “El gran libro de Android” 5º edición 2016.
- Miguel Ángel Lozano Ortega y Antonio Javier Gallego Sánchez, “Desarrollo de aplicaciones Android con Java” 2017.
- https://stackoverflow.com/
- https://developer.android.com/
- http://codersblock.com/blog/motion-detection-with-javascript/
- https://www.encoding.com/http-live-streaming-hls/
- https://v4-alpha.getbootstrap.com/
- https://github.com/eligrey/FileSaver.js
- http://fabricjs.com/
- https://clipboardjs.com/
- https://ned.im/noty/#/
- https://github.com/yanzhenjie/AndServer