



UNIVERSIDAD DE VALLADOLID

E. T. S INGENIERÍA INFORMÁTICA

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

VISUALIZACIÓN LEGAL DE CONTENIDOS
DIGITALES A TRAVÉS DE LA RED.

Autor:
Jorge Nebreda González

Tutor:
Dr. Diego R. Llanos Ferraris

Resumen

La difusión de contenido multimedia a través de la red se ha realizado de múltiples formas, a través de distintas herramientas y permitiendo al usuario visualizar y/o descargar el contenido. La idea principal de este Trabajo de Fin de Grado es realizar un estudio detallado de las herramientas que permiten la visualización o la descarga del contenido. Una vez realizado este estudio se ha buscado escoger la forma más segura de conseguir la visualización del contenido a través de los protocolos RTP/RTSP y del control de procesos en el equipo del usuario. Partiendo de estas ideas se ha desarrollado un cliente y un servidor RTSP que permiten la transmisión y la reproducción del contenido multimedia, así como el control de procesos en el lado del cliente durante la reproducción. De esta forma se consigue una visualización del contenido con mayor seguridad al impedir, o al menos en gran medida, dificultar la copia o descarga del propio contenido.

Abstract

The spreading of multimedia content through the net has been carried out in multiple ways, using different tools and allowing the user to visualize and/or download that content. The main idea in this Project is to carry out a detailed study of the tools that allow the visualization or downloading of content. Once this study has been done, choosing the most secure way to visualize content through RTP/RTSP and the control of processes in user's equipment has been sought. Taking these ideas as a starting point, an RTSP client and server that allow the transmission and reproduction of multimedia content, as well as the control of processes by the client's side along the reproduction have been developed. In this way, a more secure visualization of the content is achieved, by preventing or, at least, making more difficult the copy or download of the content being viewed.

Agradecimientos

Con estas palabras quiero mostrar mi agradecimiento a aquellas personas que, tanto durante la realización de este Trabajo de Fin de Grado, como durante estos años, me han ayudado o me han servido de inspiración.

En primer lugar, quiero dar las gracias a Diego R. Llanos, ya que sin sus ideas y su asesoramiento este proyecto no se hubiese podido realizar. También me gustaría mostrar mi agradecimiento a muchos profesores y compañeros por la ayuda recibida durante estos años. Es difícil nombrarles a todos sin dejarme algún nombre en el tintero, pero aún así, gracias.

En un ámbito más personal son muchas las personas que me han animado o me han servido de inspiración durante estos años, por lo que me gustaría dar las gracias a mi familia y amigos. Nuevamente es difícil citar nombres sin olvidarme alguno, pero quiero hacer una mención especial para mis padres, mis hermanas y Leticia. Gracias por vuestros consejos y apoyo incondicional durante estos años. Sin vuestra ayuda esto no hubiese sido posible, así que, gracias de corazón.

Índice general

Resumen	III
Lista de figuras	IX
Lista de tablas	XI
1. Introducción	1
1.1. Motivación del proyecto.	1
1.2. Objetivos.	2
1.3. Estructura del documento.	3
2. Estado del arte	5
2.1. Navegadores.	5
2.1.1. Mozilla Firefox.	5
2.1.1.1. Historia.	5
2.1.1.2. Características.	6
2.1.1.3. Funcionamiento de los plugins.	7
2.1.1.4. API de los plugins.	11
2.1.2. Internet Explorer.	12
2.1.2.1. Historia.	12
2.1.2.2. Características.	13
2.1.2.3. Funcionamiento de los plugins.	16
2.1.2.4. API de los plugins.	17
2.1.3. Google Chrome.	17
2.1.3.1. Historia.	18
2.1.3.2. Características.	19
2.1.3.3. Funcionamiento de los plugins.	19
2.1.3.4. API de los plugins.	20
2.2. Tecnologías de reproducción de vídeo.	20
2.2.1. HTML5.	20
2.2.1.1. Historia.	20
2.2.1.2. Características.	22
2.2.2. Flash Player.	23

2.2.2.1.	Historia.	23
2.2.2.2.	Características.	24
2.2.3.	Otras alternativas para la reproducción de vídeo.	26
2.2.3.1.	Gnash.	26
2.2.3.2.	Lightspark.	27
2.2.3.3.	Swfdec.	27
2.2.3.4.	Shumway.	28
2.2.3.5.	VLC.	28
2.2.4.	Frameworks para la creación de plugins.	29
2.2.4.1.	FireBreath.	29
2.2.4.2.	JUCE.	30
2.2.4.3.	Qt.	31
2.3.	Vídeo a través del navegador	33
2.3.1.	Mecanismos de reproducción de vídeo	33
2.3.1.1.	Streaming	33
2.3.1.2.	Códec	35
2.3.1.3.	Real Time Streaming Protocol	37
2.3.2.	Captura y descarga de vídeo.	41
2.3.2.1.	Complementos para navegadores.	41
2.3.2.2.	Programas.	42
2.3.2.3.	Páginas web.	42
2.3.2.4.	Archivos temporales.	42
2.3.3.	Mecanismos de cifrado de vídeo.	43
2.3.3.1.	HTTPS	43
2.3.3.2.	TLS/SSL	45
2.3.4.	Autenticación de usuarios	47
2.4.	Conclusiones y valoraciones	48
3.	Análisis	51
3.1.	Alcance del proyecto.	51
3.2.	Requisitos.	52
3.2.1.	Requisitos funcionales.	52
3.2.2.	Requisitos no funcionales	52
3.3.	Modelo de dominio.	53
3.4.	Diagrama de casos de uso.	54
3.5.	Casos de uso.	55
4.	Gestión del proyecto	59
4.1.	Estudio de los riesgos.	59
4.2.	Planificación temporal.	64
4.3.	Plan de fases.	66

4.3.1. Fase de análisis	67
4.3.2. Fase de documentación	68
4.3.3. Fase de diseño	69
4.3.4. Fase de implementación	70
4.3.5. Fase de pruebas	72
4.3.6. Otras tareas	72
4.4. Presupuesto	73
5. Diseño	77
5.1. Descripción de la arquitectura	77
5.2. Patrones de arquitectura	77
5.2.1. Experto	78
5.2.2. Controlador	78
5.2.3. Estado	78
5.3. Modelo estructural	79
5.4. Diseño de la interfaz de usuario	80
5.5. Diagramas de secuencia	82
6. Implementación	87
6.1. Modelo de desarrollo	87
6.2. Entorno de desarrollo	87
6.3. Lenguajes de programación	88
6.4. Protocolos de comunicación	89
6.5. Control de procesos	89
6.6. Reproducción y transmisión del contenido	92
6.7. Estructura del código	94
7. Pruebas	95
7.1. Plan de pruebas	95
7.2. Casos de prueba	96
8. Manual de usuario e instalación	101
8.1. Descarga e inicio del cliente	101
8.2. Acceso al contenido	102
8.3. Reproducción del contenido y controles	103
9. Conclusiones y trabajo futuro	105
9.1. Consecución de objetivos	105
9.2. Valoración	106
9.3. Trabajos futuros	106
A. Anexos	107

Anexos	107
A.1. Creación de los archivos ejecutables	107
A.2. Contenido del DVD	107
Bibliografía	108

Índice de figuras

2.1. Arquitectura de Internet Explorer	14
2.2. Protocolos en RTSP.	38
2.3. Comunicación entre el navegador y un servidor RTSP.	39
2.4. Cabecera de un paquete RTP.	40
2.5. Posibles estados del servidor y el cliente RTSP.	40
2.6. Modelos de red de comunicación.	45
3.1. Modelo de dominio.	53
3.2. Diagrama de casos de uso.	54
4.1. Fases planificación temporal.	66
4.2. Diagrama fases planificación temporal.	66
4.3. Fase de análisis.	67
4.4. Diagrama fase de análisis.	67
4.5. Fase de documentación.	68
4.6. Diagrama fase de documentación.	69
4.7. Fase de diseño.	70
4.8. Diagrama fase de diseño.	70
4.9. Fase de implementación.	71
4.10. Diagrama fase de implementación.	71
4.11. Fase de pruebas.	72
4.12. Diagrama fase de pruebas.	72
4.13. Otras tareas.	73
4.14. Diagrama de otras tareas.	73
5.1. Arquitectura del sistema.	77
5.2. Patrón estado.	78
5.3. Diagrama de clases del lado del cliente.	79
5.4. Diagrama de clases del lado del servidor.	80
5.5. Interfaz del servidor.	80
5.6. Interfaz de acceso.	81
5.7. Interfaz del reproductor.	81
5.8. Interfaz del reproductor mostrando el contenido.	82

5.9. Diagrama de secuencia CU acceso.	83
5.10. Diagrama de secuencia CU play.	84
5.11. Diagrama de secuencia CU pause.	85
5.12. Diagrama de secuencia CU exit.	86
6.1. Creación del listado de procesos sospechosos.	90
6.2. Envío del listado de procesos sospechosos.	91
6.3. Recepción del listado de procesos sospechosos.	91
6.4. Comprobación del listado de procesos sospechosos.	92
6.5. Transformación del contenido.	93
6.6. Transformación de los paquetes RTP en imágenes.	93
8.1. Enlaces de descarga del cliente RTSP.	101
8.2. Mensaje de error: Servidor no disponible.	102
8.3. Pantalla de acceso.	102
8.4. Pantalla de reproducción del contenido.	103
8.5. Cliente durante la reproducción del contenido.	103

Índice de tablas

3.1. Caso de uso 1	56
3.2. Caso de uso 2	56
3.3. Caso de uso 3	57
3.4. Caso de uso 4	57
3.5. Caso de uso 5	58
4.1. Planificación temporal	65
4.2. Costes del proyecto	75
7.1. Caso de prueba 1	96
7.2. Caso de prueba 2	96
7.3. Caso de prueba 3	96
7.4. Caso de prueba 4	97
7.5. Caso de prueba 5	97
7.6. Caso de prueba 6	97
7.7. Caso de prueba 7	98
7.8. Caso de prueba 8	98
7.9. Caso de prueba 9	98
7.10. Caso de prueba 10	98
7.11. Caso de prueba 11	99
7.12. Caso de prueba 12	99
7.13. Caso de prueba 13	99
7.14. Caso de prueba 14	99

Capítulo 1

Introducción

1.1. Motivación del proyecto.

Actualmente, una gran cantidad de contenido multimedia es descargado o copiado de forma ilícita desde la web. Existen programas, complementos para navegadores y páginas web cuyo objetivo es descargar archivos de páginas web que ofrecen la reproducción pero no la descarga de estos contenidos. Otra forma de obtener este contenido es a través de la captura de la pantalla durante la reproducción del contenido.

Estas descargas y copias ilegales perjudican el trabajo tanto de los creadores de los contenidos, como de quien provee este contenido, ya sea de forma online o a través de otros medios. Las descargas ilícitas atentan contra la propiedad intelectual y los derechos de autor. Este es un tema controvertido, dado que las leyes de la propiedad intelectual varían en función de cada país.

Según la Organización Mundial de la Propiedad Intelectual (OMPI), "el propietario o titular de una obra puede disponer de ésta como le plazca, y ninguna persona física o jurídica podrá disponer legalmente de su propiedad sin su consentimiento".[1] Naturalmente el ejercicio de este derecho está sujeto a limitaciones.

Según la OMPI, el contenido multimedia que es de interés para el presente trabajo se encuentra dentro de la categoría de derechos de autor y derechos conexos, ya que incluye las obras cinematográficas y los contenidos digitales, que son los contenidos con los que trabajaremos.

Durante los últimos años se han tomado medidas legales y se han cerrado algunas páginas que ofrecían la posibilidad de acceder o descargar contenido obtenido de forma ilegal (algunos ejemplos de estas páginas son *Megaupload* o *Series.ly*). Sin embargo es complicado actuar legalmente contra el software que permite la copia y la descarga del contenido, porque, por ejemplo, el software que permite la captura de la pantalla tiene otras utilidades, como la grabación de la actividad del usuario para la creación de tutoriales.

En este Trabajo de Fin de Grado se ha buscado encontrar la forma de reproducir contenido multimedia de una forma segura a través de la red para tratar de evitar la copia o la descarga del mismo.

Se trata de conseguir que los usuarios puedan acceder al contenido multimedia a través de la red, es decir, que puedan obtener un contenido que no está en su equipo, pero evitando que los usuarios descarguen o copien dicho contenido. De este modo, solo los proveedores del contenido podrán distribuirlo.

La principal motivación de este proyecto ha sido la investigación de las tecnologías y herramientas de vídeo a través de la red, para ver si es posible implementar una aplicación que nos permita proveer contenido multimedia y a la vez evitar que éste sea copiado, y una vez definida la idea de esta tecnología llevar a cabo su desarrollo.

1.2. Objetivos.

El principal objetivo de este Trabajo de Fin de Grado es conseguir la visualización de contenido digital a través de internet, ya sea a través del navegador o de otros protocolos. Algunos de los objetivos básicos que debemos cumplir con la implementación de nuestro software son los siguientes:

- Permitir la correcta reproducción del contenido.
- Permitir el control por parte del usuario de la reproducción del contenido.
- Impedir, o al menos, dificultar la copia o descarga del contenido.

Para escoger las tecnologías más adecuadas para cumplir los objetivos previamente mencionados, antes de realizar la implementación del software realizaremos un estudio sobre los navegadores, los plugins y las principales tecnologías que permiten reproducir y copiar contenido a través de la red.

Para la consecución de los objetivos finales de este Trabajo de Fin de Grado previamente hemos de cumplir una serie de objetivos intermedios, como son:

- Realizar un estudio sobre las principales características de los diferentes navegadores.
- Realizar un estudio de las tecnologías utilizadas para la reproducción y copia de vídeo en la red.
- Analizar los métodos de desarrollo de plugins.
- Realizar un estudio sobre los métodos de cifrado y autenticación.

- Elegir la mejor alternativa para la reproducción del contenido.
- Analizar de funciones a realizar por el software.
- Diseñar la tecnología a desarrollar.
- Implementar el software para el envío y la reproducción del contenido.
- Probar el correcto funcionamiento de la tecnología implementada.
- Documentar de la realización del Trabajo de Fin de Grado.

En los siguientes capítulos describiremos de forma detallada los pasos realizados durante cada una de las etapas para la consecución de los objetivos previamente mencionados.

1.3. Estructura del documento.

Este proyecto se divide en nueve capítulos en los que detallaremos los conocimientos adquiridos y el trabajo realizado durante la realización de este TFG. Podemos describir estos capítulos del siguiente modo:

- Capítulo 1: Introducción. En este capítulo se describen las motivaciones que nos han llevado a realizar este Trabajo de Fin de grado y los objetivos que buscamos conseguir con la realización de este proyecto. También se realiza una breve descripción de la estructura de la memoria.
- Capítulo 2: Estado del arte. A lo largo de este apartado se realiza un detallado estudio sobre las múltiples tecnologías relacionadas con la reproducción de contenido multimedia a través de la red, y se escogen las tecnologías más adecuadas para la implementación de un software que nos permita cumplir con los objetivos previamente fijados para el Trabajo de Fin Grado. Cabe destacar la importancia de este capítulo por la gran cantidad de horas de trabajo que ha supuesto la búsqueda de información y la documentación sobre las tecnologías, y por la influencia que tiene sobre el resto del proyecto la elección de las tecnologías a utilizar para el desarrollo del mismo.
- Capítulo 3: Gestión del proyecto. En esta sección se muestra la planificación temporal, detallando las horas de trabajo dedicadas a cada tarea y los cambios en la planificación que han sido necesarios. Además se realiza el estudio de los riesgos que se pueden dar a lo largo del proyecto y se estima el presupuesto del mismo.
- Capítulo 4: Análisis. A lo largo de este capítulo se describen los requisitos que debe cumplir el software implementado y se detalla el funcionamiento del mismo a través de los casos de uso, el modelo de dominio y los diagramas de secuencia.

- Capítulo 5: Diseño. En esta sección se describe el tipo de desarrollo utilizado para la implementación del software y la arquitectura del propio software. También se detalla el diseño de la interfaz de usuario de la aplicación a realizar.
- Capítulo 6. Implementación. En este apartado se detalla el software implementado. Se describen tanto los lenguajes utilizados como las tecnologías y protocolos implementados.
- Capítulo 7: El capítulo de pruebas. En este capítulo se muestran las pruebas realizadas para determinar la validez y la corrección del software implementado y del funcionamiento del mismo.
- Capítulo 8: Manual de usuario e instalación. A lo largo de este apartado se detallan las instrucciones a seguir para la instalación de la aplicación desarrollada y los pasos que el usuario que debe seguir para aprovechar la funcionalidad del software.
- Capítulo 9: Conclusiones y trabajo futuro. En esta sección del trabajo se muestran las conclusiones alcanzadas después de la realización de este proyecto. Además se plantean una serie de posibles mejoras o de nuevas funcionalidades que pueden incluirse en la aplicación implementada.
- Anexos.
- Bibliografía.

Capítulo 2

Estado del arte

Durante este capítulo se realiza un estudio detallado de las tecnologías relacionadas con la reproducción y la descarga de contenido digital a través de la red. Esto nos permite seleccionar para su posterior implementación la tecnología que mejor se adapte a nuestros objetivos.

2.1. Navegadores.

2.1.1. Mozilla Firefox.

Mozilla Firefox es un navegador web de código abierto desarrollado para diferentes sistemas operativos como Windows, Linux y Mac OS X. Actualmente es el tercer navegador más usado a nivel mundial con una cuota de mercado entorno al 17% [2].

2.1.1.1. Historia.

La primera versión de Firefox fue publicada el 9 de noviembre de 2004 [3]. Fue creado como una rama del proyecto Mozilla por David Hyatt, Joe Hewitt y Blake Ross. Posteriormente se han lanzado múltiples versiones. Cabe destacar que desde la versión 5 en junio de 2011 se inició un ciclo de lanzamientos cada seis semanas. La última versión, Firefox 40.0.3 fue lanzada el 27 de agosto de 2015.

Los creadores de Firefox creían que las exigencias comerciales del patrocinio de Netscape y el elevado número de características de Mozilla Application Suite limitaban la utilidad de la misma. Esta suite incluía funciones de gestión del correo electrónico, el protocolo de comunicación IRC (Internet Relay Chat) y un editor HTML.

Se consideró que la suite era demasiado grande, pesada y dependiente de su promotor (Netscape) y que esto podía comprometer su futuro. Por lo que en la primavera de 2002 se creó una vertiente experimental del proyecto Mozilla. Con esto se buscaba una mejora del código y la interfaz del navegador, además de buscar eliminar las funciones que se consideraban ajenas al

navegador.

Tras una serie de problemas legales con el nombre del proyecto, el 9 de noviembre de 2004 se publica la versión 1.0 de Firefox. En menos de un año alcanzó los 100 millones de descargas (muchas de usuarios particulares) convirtiéndose en una de las aplicaciones de código libre más descargadas.

Hoy en día Firefox es uno de los navegadores más usados con un gran éxito en algunos países y millones de usuarios en todo el mundo. Con la publicación de las sucesivas versiones de Firefox se han ido realizando cambios y añadiendo funcionalidades. A continuación explicaremos las principales características de Firefox y las que resulten de mayor interés para este proyecto.

2.1.1.2. Características.

Firefox incluye las características habituales de la mayoría de navegadores, como son: navegación por pestañas, corrector ortográfico, integración del motor de búsqueda que desee el usuario, navegación con georreferenciación, administrador de descargas, lector RSS, navegación privada o marcadores dinámicos, entre muchas otras funcionalidades [4].

Un elemento representativo de Firefox es la posibilidad de añadir funcionalidad al navegador a través de complementos, entre los que están las extensiones y los plugins. Estos complementos pueden ser desarrollados tanto por Mozilla como por terceros, ya sean aficionados o comerciales.

Podemos personalizar el navegador con múltiples complementos a través de Mozilla Addons, y con ello obtener una gran variedad de funciones. Según algunos estudios, esta característica convierte a Firefox en el navegador más personalizable y seguro del momento [5].

Otras funciones adicionales de Firefox son: la posibilidad de realizar un informe de fallo con la información técnica del sistema cuando tiene lugar un error inesperado; a través del programa Mozilla Telemetry [6] se puede estudiar el rendimiento del navegador al obtener datos sobre el hardware, el uso y la personalización del navegador; además los usuarios pueden sincronizar las contraseñas, el historial y los marcadores entre ordenadores y móviles mediante Firefox Sync.

El navegador posee una serie de herramientas incorporadas de interés para el desarrollo web, como son: una consola de errores; la herramienta Scratchpad [7] que proporciona un entorno para probar código JavaScript pudiendo escribir, ejecutar y ver el resultado del código; un editor HTML; el Inspector DOM o extensiones como Firebug que permite editar, monitorizar y depurar código fuente.

Firefox es compatible con múltiples lenguajes web, entre ellos: HTML, XML, XHTML, CSS, JavaScript o DOM . El navegador también es desarrollado en C++, XUL y XBL. Para el desarrollo del proyecto se han utilizado varios de estos lenguajes que posteriormente serán detallados.

En cuanto a temas de seguridad Firefox implementa el sistema SSL/TLS, buscando proteger la comunicación con los servidores web empleando una fuerte criptografía con el protocolo https. Además tiene una protección antiphishing, antimalware y está integrado con el antivirus. Por todo esto se considera que Firefox es un navegador que se preocupa por la seguridad y la privacidad de sus usuarios.

Firefox es un navegador multiplataforma con disponibilidad para múltiples sistemas operativos, entre ellos: Microsoft Windows (desde Windows 98 hasta Windows 8.1), Mac OS X y sistemas operativos basados en Linux que usen el sistema de ventanas X Window.

El código de Mozilla Firefox es libre y abierto, y está distribuido bajo tres licencias: Licencia Pública de Mozilla (MPL), Licencia pública general de GNU (GPL) y la Licencia Pública General Reducida de GNU (LGPL) [8]. Esto permite ver, modificar y redistribuir el código fuente a cualquiera que lo desee. Cabe destacar que aproximadamente el 40 % del código de Firefox está escrito por voluntarios.

Para la realización de este proyecto, de todas las características y funcionalidades que Firefox nos ofrece vamos a tratar más en profundidad los complementos, en concreto los plugins, así como los Frameworks, lenguajes y métodos utilizados para el desarrollo de los mismos.

2.1.1.3. Funcionamiento de los plugins.

Los plugins son bibliotecas compartidas que los usuarios pueden instalar para poder ver ciertos contenidos que la aplicación, en este caso el navegador, no puede mostrar por sí misma [9]. Por ejemplo, algunos plugins como Adobe Flash Player o QuickTime Player permiten reproducir vídeos en determinados formatos en una página web. Otros plugin como Adobe Reader permiten abrir directamente en el navegador archivos en PDF.

Con esta definición de los plugins podemos ver que son totalmente distintos a las extensiones, ya que estas añaden, modifican o mejoran funcionalidades del propio navegador. Los plugins también son diferentes de los plugins de búsqueda cuya función es añadir motores de búsqueda adicionales.

Un antecesor de los plugins eran las helper application. Se trata de programas externos creados para mostrar contenido obtenido con un navegador. Al contrario que un plugin, el cual incluye todo su código en el espacio de direcciones del navegador, una helper application es una aplicación estándar.

Los plugins se escriben usando NPAPI, un API del navegador para plugins. El código principal de la documentación para NPAPI es la referencia a Gecko. Para hacer un plugin programable para

páginas web se utiliza `npruntime`.

Desde Scratch se pueden escribir plugins completos usando APIs de C (o C++) o pueden ser contruidos con un framework para plugins como FireBreath, JUCE o QtBrowserPlugin. Existen también una serie de herramientas de generación de código que pueden ser útiles.

Los plugins ofrecen una gran variedad de características que permiten mejorar la flexibilidad de los navegadores basados en Gecko. Gecko es un layout engine desarrollado por el proyecto Mozilla, su función es leer el contenido web como HTML, CSS, XUL o JavaScript y mostrarlo en la pantalla del usuario.

Algunos ejemplos de estos plugins son, entre otros: plugins de vídeo, visores de documentos o servicios de compresión/descompresión. Son muchas las posibilidades que nos ofrece el uso de los plugins. Creando plugins con el API podemos conseguir las siguientes funcionalidades de los mismos: recibir los eventos del ratón y el teclado, obtener datos de la red usando URLs, comunicarse con JavaScript/DOM desde el código nativo o registrar uno o más tipos MIME (Multipurpose Internet Mail Extensions), entre otras funcionalidades. MIME es una extensión del protocolo original de correo electrónico, que permite el intercambio de archivos de diferentes tipos de datos (imágenes, audio, vídeo y otros) en Internet [10].

Cuando el navegador encuentra un tipo MIME siempre busca en primer lugar un plugin registrado, si no encuentra un plugin para ese tipo MIME busca una helper application que es iniciada por el navegador pero se ejecuta de forma separada.

Un Internet media type es un identificador estándar usado en internet para indicar el tipo de datos que contiene un archivo. Los navegadores les utilizan para determinar cómo mostrar o enviar archivos que no estén en formato HTML. Los identificadores fueron definidos en RFC 2046 y se conocen como tipos MIME porque se refieren a la parte no ASCII de los emails creados usando la especificación MIME. También se les conoce como Content-types. Su uso se ha extendido desde el envío de emails a través de SMTP a otros protocolos como HTTP, RTP o SIP.

Al contrario que en las aplicaciones, el ciclo de vida de un plugin está totalmente controlado por la página web que lo llama. Cuando se inicia Gecko busca plugins para funciones concretas del navegador [11].

Cuando el usuario abre una página que contiene datos de tipo multimedia y se invoca un plugin, el navegador responde con la siguiente secuencia de acciones: buscar un plugin con el tipo MIME adecuado, cargar el código del plugin en memoria, inicializar el plugin y crear una nueva instancia del mismo.

Gecko puede cargar múltiples instancias del mismo plugin en una misma página o en varias ventanas abiertas al mismo tiempo. Por ejemplo, si estamos navegando en una página con varios vídeos de Youtube disponibles para reproducir y comenzamos a reproducirlos simultáneamente, se crearán tantas instancias del plugin de reproducción de vídeo como vídeos estemos reproduciendo simultáneamente.

Cuando se abandona la página o se cierra la ventana, las instancias del plugin (correspondientes a dicha ventana) serán eliminadas. En el momento que la última instancia del plugin es eliminada el código del plugin será descargado de la memoria.

Los plugins son módulos de código dinámico que están asociados con uno o más tipos MIME. Cuando se inicia el navegador se enumeran los plugins disponibles, se leen los recursos de cada plugin para determinar los tipos MIME de cada plugin y se registra cada plugin para su tipo MIME. Estos son los pasos en la vida de un plugin desde que se carga hasta que se elimina:

1. Cuando Gecko encuentra datos del tipo MIME registrado para un plugin, si éste no está cargado en memoria, lo carga y crea una nueva instancia del mismo. La primera vez que se carga el código del plugin en memoria Gecko llama a la función **NP_Initialize**.
2. El navegador llama a la función del API del plugin **NPP_New** cuando la instancia es creada. Como ya hemos mencionado previamente, pueden existir varias instancias del mismo plugin, ya sean varios objetos en una misma página o en ventanas diferentes.
3. Cuando se elimina una instancia del plugin (abandonando la página o cerrando la ventana), Gecko llama a la función **NPP_Destroy** para informar al plugin de que la instancia ha sido eliminada.
4. Cuando la última instancia del plugin sea eliminada, el código del plugin se descarga de la memoria y Gecko llama a la función **NPP_Shutdown**. Cuando los plugins no están cargados en memoria no consumen recursos, salvo el correspondiente espacio de almacenamiento en disco.

Un plugin es una librería cuyo código está conforme a los estándares de la sintaxis de C. La API (Application Programming Interface) de los plugins está compuesta por dos grupos de funciones y un grupo de estructuras de datos compartidas [12].

Los métodos de un plugin son funciones implementadas dentro del mismo, pero es Gecko quien las llama. El nombre de todas las funciones en el API empieza por **NPP_**, salvo **NP_Initialize** y **NP_Shutdown** que son puntos de entrada directa a la librería y no se refiere a ninguna instancia del plugin en particular.

Los métodos del navegador son funciones implementadas por Gecko y el plugin se encarga de llamar a estas funciones. El nombre de todas las funciones del navegador en el API empieza por

NPN_.

Las estructuras de datos son tipos específicos del plugin definidos para su uso en el API del plugin. El nombre de estas estructuras empieza por NP. En general las operaciones de todas las funciones del API son las mismas en todas las plataformas. Cuando esto cambia, hay una sección de referencia para la función, donde se explica la diferencia.

Un plugin es un módulo de código dinámico que es nativo a la plataforma específica en la que se está ejecutando el navegador. Se trata de una librería de código que solo puede ser ejecutada desde el navegador. Aunque los plugins son específicos para cada sistema operativo, la API del plugin está diseñada para proporcionar el mayor grado de flexibilidad y tener una funcionalidad consistente en todos los sistemas operativos.

Se puede utilizar el API del plugin para escribir plugins dirigidos a los media type y que proporcionan un alto rendimiento aprovechando las ventajas del código nativo. Los plugins dan la oportunidad de integrar el código dependiente de la plataforma sin errores y mejorar la funcionalidad de Gecko proporcionando soporte para nuevos tipos de datos.

El tipo de archivo del plugin depende del sistema operativo:

- **MS Windows:** .DLL (Dynamic Link Library).
- **Unix:** .SO or .DSO (Shared Objects) .
- **Mac OS X:** PPC/x86/Universal loadable Mach-O bundle.

La forma en que Gecko localiza los plugins también varía según el sistema operativo. Dependiendo de si se trata de Windows, Mac OS X o Linux las librerías están situadas en diferentes directorios.

Los streams son objetos que representan URLs (Uniform Resource Locator) y los datos que contienen o datos enviados por un plugin sin URL asociada. Aunque un stream esté asociado con una única instancia específica de un plugin, éste puede tener asociado más de un stream por instancia.

Los streams pueden ser producidos por el navegador y consumidos por una instancia del plugin o producidos por una instancia y consumidos por el navegador. Cada stream tiene asociado un tipo MIME identificando el tipo de los datos que contiene. El API de los plugins dispone de funciones para recibir y enviar streams [13].

Cuando el navegador produce un stream este puede ser automáticamente enviado o requerido por una instancia de un plugin. El navegador llama a los métodos del plugin `NPP_NewStream`,

NPP_WriteReady, NPP_Write y NPP_DestroyStream para respectivamente crear un stream, ver la cantidad de datos que el plugin puede manejar, insertar los datos en el stream y destruirlo.

La instancia del plugin elige el modo de transmisión para los streams producidos por el navegador:

- **Modo normal:** el navegador usa el método NPP_Write para insertar los datos del stream en la instancia a medida que están disponibles.
- **Modo acceso aleatorio:** el plugin llama al método NPN_RequestRead para conseguir los datos del stream. Normalmente este método es menos eficiente, porque se debe descargar el stream completo a un archivo temporal antes de usarse. A menos que el stream provenga de un archivo local o de un servidor HTTP que soporte la extensión del rango de byte de HTTP.
- **Modo archivo:** el navegador guarda el stream completo en un archivo local y pasa la ruta del archivo a la instancia del plugin a través del método NPP_StreamAsFile.

Los streams enviados por el plugin al navegador funcionan de forma opuesta a como lo hacen los streams en modo normal producidos por el navegador. Con los streams en modo normal el navegador llama al plugin para decirle cuándo un stream es creado y para insertar más datos. En cambio en los streams producidos por el plugin, el plugin llama a los métodos NPP_NewStream, NPP_Write y NPP_DestroyStream para crear el stream, insertar datos en él y destruirlo.

2.1.1.4. API de los plugins.

NPAPI (Netscape Plugin Application Programming Interface) es una arquitectura multiplataforma para los plugins utilizada por bastantes navegadores. En un principio se desarrolló para Netscape, comenzando en 1995 con Netscape Navigator 2.0, pero fue adoptada por Internet Explorer en 1996 y posteriormente por muchos otros navegadores [14].

Los plugins declaran el tipo de contenido que pueden controlar. Cuando el navegador encuentra contenido, carga el plugin asociado a dicho tipo de contenido. Además, reserva espacio para que el plugin pueda renderizar los datos. Los plugins se ejecutan dentro la página. Esto ha cambiado con respecto a las anteriores versiones de los navegadores, en las que se llamaba a una aplicación externa para manejar los tipos de datos desconocidos para el navegador.

El API requiere que cada plugin implemente una serie de funciones para su creación, inicialización, destrucción y el tratamiento del contenido. Además NPAPI soporta funciones para: encriptado, plugins a pantalla completa, plugins sin ventanas y streaming del contenido, entre otros [15].

Los siguientes navegadores soportan los plugins NPAPI:

- Mozilla Camino, Mozilla Firefox, Mozilla Application Suite y Mozilla SeaMonkey.

- Safari.
- Konqueror.
- Opera.
- Google Chrome / Chromium: Chrome ya ha retirado el soporte para NPAPI en la versión 35 de Chrome para Linux y en septiembre de 2015 se retirará el soporte de NPAPI para todas las plataformas.
- Internet Explorer hasta su versión 5.5 SP2.
- Y otros navegadores como: Web, WebOS Isis browser, Netscape Navigator, Qupzilla o Odyssey Web Browser.

2.1.2. Internet Explorer.

Internet Explorer es un navegador web para Windows desarrollado por Microsoft desde 1995, la primera versión de IE estaba basada en Spyglass Mosaic la rama comercial de navegador Mosaic desarrollado por la NCSA (National Center for Supercomputing Applications).

2.1.2.1. Historia.

El proyecto de Internet Explorer empezó en verano de 1994 dirigido por Thomas Reardon, posteriormente fue dirigido por Benjamin Slivka [16]. Microsoft lanzó Internet Explorer 1.0 en agosto de 1995, la versión 2.0 fue lanzada en noviembre del mismo año incluyendo avances como el soporte de cookies o conexiones cifradas SSL.

La versión 3.0 de IE se lanzó en agosto de 1996 de forma gratuita al incluirse con Windows 95 OEM Service Release 2 (OSR2), con esta versión IE se convirtió en una suite de aplicaciones al incluir Internet Mail and News, la libreta de direcciones de Windows y posteriormente Netmeeting y el Reproductor Multimedia. Con esta versión IE se acercaba al estilo de navegador impuesto por Netscape incluyendo las siguientes tecnologías: CSS, ActiveX, Plugins (NPAPI) y JScript. A partir de este momento comenzaron las controversias con IE, en forma de demandas por monopolio y problemas de seguridad.

En septiembre de 1997 se lanzó IE 4.0 con bastantes novedades. En primer lugar IE 4.0 tenía una profunda integración con Windows 98, se podían explorar las carpetas y unidades a través del propio navegador. Se incorporó HTML dinámico y la tecnología Channel Definition Format, antecesor de RSS. En esta versión Outlook Express fue incluido en el navegador como sustituto de Internet Mails and News.

Podríamos señalar el lanzamiento de esta versión de IE como el comienzo de la guerra de navegadores que mantuvieron Microsoft y Netscape por conseguir el dominio del mercado de los

navegadores. Este enfrentamiento terminó con la desaparición de Netscape y con un dominio total del mercado de los navegadores por parte de IE, alcanzando una cuota de mercado superior al 95%. Esta victoria de Microsoft se basó en una gran inversión tanto en publicidad, como en la mejora de la tecnologías, consiguiendo superar las de Netscape. A pesar de que en un principio Netscape tenía una cuota de mercado entorno al 90%, Microsoft era una compañía mucho mayor y poder incluir IE por defectos en sus sistemas operativos le daba una gran ventaja.

Durante los siguientes años IE dominó el mercado de los navegadores de forma aplastante, pero tuvo que enfrentarse a demandas por monopolio y competencia desleal. Aún durante sus años de dominio IE ha tenido muchos problemas relacionados con la seguridad y los virus. Con el tiempo IE ha seguido incluyendo tecnologías y lanzando nuevas versiones, la más reciente es Internet Explorer 11.

Con el paso del tiempo la cuota de mercado de IE ha sufrido un constante descenso, debido a la competencia de Google Chrome y Mozilla Firefox, estos navegadores son más versátiles, más ligeros y más rápidos. Actualmente IE tiene una cuota de mercado entorno al 20%, en gran parte debido a su uso en determinadas empresas y a que viene integrado en el sistema operativo Microsoft Windows. Hoy en día Internet Explorer sigue siendo el segundo navegador más usado, por detrás de Google Chrome y seguido muy de cerca por Mozilla Firefox.

2.1.2.2. Características.

Internet Explorer proporciona las características habituales que ofrecen la mayoría de los navegadores, aunque con alguna peculiaridades.

IE utiliza el motor de diseño Trident (MSHTML) desde la versión 4.0 de Explorer. Trident es un componente software que busca permitir a los desarrolladores añadir funcionalidades de navegación web a sus aplicaciones. Tiene una interfaz COM para acceder y editar páginas web en un entorno que soporte COM, como C++ o .NET.

Usando el motor de diseño Trident, Internet Explorer soporta HTML 4.01, HTML 5, CSS 1,2 y 3 XML 1.0 y DOM, con algunos pequeños problemas de implementación. Admite sin problemas XSLT 1.0 y proyecta soporte para XSLT 2.0.

Internet Explorer ha introducido múltiples extensiones propietarias en muchos de los estándar, incluyendo HTML, CSS y DOM. Esto hace que algunas páginas web solo puedan visualizarse de forma correcta con IE. Se han introducido algunas extensiones en DOM que posteriormente han adoptado otros navegadores.

Una característica que diferencia a IE de sus competidores es que está desarrollado prácticamente en exclusiva para los sistemas operativos de Microsoft Windows. Se sacaron versiones para

Mac OS X y para sistemas tipos UNIX pero no tuvieron continuidad. Se puede instalar en estos sistemas operativos a través de Wine, pero no es una alternativa competitiva.

IE usa una arquitectura dividida en componentes, desarrollada con la tecnología COM. Se divide en varios componentes principales contenidos en archivos .dll, con un conjunto de interfaces COM. Estos componentes son organizados por el ejecutable de Internet Explorer IExplore.exe. Estos son los principales componentes [17]:

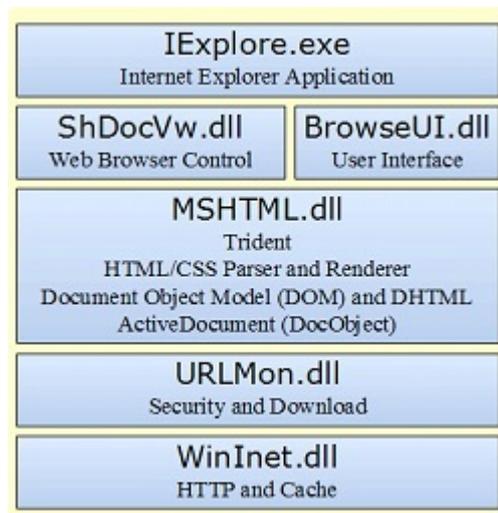


Figura 2.1: Arquitectura de Internet Explorer

- **IExplore.exe:** está en el nivel más alto y es el ejecutable de IE. Es una pequeña aplicación que confía a los otros componentes de Internet Explorer el trabajo de renderizado, navegación, implementación de protocolos, etc.
- **BrowsUI.dll:** proporciona la interfaz de usuario para Internet Explorer. A menudo referido como chrome, este DLL incluye las barra de direcciones de IE, la barra de estado, menús, etc.
- **WinInet.dll:** controla el protocolo de Internet Explorer. Implementa los protocolos HTTP y FTP junto con la gestión de caché.
- **URLMon.dll:** ofrece funcionalidad para manejar los tipos MIME y la descarga de código.
- **MSHTML.dll:** es el corazón de Internet Explorer y se encarga del análisis y el renderizado de HTML y CSS. Mshtml.dll se conoce a menudo como Trident. Este DLL expone interfaces que permiten su alojamiento como un documento activo. Otras aplicaciones, como Microsoft Word, Microsoft Excel, Microsoft Visio y muchas otras aplicaciones que no pertenecen a la compañía, también presentan interfaces de documento activo, por lo que pueden ser alojadas por shdocvw.dll. Por ejemplo, cuando un usuario navega de una página en HTML a un documento de Word, mshtml.dll se intercambia por un DLL proporcionado por Word, el cual, después renderiza este tipo de documento. Mshtml.dll puede ser útil después para alojar

otros componentes dependiendo del contenido del documento HTML, como herramientas de scripting (Microsoft JScript o Visual Basic Scripting Edition, controles ActiveX, datos XML, etc.).

- **ShDocVw.dll:** proporciona funcionalidad como la navegación y el historial, y es comúnmente conocido como el control del navegador web. Este DLL muestra las interfaces de control de ActiveX, permitiendo alojar de una forma sencilla el DLL en una aplicación Windows mediante Frameworks de Windows, como Microsoft Visual Basic, Microsoft Foundation Classes (MFC), Active Template Library (ATL) o Microsoft .NET. Cuando la aplicación albergue un control de navegador web, obtiene toda la funcionalidad de Internet Explorer excepto la interfaz de usuario proporcionada por Browseui.dll. Esto significa que será necesario facilitar las implementaciones de barras de tareas y menús del usuario.

La seguridad de Internet Explorer se basa en zonas de seguridad y agrupa los sitios en función de algunas condiciones, por ejemplo, el tipo de red o de usuario. Las restricciones de seguridad se aplican por zonas, de forma que los sitios de una misma zona estén sujetos a las mismas restricciones [18]. Algunos ejemplos de medidas de seguridad de IE son:

- Desde la versión 6 de IE los archivos ejecutables descargados de internet son marcados por el Servicio de Ejecución de Windows como potencialmente peligrosos. Así se previene la instalación de malware.
- Con IE 7 se añade un filtro contra phishing (robo de identidad), que no permite el acceso a sitios falsos a no ser que el usuario anule dicha restricción. Desde la versión 8 también se impide el acceso a los sitios conocidos por almacenar malware, y se analizan las descargas en busca de software malicioso.
- Desde Windows Vista el navegador se ejecuta en modo protegido, de forma que los privilegios del navegador están bastante restringidos. De esta forma no se pueden realizar grandes cambios en el sistema. Se puede ejecutar el navegador sin el modo protegido, pero no es lo recomendable.
- A través del servicio de actualizaciones de Windows se proveen periódicamente actualizaciones y parches de seguridad para el navegador.

Internet Explorer ha tenido muchos problemas de seguridad y ha sido objeto de bastantes vulnerabilidades. Una buena parte de los virus, spyware y adware que circula por Internet se han generado y distribuido aprovechando fallos y defectos de la arquitectura de seguridad de IE [19].

Una parte importante de los fallos de seguridad que afectan a Internet Explorer no se han generado en el propio navegador, sino en las extensiones basadas en ActiveX que utiliza. Estas extensiones tienen los mismo privilegios que IE y una fallo de seguridad en las mismas tiene la misma gravedad que un fallo de seguridad en el navegador.

Hay varios casos muy conocidos de problemas de seguridad relacionados con Internet Explorer, estos han generado muchas críticas hacia IE y le ha hecho perder prestigio frente a rivales como Google Chrome o Mozilla Firefox. Esto ha llevado a Microsoft a la creación de su nuevo navegador Microsoft Edge, que en un futuro se incluirá con Windows 10.

2.1.2.3. Funcionamiento de los plugins.

La arquitectura de componentes de Internet Explorer está basada en COM, hay bastantes formas de añadir funcionalidades, aunque lo podemos dividir en tres grandes bloques:

- **Extensiones del navegador:** este tipo de extensiones añaden funcionalidades a IE con características como barras de herramientas personalizadas, barras de búsqueda, atajos para el menú o BHOs (Browser Helper Objects).
- **Extensiones de contenido:** estas extensiones aumentan los tipos de contenido que puede ser analizado y mostrado por el navegador. Esta categoría de extensiones incluye los controles ActiveX y los documentos activos. Su uso depende de que el contenido sea accedido desde el navegador. Posteriormente trataremos más en profundidad estas extensiones ya que son las de mayor interés para este proyecto.
- **Alojamiento y reutilización:** alojando y reutilizando los componentes de Internet Explorer como una parte de tu aplicación puedes crear tu propio navegador y añadir nuevas funcionalidades.

Las extensiones de contenido son invocadas específicamente por el contenido de Internet Explorer. Podemos diferenciar varios tipos de extensiones de contenido:

- **Documentos activos:** son apropiados para reemplazar HTML con nuestro propio renderizado de un contenido específico. Por ejemplo, cuando un usuario pasa de una página HTML a un documento Word, se intercambia el documento activo mshtml.dll por el documento activo del visor de Word. El soporte de documentos activos de Internet Explorer permite combinar los menús para poder mostrar toda la funcionalidad disponible con el nuevo documento activo.
- **Controles de ActiveX:** son un mecanismo muy potente para extender las funcionalidades de HTML. Un control de ActiveX es básicamente un objeto OLE (object linking and embedding) que soporta la interfaz IUnknown, normalmente soporta muchas otras interfaces para ofrecer más funcionalidad, aunque cualquier otra interfaz se considera opcional. Los controles de ActiveX se han convertido en la principal arquitectura para desarrollar componentes de software usados en múltiples contenedores.
- **Comportamientos (Behaviors):** permiten una integración más profunda con la tecnología de renderizado de HTML que los controles de ActiveX. Hay dos tipos de comportamientos, los comportamientos basados en script y los comportamientos binarios. Los

comportamientos basados en script no son verdaderas extensiones para la funcionalidad del navegador. Los comportamientos binarios son objetos COM integrados en profundidad con el análisis y el renderizado de HTML. Algunos ejemplos de comportamientos binarios incluyen extensiones para MathML y SVG (Scalable Vector Graphics), esto permite mejorar la integración de los gráficos vectoriales y matemáticos con HTML.

- **Controles de Windows Forms:** se trata de un API incluida en Microsoft .NET Framework que permite crear controles para Internet Explorer. Cada control tiene una serie de propiedades, eventos y métodos que lo hacen adecuado para su propósito.
- **Protocolos de conexión (Pluggable Protocols):** dan soporte a los protocolos de comunicación de Internet Explorer, los protocolos suelen ser específicos para el tipo de datos que soportan. La funcionalidad de los protocolos de conexión se exporta a través de Urlmon.dll.

De estos tipos de extensiones, la que sería de interés para este proyecto es ActiveX, aunque en las últimas versiones de Internet Explorer se está intentando minimizar el uso de ActiveX y los plugins, aprovechando las nuevas funcionalidades introducidas con HTML5 para la reproducción y el tratamiento de vídeo. En la siguiente sección trataremos en profundidad estas nuevas funcionalidades que se han incluido en HTML5.

2.1.2.4. API de los plugins.

Internet Explorer dio soporte a los plugins basados en NPAPI hasta la versión de IE 5.5 SP 2. Desde entonces los plugins en Internet Explorer han tomado la forma de objetos de ActiveX o Browser Helper Object (BHOs). A pesar de esto, existe un control de ActiveX, pluginhostctrl.dll, que permite que versiones posteriores de Internet Explorer soporten los plugins basados en NPAPI, aunque con ciertas limitaciones [20].

Para el desarrollo de un control de ActiveX o de un Browser Helper Object nos podemos basar en Internet Explorer API Reference o Internet Explorer Platform APIs. Aunque, como ya hemos indicado anteriormente, Internet Explorer de cara al tratamiento de contenido multimedia ha buscado prescindir de los plugins y los controles de ActiveX y tratar de aprovechar las nuevas funcionalidades de HTML5. Es probable que Microsoft Edge, el sustituto de Internet Explorer, continúe con esta forma de tratamiento para el contenido multimedia, pero aún no lo sabemos.

2.1.3. Google Chrome.

Google Chrome es un navegador web de dominio público desarrollado por Google. A día de hoy Google Chrome es el navegador web más usado a nivel mundial, con una cuota de mercado cercana al 47% [21].

2.1.3.1. Historia.

Primero se publicó una versión beta para Microsoft Windows en septiembre de 2008. En diciembre de ese mismo año se publicó una versión estable. Google libera la mayor parte del código fuente de Chrome en el proyecto Chromium, pero un componente notable que no es de código libre es Adobe Flash Player integrado en el navegador.

Google Chrome es un navegador bastante reciente en comparación con sus competidores, aunque eso no le ha impedido conseguir un gran éxito en tan poco tiempo. Durante años, el CEO de Google, Eric Schmidt, se opuso a la creación de un navegador propio. Afirmaba que Google era una empresa pequeña y no quería entrar en una guerra de navegadores. Aunque después de que los cofundadores de Google, Sergey Brin y Larry Page, contrataran varios desarrolladores de Mozilla Firefox y crearan una versión preliminar de Chrome, Schmidt cambió de idea.

El anuncio del lanzamiento de Chrome estaba programado para el 3 de septiembre de 2008. Se iba a enviar un cómic a los periodistas y bloggers explicando las características del nuevo navegador. Pero las copias destinadas a Europa se enviaron antes de tiempo, y un blogero alemán escaneó el cómic y lo colgó en la red el 1 de septiembre de 2008 [22].

El navegador se lanzó al público el 2 de septiembre de 2008. Se publicó una versión beta para Microsoft Windows (XP y versiones posteriores) disponible en 43 idiomas. Ese mismo día surgió la polémica al publicarse una noticia con parte de los términos de uso de esta versión beta de Chrome. El problema era que, según estos términos, se concedía una licencia a Google sobre todo el contenido transferido a través del navegador Chrome. Google respondió a las críticas afirmando que esos procedían de otro software de Google, y eliminaron esa parte de los Términos de uso de Chrome.

Google Chrome comenzó a conseguir cuota de mercado, aunque con algunos altibajos. A principios de 2009 se anunció el lanzamiento de versiones de Chrome para Mac OS X y Linux. En diciembre de 2009 se lanzó la versión beta de Chrome para Mac OS X y Linux. La primera versión estable de Chrome con soporte para las tres plataformas fue Google Chrome 5.0, lanzado en mayo de 2010.

Chrome fue ensamblado a partir de 25 bibliotecas de código diferentes, tanto de Google como de terceros, entre ellas, por ejemplo, Mozilla Netscape Portable Runtime o NPAPI.

En febrero de 2012, Google lanzó Chrome Beta para dispositivos android, y desde la versión 4.1 de Android, Chrome viene preinstalado y es el navegador por defecto en muchos de estos dispositivos.

En menos de 7 años de historia, el navegador Google Chrome ha conseguido convertirse en el

más utilizado a nivel mundial, y además, con Chrome se han introducido múltiples funcionalidades en la web y se ha cambiado el concepto del propio navegador web. Actualmente la última versión estable de Google Chrome es la versión 41.0.2272.

2.1.3.2. Características.

Google Chrome utiliza una interfaz de usuario bastante minimalista. Por ejemplo, ha unido la barra de direcciones y la barra de búsqueda en un solo cuadro multifunción. Con el tiempo este tipo de interfaz se ha extendido a otros navegadores. Chrome además tiene la reputación de ser un navegador con un gran rendimiento [23].

Chrome permite a sus usuarios la sincronización del historial, los marcadores y la configuración del navegador a través de todos los dispositivos con el navegador instalado. Esto se realiza mediante el envío y la recepción de datos a través de la cuenta de Google del usuario, desde la que se actualiza la información de la cuenta a todas las instancias del navegador registradas con dicha cuenta.

De cara a la seguridad, Chrome mantiene actualizadas dos listas negras, una para phishing (o robo de identidad) y otra para malware, y advierte a los usuarios cuando intentan visitar un sitio potencialmente peligroso. Este servicio está disponible en el API gratuito Google Safe Browsing API. Chrome asigna un entorno limitado a sus pestañas, aplicando el principio de privilegios mínimos. Esto hace que una pestaña no pueda interactuar con las funciones críticas de memoria, como por ejemplo la memoria del sistema operativo o los archivos de usuario, o que no influya en los procesos de otras pestañas. Es similar al modo protegido que Microsoft utilizó en Internet Explorer 9 y versiones posteriores.

Una prueba de la seguridad de Chrome es que en el concurso Pwn2Own, en el que se intenta encontrar vulnerabilidades para software de uso extendido, no se consiguieron detectar vulnerabilidades en las ediciones 2009, 2010 y 2011. En la edición de 2012 sí se consiguieron encontrar vulnerabilidades en Google Chrome, y Google felicitó oficialmente a los investigadores que las detectaron y reconoció oficialmente el mérito de su trabajo. Sin embargo a las 10 horas ya se habían desarrollado las correspondientes correcciones para estas vulnerabilidades.

2.1.3.3. Funcionamiento de los plugins.

Desde sus comienzos Google Chrome dio soporte a los plugins basados en NPAPI. De hecho, NPAPI fue una de las 25 bibliotecas de código a partir de las cuales se ensambló Chrome. Pero con el tiempo la política de Google Chrome hacia NPAPI y los plugins ha cambiado mucho [24].

En marzo de 2010 Google anunció que las siguientes versiones de Chrome incluirían Ado-

be Flash dentro del navegador, eliminando la necesidad de descargarlo e instalarlo por separado. Flash además se mantiene actualizado como una parte de Chrome con sus propias actualizaciones.

En agosto de 2009 Google presentó un sustituto para NPAPI, PPAPI (Pepper Plugin API) buscando un API más seguro y más portable. En las sucesivas versiones de Chrome se ha ido sustituyendo progresivamente NPAPI por PPAPI. En septiembre de 2013 Google anunció que el soporte para NPAPI sería retirado. El soporte para NPAPI en Linux se ha retirado desde la versión 35 de Chrome, y en otras plataformas se irá retirando de forma progresiva.

A pesar del nuevo API para los plugins PPAPI de Chrome, se está intentando eliminar o minimizar el uso de los plugins en los navegadores web. Firefox por ejemplo ha optado por la opción click to play, lo que significa para que el plugin comience a ejecutarse necesitará de la autorización del usuario, con esto se busca reducir las vulnerabilidades generadas en los navegadores por lo plugins. Sin embargo las intenciones de Chrome, y posiblemente de Mozilla a largo plazo, es prescindir de los plugins e integrar sus funcionalidades dentro del navegador o de HTML.

2.1.3.4. API de los plugins.

PPAPI (Pepper Plugin API) es un API multiplataforma para plugins en navegadores web que utilizan la tecnología Google Native Client. En un principio se basó en NPAPI, pero posteriormente se ha reescrito con Scratch. Actualmente se utiliza en Google Chrome y en Chorium para habilitar las versión PPAPI de Flash y del visor de PDF.

Como hemos mencionado anteriormente, el proyecto PPAPI comenzó en agosto de 2009 como un sustituto para NPAPI. PPAPI se ha diseñado específicamente para facilitar una implementación de los plugins fuera de proceso. En un principio PPAPI buscaba una mayor uniformidad y la estandarización en el desarrollo, registro y ejecución de los plugins, pero Mozilla no mostró interés en este proyecto, y ya hemos mencionado la política que Google Chrome ha tomado con los plugins.

2.2. Tecnologías de reproducción de vídeo.

2.2.1. HTML5.

Con HTML5 se han introducido una serie de funcionalidades para la reproducción y el tratamiento de vídeo, que las anteriores versiones de HTML no incluían. La versión definitiva de HTML5 se publicó en octubre de 2014 [25].

2.2.1.1. Historia.

Aunque hasta 1991 no se publicó el primer documento oficial de HTML (HTML Tags) en el que se realizaba una descripción del propio lenguaje, el proyecto comienza en 1980. Tim Berners-Lee,

en ese momento, un físico que trabajaba para el CERN (Organización Europea para la Investigación Nuclear), decide crear un sistema de hipertexto para compartir documentos. Después de crear su sistema de hipertexto y con la ayuda de Robert Cailliau, presentaron WorldWideWeb (W3) un sistema de hipertexto para Internet [26].

IETF (Internet Engineering Task Force) realizó en 1993 la primera propuesta oficial para que HTML se convirtiese en un estándar. A pesar de avances como las etiquetas para tablas, formularios e imágenes, ninguna de las dos propuestas (HTML y HTML+) se convirtió en estándar. Gracias al trabajo de IETF con HTML el 22 de septiembre de 1995 se consigue publicar el primer estándar oficial de HTML. A pesar de ser el primer estándar, se publicó como HTML 2.0.

Desde 1997 W3C (World Wide Web Consortium) se encargó de publicar los estándares de HTML. El 14 de enero de 1997 W3C publica por primera vez un estándar de HTML, con la versión 3.2 de HTML. Esta versión incluye novedades como los applets de Java. En diciembre de 1997 se publicó HTML 4.0. Esta versión incluye grandes avances como las hojas de estilo CSS o la opción de incluir scripts en las páginas web. HTML 4.0 ofrecía tres alternativas: transicional (permite elementos obsoletos), estricta (no permite elementos obsoletos) y conjunto de marcos (para las páginas web formadas por frames).

En diciembre de 1999 se publicó HTML 4.01. Con esta versión se buscaba revisar y actualizar la versión 4.0, por lo que no incluía novedades. Después de esta versión, W3C dejó su labor en la estandarización de HTML y se centró en desarrollar un estándar de XHTML. Tras varios años sin nuevos estándares de HTML por parte de W3C, en 2004 Apple, Mozilla y Opera decidieron crear la asociación WHATWG (Web Hypertext Application Technology Working Group) [27].

WHATWG se centró en la creación de un estándar de HTML5, publicando a principios de 2008 un primer borrador. WHATWG y este nuevo estándar buscaban dar a la web la funcionalidades que hasta ese momento solo se conseguían con Flash. La dependencia de una tecnología como Flash es contraria a la filosofía de la web, y más teniendo en cuenta que Flash es una herramienta de pago (para la creación de contenido). En esta nueva especificación de HTML se añaden nuevas funcionalidades en el apartado multimedia, permitiendo la creación de contenido en 2 y 3 dimensiones con Canvas o el uso de gráficos vectoriales, vídeo y audio. Debido a los avances de WHATWG y la publicación de los borradores de HTML5, W3C retomó la estandarización de HTML.

En el siguiente apartado vamos a analizar las nuevas funcionalidades que incluye HTML5, especialmente las relacionadas con el apartado multimedia y en especial el uso del vídeo, que son las de mayor interés para este proyecto.

2.2.1.2. Características.

HTML5 es el último estándar de HTML. Esta nueva versión del lenguaje añade nuevos elementos y tecnologías, lo que permite una mayor diversidad y un mayor alcance de la web. Este nuevo conjunto de tecnologías se llama HTML5 y amigos, aunque se suele abreviar a HTML5. Estas tecnologías han sido clasificadas en varios grupos según sus funcionalidades: semántica, desconectado y almacenamiento, dispositivos de acceso, conectividad, rendimiento e integración, CSS3, gráficos y efectos 2D/3D y multimedia. En el desarrollo de este trabajo nos centraremos en las tecnologías del apartado multimedia de HTML5 que son de interés para nuestro proyecto.

Con la introducción de los elementos `<audio>` y `<video>` en HTML5 podemos insertar contenido multimedia en los documentos HTML. Con estos elementos se sustituye parcialmente al elemento `<object>`. De cara a este proyecto el elemento de mayor interés es `<video>`, cuyo propósito es la reproducción de vídeo.

El elemento `<video>` fue propuesto por Opera Software a principios de 2007. Opera también realizó un prototipo y creó un manifiesto para recalcar la importancia que tendría el elemento `<video>` en la web.

HTML5 video se ha creado con la intención de ser un estándar para la reproducción de vídeo en la web sin la necesidad de utilizar plugins. Con esto se busca sustituir a Adobe Flash Player al que anteriormente podíamos considerar como el estándar para la reproducción de vídeo en la web. La estandarización de HTML5 video está siendo obstaculizada por la falta de acuerdo sobre qué formatos de codificación de vídeo deben soportar los navegadores web.

HTML5 no especifica qué formatos de vídeo deben soportar los navegadores; pueden soportar cualquier formato de vídeo que consideren apropiado, pero los proveedores del contenido deben asumir que los vídeos pueden no ser accesibles a todas los navegadores, al menos hasta que no se fije un conjunto mínimo de formatos de vídeo que deban ser soportados.

El grupo de trabajo de HTML5 dio una serie de pautas de cara a la elección del formato de vídeo, como por ejemplo que tenga una buena calidad de imagen o una buena compresión, que sea royalty-free o libre de derechos. Recomienda además que el formato de vídeo disponga de un decodificador hardware de vídeo además de los decodificadores software.

HTML5 tiene como ventaja que cualquier navegador soporta HTML, además con los elementos `<audio>` y `<video>` es trivial insertar contenido multimedia en un documento HTML. Pero la falta de acuerdo acerca de los formatos de codificación de vídeo que deben ser soportados, están perjudicando la estandarización y la difusión de HTML5 video.

2.2.2. Flash Player.

Adobe Flash Player es un reproductor multimedia, que permite reproducir archivos en formato SWF. Estos archivos pueden ser creados con Adobe Flash, Adobe Flex y otras herramientas de Adobe. Adobe Flash Player puede funcionar como una aplicación del sistema si se instala directamente en el sistema operativo, o como un plugin o un objeto ActiveX si se instala en el navegador.

2.2.2.1. Historia.

En 1996 Flash fue publicado por Macromedia como una herramienta de animación que también permitía opcionalmente su uso como un plugin para navegadores. El proyecto de Flash comienza antes de su compra y publicación por Macromedia, con los trabajos de Jonathan Gay y Charlie Jackson en la creación de videojuegos y programas para la creación de gráficos. Lo que les lleva a la creación de la compañía FutureWave Software, en la que entre otro software desarrollaron un plugin para Netscape [28].

Desde FutureWave Software se estaba desarrollando el proyecto de animación vectorial FutureSplash Animator. Con la compra de FutureWave Software por parte de Macromedia este programa pasa a llamarse Macromedia Flash 1.0. Esto permitió incluir en las páginas web entornos gráficos vectoriales e interactivos como botones y paneles de navegación entre otros.

En un principio se consideró que Flash tendría poco éxito porque se consideraba una tecnología difícil de manejar, aunque con el tiempo Flash aumentó su popularidad y cada vez más usuarios desarrollaban sus gráficos con esta herramienta. En 1997 ya se publica Macromedia Flash 2 y en 1998 la versión 3 que incluía mejoras en la animación, la reproducción y permitía obtener mayor interactividad.

En mayo de 1999 se publica Macromedia Flash 4.0, que introduce el streaming MP3 y la interpolación de movimiento. El plugin Flash Player no se incluía inicialmente en los navegadores web más utilizados y había que descargarlo desde el sitio web de Macromedia. Sin embargo, a partir del año 2000 Netscape e Internet Explorer comenzaron a distribuir Flash Player. Posteriormente Flash Player fue incluido como complemento en todas las versiones de Windows XP. La popularidad de Flash creció hasta conseguir que el 92 % de los usuarios de internet tuviesen instalado este software.

En agosto de 2000 se publica Macromedia Flash 5, que supuso un gran avance gracias a la mejora de las capacidades de scripting de Flash. En octubre de este mismo año Jakob Nielsen escribió un polémico artículo criticando la usabilidad de los contenidos de Flash. Lo que hizo Macromedia fue contratar a Jakob Nielsen para la mejora de la usabilidad de Flash.

En mayo de 2005 Adobe compra Macromedia y en septiembre de ese mismo año publica Flash

8 con novedades como un simulador de dispositivos móviles o vídeo con canal alfa (On2 VP6), entre otros avances.

Con el tiempo Flash se ha convertido en la opción más utilizada para crear aplicaciones web con audio, vídeo e interactividad. Por otro lado Adobe Flash Player llegó a convertirse en un estándar para la reproducción de vídeo en la web. Actualmente está disponible para Mozilla Firefox, Internet Explorer, Opera o Safari, entre otros. Google Chrome no necesita Adobe Flash Player, porque Google publica su propia versión con el programa.

Hasta la publicación de HTML5, para poder mostrar un vídeo desde el navegador era necesario algún tipo de plugin. Aunque existen algunas alternativas de código abierto, que posteriormente detallaremos, Adobe Flash Player ha sido la tecnología más utilizada para la reproducción de vídeo en la web. Como anteriormente comentamos, se convirtió prácticamente en un estándar, pero también han surgido algunos problemas. En 2010 Adobe Flash Player y la tecnología Flash en general fueron criticados públicamente por Apple. Además la falta de soporte en sistemas Linux para Adobe Flash Player ha hecho que surjan alternativas de código abierto. Y por último, la irrupción de HTML5 y la política que algunos navegadores como Chrome han tomado hacia los plugins, hace que el éxito que hasta ahora ha tenido Adobe Flash Player esté en peligro.

2.2.2.2. Características.

Flash Player puede ser ejecutado desde un navegador como un plugin o también está disponible como una aplicación independiente para los sistemas operativos Windows y Mac OS X, aunque esta aplicación está destinada principalmente para desarrolladores. Como ya mencionamos anteriormente, Flash Player permite reproducir archivos en formato SWF, ya sean creados con herramientas de Adobe o de terceros [29].

Flash Player soporta gráficos vectoriales, gráficos en 3D, el lenguaje de scripting ActionScript y streaming tanto de audio como de vídeo. ActionScript está basado en ECMAScript y tiene soporte para código orientado a objetos.

Adobe Flash Player ejecuta y muestra el contenido proporcionado por un archivo SWF, aunque no tiene funcionalidades para modificar los archivos SWF en tiempo de ejecución. Puede ejecutar software escrito en el lenguaje ActionScript que le permite manipular datos, gráficos vectoriales, texto, sonido y vídeo en tiempo de ejecución. Flash Player además permite acceder a los dispositivos de hardware que tengamos conectados, como por ejemplo webcams o micrófonos, siempre y cuando el usuario lo permita.

Flash Player es utilizado internamente por Adobe Integrated Runtime (Adobe AIR), para proporcionar un entorno multiplataforma en tiempo de ejecución para las aplicaciones de escritorio y las aplicaciones para dispositivos móviles. Adobe AIR tiene soporte para aplicaciones instalable

en Windows, Mac OS X, Linux y en algunos sistemas operativos móviles como iOS y Android. Las aplicaciones Flash deben estar construidas específicamente para Adobe AIR para poder utilizar las funcionalidades adicionales como la integración del sistema de archivos, extensiones nativas, integración nativa con la pantalla o integración hardware con acelerómetros o dispositivos GPS conectados.

Flash Player incluye soporte nativo para múltiples formatos de datos:

- **AMF:** Flash Player permite el almacenamiento de cookies en los ordenadores de los usuarios en forma de objetos locales compartidos (el equivalente de Flash a las cookies de los navegadores). Flash Player además puede leer y escribir de forma nativa archivos en Action Message Format, el formato de datos por defecto de los objetos locales compartidos (Local Shared Objects).
- **JSON:** Flash Player 11 incluye soporte nativo para importar y exportar datos en formato JavaScript Object Notation (JSON). Esto permite tener interoperabilidad con los servicios web y los programas JavaScript.
- **SWF:** la especificación para el formato de archivo SWF fue publicado por Adobe, permitiendo el desarrollo del formato SWX. Este formato utiliza archivos SWF y AMF para, desde aplicaciones Flash, intercambiar datos con aplicaciones del lado del servidor.
- **XML:** desde la versión 8 de Flash Player se ha incluido soporte nativo para la generación y el análisis de XML.

Flash Player es principalmente una plataforma multimedia y para gráficos que ha tenido soporte para mapas de bits y gráficos vectoriales desde sus primeras versiones. Tiene soporte para los siguientes formatos multimedia que puede decodificar y reproducir de forma nativa:

- **FLV:** Flash Player tiene soporte para decodificar y reproducir vídeo y audio de archivos Flash Video (FLV y F4V). Este formato fue desarrollado por Adobe Systems y Macromedia. Flash Video es un formato contenedor y soporta múltiples codecs de vídeo, como Sorenson Spark, VP6 o H.264.
- **GIF:** las imágenes comprimidas en formato GIF (Graphics Interchange Format) pueden ser renderizadas y decodificadas desde Flash Player, aunque solo la primera imagen del GIF.
- **JPEG:** Flash Player tiene soporte para decodificar y renderizar imágenes comprimidas en JPEG. Desde Flash Player 10 se incluye soporte para JPEG-XR, un avanzado estándar de compresión de imagen desarrollado por Microsoft Corporation, que permite una mejor compresión y calidad de imagen que JPEG.
- **MP3:** desde la versión 4 de Flash Player se ha incluido soporte para MP3. Los archivos MP3 pueden ser accedidos y reproducidos desde un servidor vía HTTP, o embebidos en un archivo SWF, que también es un formato de streaming.

- **PNG:** las imágenes en formato PNG (Portable Network Graphics) pueden ser decodificadas y renderizadas por Flash Player tanto en la variante opaca de 24 bit como en la semitransparente de 32 bits.

Además soporta streaming a través de los protocolos HTTP, RTMP y TCP.

2.2.3. Otras alternativas para la reproducción de vídeo.

2.2.3.1. Gnash.

Gnash es un reproductor multimedia que permite reproducir archivos en formato SWF. Gnash está disponible como un reproductor independiente tanto para ordenadores como para sistemas embebidos. Además está disponible como un plugin para varios navegadores. Es parte del proyecto GNU y es una alternativa libre y de código abierto a Adobe Flash Player. Es desarrollado por el proyecto GameSWF. Gnash fue anunciado a finales de 2005 por el desarrollador de software John Gilmore y actualmente es mantenido por Rob Savoye [30].

Desarrollar una alternativa a Flash Player de código libre ha sido una prioridad para el proyecto GNU desde hace un tiempo. Antes de la aparición de Gnash el proyecto GNU estaba buscando desarrolladores para el proyecto GPLFlash. La mayoría de los desarrolladores del proyecto GPLFlash han pasado al proyecto Gnash y el código de GPLFlash se ha redirigido al soporte de sistemas embebidos.

Gnash está distribuido bajo los términos de la licencia GNU GPL. Aunque desde que Gnash comenzó a utilizar el código base del proyecto GameSWF, que es de dominio público, el código desarrollado por el proyecto Gnash, que puede ser de utilidad para GameSWF, también es de dominio público.

Adobe únicamente ofrece para Linux una versión obsoleta (11.2) en IA-32 y una versión preliminar en formato binario para AMD64. Sin embargo, Gnash puede ser compilado y ejecutado en múltiples arquitecturas como x86, ARM, PowerPC y MIPS. Además es compatible con los sistemas operativos basados en BSD.

Gnash necesita AGG, Cairo o OpenGL para la realización del renderizado. Al contrario que la mayoría de los proyectos GNU, que normalmente están escritos en C, Gnash está escrito en C++ debido a su relación con GameSWF. Actualmente Gnash puede reproducir archivos SWF hasta la versión 7 (y parte de las versiones 8 y 9) y el 80% de ActionScript 2.0. Por otra parte Gnash ofrece algunas funcionalidades que Adobe Flash Player no ofrece, como la extensión de clases de ActionScript a través de bibliotecas compartidas [31].

Gnash permite la reproducción de vídeos en formato FLV así como reproducir algunos archivos FLV de YouTube, Myspace o webs similares. Para poder soportar el formato FLV es necesario

tener instalado FFmpeg o GStreamer. De momento con Gnash no se pueden reproducir múltiples archivos que con Flash Player sí se puede, pero puede funcionar como plugin en Konqueror, Chorium, Firefox y SeaMonkey.

2.2.3.2. Lightspark.

Lightspark es un reproductor de archivos en formato SWF libre y de código abierto, publicado bajo los términos de la licencia GNU LGPL v3 (Lesser General Public License) [32]. Lightspark soporta ActionScript 3.0 y tiene un plugin compatible con Mozilla. Lightspark utiliza Gnash para la reproducción de SWF más antiguos que utilicen código ActionScript en las versiones 1.0 o 2.0.

El reproductor Lightspark es completamente portable. Se ha instalado satisfactoriamente en Ubuntu 11.04 en las arquitecturas PowerPC, x86, ARM y AMD64. Lightspark dispone de una rama Win32 para Microsoft Visual Studio y creó un plugin compatible con Mozilla en Windows en la versión 0.5.3, pero desde entonces no se han publicado nuevas versiones para Windows.

La última gran versión de Lightspark es la versión 0.7.2 publicada en marzo de 2013. Desde entonces se han publicado varias subversiones. La última ha sido 0.7.2-6 en octubre de 2014. Desde la versión 0.4.5.1 publicada en noviembre de 2010 las funcionalidades de Lightspark han aumentado considerablemente.

Lightspark aprovecha el feedback con sus usuarios para solventar los errores que surgen en el proyecto. Con el tiempo Lightspark está consiguiendo cada vez más funcionalidades de las disponibles en Flash Player. Lightspark soporta renderizado basado en OpenGL y la ejecución de ActionScript basada en LLVM. El reproductor es compatible con los vídeos Flash en formato H.264 de YouTube. Además ha mejorado su soporte para imágenes en múltiples formatos, soporta el reproductor de vídeo de la BBC y ha conseguido algunas de las funcionalidades utilizadas por Flash Player para los juegos.

2.2.3.3. Swfdec.

Swfdec es una alternativa obsoleta libre y de código abierto para Adobe Flash Player. Funciona en Linux y FreeBSD y se distribuye bajo la licencia GNU LGPL (Lesser General Public License). La última versión fue publicada en diciembre de 2008 [33].

Swfdec es una biblioteca que se puede utilizar para reproducir archivos de Flash. La biblioteca puede ser utilizada tanto por un reproductor, como por el plugin de Mozilla. Swfdec soporta la versión 4 de Flash y la mayoría de las funcionalidades hasta la versión 9. El reproductor se fue actualizando para soportar las funcionalidades requeridas por YouTube, Google Video o CNN video.

En 2007 Swfdec fue elegido como el sustituto de Flash Player para Fedora [34]. Además fue

portado a DirectFB para ser embebido con X11 y GTK+. Utiliza la biblioteca gráfica Cairo para el renderizado, GStreamer para la decodificación de audio y vídeo, y PulseAudio, OSS o ALSA para la reproducción de audio.

El desarrollo de Swfdec prácticamente no ha avanzado. A día de hoy la última actualización de su repositorio en Git es de diciembre de 2009.

2.2.3.4. Shumway.

Shumway es una alternativa de código abierto para Adobe Flash Player. Ha sido desarrollado por Mozilla desde 2012. Es la evolución de un proyecto llamado Gordon, en alusión a Flash Gordon y Gordon Shumway [35]. Shumway renderiza el contenido de Flash traduciendo el contenido dentro de los archivos de Flash a elementos de HTML5, ActionScript y JavaScript.

Shumway busca la ejecución de contenido de tipo Flash en el navegador sin tener que utilizar plugins. Esto ayudaría a reducir el consumo de memoria por parte del navegador y en teoría podría eliminar los potenciales riesgos de seguridad que Adobe Flash Player. Los tipos de ataque contra los plugins obviamente desaparecerían, aunque el rendimiento puede verse ligeramente afectado.

El funcionamiento de Shumway sería similar al de pdf.js, que permite visualizar ficheros en formato PDF dentro del navegador Firefox, empleando el motor del propio navegador. Un proyecto similar a Shumway es Google Swiffy.

2.2.3.5. VLC.

VLC es un reproductor multimedia multiplataforma libre y de código abierto que además puede ofrecer servicio de streaming. VLC soporta múltiples formatos tanto de audio como de vídeo. Además dispone de muchas librerías de codificado y decodificado. VLC es desarrollado por el proyecto VideoLAN [36].

El proyecto VideoLAN comenzó en 1996 como un proyecto académico. VLC en un principio significaba VideoLAN Client. La idea era crear un cliente y un servidor de vídeo en streaming para la red del campus en la Escuela Central de París. Reescrito con Scratch en 1998 y publicado bajo la licencia GNU GPL en 2001, las funcionalidades del programa servidor VideoLAN Server (VLS) se han incluido en VLC y el proyecto se quedó con el nombre de VLC media player dado que ya no existía una arquitectura de cliente/servidor.

Después de años de desarrollo, la versión 1.0.0 de VLC se publicó en julio de 2009. La versión 2.0.0 se publicó en febrero de 2012. Hoy en día VLC está disponible para múltiples plataformas, incluidos dispositivos móviles.

La parte que más nos interesa de VLC es que ofrece un plugin basado en NPAPI para Windows, Mac OS X, Linux y otras plataformas basadas en Unix. Este plugin permite visualizar el contenido embebido en las páginas web para los formatos de archivo de Windows Media, Ogg, QuickTime y MP3, sin la necesidad de tecnologías adicionales. A partir de la versión 0.8.2, VLC también ofrece un plugin para Internet Explorer que proporciona estas funcionalidades para Internet Explorer.

Otras aplicaciones han utilizado el plugin de VLC para aprovechar ventajas como el poder soportar archivos incompletos o permitir la vista previa de archivos que se están descargando. Estas funcionalidades se han utilizado en eMule y KCeasy. La aplicación libre de código abierto Miro, que permite ver la televisión a través de Internet, también utiliza código de VLC. Desde el proyecto Google Videos, un proyecto abandonado por Google, se creó Google Video Player basándose en VLC, aunque este proyecto no tuvo continuidad tras la compra de Youtube por parte de Google. Hay algunos ejemplos más de proyecto que utilizan el código de VLC o utilizan alguna librería de su API, como LibVLC.

2.2.4. Frameworks para la creación de plugins.

2.2.4.1. FireBreath.

FireBreath es un Framework que permite la creación de potentes plugins para navegadores. Un plugin construido con FireBreath funciona como un plugin basado en NPAPI o como un control de ActiveX (solo en Windows). FireBreath también tiene soporte para añadir otros tipos de plugin, como los construidos con C++.

El proyecto de FireBreath comienza en 2009, iniciado por Richard Bateman [37]. Su idea era crear una herramienta para facilitar la implementación de plugins para navegadores. Se trata de un proyecto de código abierto, y aunque Bateman es el principal desarrollador, para la creación de FireBreath han contribuido otros desarrolladores y empresas.

FireBreath pretende ser una arquitectura multiplataforma para los plugins, para las siguientes tecnologías y navegadores [38]:

- Navegadores NPAPI en Windows, Mac y Linux:
 - Gecko/Firefox.
 - Google Chrome (en lugar de NPAPI utiliza PPAPI, de los cuales ya hemos explicado anteriormente las diferencias).
 - Apple Safari.
 - Opera (con algunos problemas).
- Controles de ActiveX:
 - Microsoft Internet Explorer 6, 7, 8, 9, 10 y versiones posteriores.

FireBreath está distribuido bajo una licencia dual, lo que nos permite escoger entre dos tipos de licencias para su uso. FireBreath puede ser usado bajo la nueva licencia BSD o la licencia GNU LGPL (Lesser General Public License).

Para crear un plugin con FireBreath desde Windows es necesario tener instalado el siguiente software:

- Visual Studio 2005 o posterior.
- CMake 2.8.7 o alguna versión más avanzada.
- Git.
- Python 2.7 o similar (con versiones superiores a 3.0 da problemas).

Descargando con Git el código de FireBreath y siguiendo el tutorial de la página de FireBreath (para Windows y Mac) nos permite, ejecutando el archivo `fbgen.py`, crear una plugin inicial. Si registramos, en el caso de Windows, el DLL generado y abrimos con el navegador el archivo htm creado, podemos ver que el plugin funciona y muestra varios mensajes. A partir de este plugin inicial podemos crear nuestros propios plugins.

FireBreath proporciona además algunos ejemplos de código y documentación para el desarrollo de plugins. Hemos probado a crear el plugin inicial y se ha probado su correcto funcionamiento en varias versiones de Windows con Internet Explorer, Mozilla Firefox y Google Chrome.

2.2.4.2. JUCE.

JUCE (Jules Utility Class Extensions) es un Framework multiplataforma de código abierto, utilizado para el desarrollo de aplicaciones, tanto de escritorio como móviles en C++ . JUCE se utiliza especialmente para el desarrollo de interfaces gráficas y librerías de plugins [39].

JUCE busca que el software sea desarrollado de forma que el mismo código fuente pueda ser compilado y ejecutado de la misma forma en Windows, Mac OS X y en sistemas Linux. Soporta varios entornos de desarrollo y compiladores, como GCC, Xcode, Visual Studio y Code::Blocks.

La primera versión de JUCE se publicó en 2004 y ha sido mantenido por Jules Storer de Raw Material Software. JUCE se distribuye bajo una licencia dual GPL(General Public License)/Comercial. ROLI compró JUCE y Raw Material Software en noviembre de 2014. ROLI es una empresa dedicada a la creación de hardware y software para múltiples tecnologías, aunque está especialmente relacionada con la industria musical, de ahí su interés en JUCE.

JUCE contiene una serie de clases que permiten crear aplicaciones estándar. Estas clases son especialmente útiles para el desarrollo de aplicaciones en tiempo real y de audio. JUCE tiene

una buena reputación gracias a la calidad y la consistencia de su código base. Además ha sido acreditado por grandes compañías musicales como Korg, MAudio y Arturia.

Además de las clases, JUCE ofrece soporte para enlazar y gestionar bibliotecas y dependencias con el software Introjucer. Este software crea plantillas para los entornos de compilación más utilizados, permitiendo que la compilación sea más rápida y sencilla.

JUCE tiene soporte para las siguientes plataformas y compiladores:

- Plataformas:
 - Microsoft Windows XP, Windows Vista, Windows 7, Windows 8 y 8.1.
 - Mac OS X para la versión 10.5 y posteriores.
 - iOS versión 3 y posteriores.
 - Linux para versiones del kernel de la serie 2.6 y posteriores.
 - Android, usando NDK-v5 y posteriores.
- Compiladores:
 - GCC versión 4.0 y posteriores.
 - LLVM.
 - Microsoft Visual Studio - Visual C++ 2005 y posteriores.
 - MinGW.

Al igual que otros Frameworks como Qt o GTK+, JUCE contiene múltiples clases ofreciendo un amplio rango de funcionalidad para las interfaces de usuario, elementos de audio y gráficos, análisis de XML y JSON, networking, criptografía o aplicaciones multihilo. Esto pretende reducir el número de bibliotecas de terceros que necesitan utilizar los desarrolladores de aplicaciones.

JUCE ofrece clases para la creación de plugins para navegadores y de audio [40]. Cuando se crea un plugin de audio, un único archivo soporta múltiples formatos (VST, VST3, Audio Units, AAX, RTAS). Con los plugins para navegadores pasa algo similar: un único archivo puede servir como un plugin basado en NPAPI o un control de ActiveX.

2.2.4.3. Qt.

Qt es un Framework multiplataforma muy utilizado para el desarrollo de aplicaciones de software que pueden funcionar en distintos software y hardware sin necesidad de modificar el código base o realizando solo pequeños cambios [41]. Qt actualmente está desarrollado tanto por Qt Company (filial de Digia) como por el proyecto Qt, un proyecto de código abierto. Qt está disponible con varios tipos de licencias, tanto la licencia comercial, como las licencias de código abierto GPL

v3, LGPL v3 y LGPL v2.

El desarrollo de Qt comenzó en 1991 de la mano de Haavard Nord y Eirik Chambe-Eng. En 1992 Qt surgió como una biblioteca desarrollada por Trolltech (ahora Qt Development Frameworks), con un desarrollo basado en el código abierto, pero sin ser completamente libre. Entre 1996 y 1998 se utilizó Qt para el desarrollo del escritorio KDE. Esto generó preocupación en el proyecto GNU, ya que veían como un peligro para el software libre que se desarrollase un escritorio libre tan popular como KDE con un software propietario. Después de algunas críticas y controversias, y de la creación de KDE Free Qt Foundation, Qt se publica bajo las licencias anteriormente mencionadas. KDE Free Qt Foundation se creó para garantizar la disponibilidad de una versión libre y gratuita de las bibliotecas Qt.

Qt está disponible para las siguientes plataformas:

- Microsoft Windows XP, Windows Vista, Windows 7, Windows 8 y 8.1.
- Mac OS X.
- iOS.
- Android, usando NDK-v5 y posteriores.
- Sistemas tipo unix con X Window System (Linux, BSDs, Unix).
- Y otros sistemas como: Integrity, QNX, BlackBerry 10, VxWorks, Wayland, Windows CE y Windows RT.

Hay cuatro ediciones de Qt disponibles: Community, Indie Mobile, Professional y Enterprise, además hay múltiples versiones de Qt.

Qt es un Framework muy extendido, utilizado para la creación de software muy conocido como Google Earth, VirtualBox, VLC media player o Skype. Además es utilizado por grandes empresas y organizaciones como la Agencia Espacial Europea, Samsung o Volvo entre otras. De los tres Frameworks analizados en este proyecto, Qt es el más extendido y conocido, pero a la vez es el más general, y por lo tanto, menos específico de cara al desarrollo de plugins. Aunque dispone de las herramientas QtBrowserPlugin [42], Qt Netscape Plugin Extension [43] y ActiveQt para facilitar el desarrollo de plugins y controles de ActiveX para navegadores.

2.3. Vídeo a través del navegador

2.3.1. Mecanismos de reproducción de vídeo

2.3.1.1. Streaming

Streaming es un método de distribución de contenido multimedia que permite la reproducción del contenido de forma simultánea a la descarga del mismo, sin necesidad de tener completamente descargado el archivo que queremos reproducir [44]. Se trata de almacenar en un buffer de datos el contenido que se va descargando y al mismo tiempo reproducir la parte ya descargada. Con una velocidad de descarga suficiente se puede reproducir el contenido completo de un archivo sin interrupciones.

El streaming funciona siguiendo los siguientes pasos:

- El cliente conecta con el servidor y comienza la transmisión del contenido.
- El cliente empieza la recepción del archivo y lo almacena en un buffer.
- Cuando una parte del contenido está almacenado en el buffer, el cliente lo reproduce a la vez que continúa de forma simultánea la descarga de contenido.
- Aunque la velocidad de descarga descienda o se frene, el cliente puede seguir mostrando el contenido a partir de la información que se ha ido almacenando en el buffer. Si se consume toda la información del buffer, la reproducción deberá interrumpirse hasta que se descargue nuevo contenido.

El término Streaming no se refiere al contenido por sí mismo, sino a la forma en que se distribuye o entrega el contenido. La distribución de contenido multimedia a través de la red no es el único ejemplo de streaming. La radio o la televisión son ejemplos de medios de comunicación que transmiten su contenido sin necesidad de que esté completamente descargado [45].

A través de streaming se puede distribuir cualquier tipo de archivo, no solo los de tipo multimedia, aunque debido al tamaño de los archivos multimedia resulta de mayor interés poder reproducirlos sin necesidad de su completa descarga. Por ejemplo los archivos de texto podrían ser distribuidos por Streaming, pero no resulta de gran interés porque habitualmente son archivos de un tamaño reducido y descargarlos completamente requiere poco tiempo. Otra de las ventajas del streaming es que permite transmitir contenido tanto en directo como diferido.

Podríamos considerar como medios de distribución de contenido no streaming a cualquier transmisión de contenido en un formato físico, ya en un libro, un DVD o un USB. Dentro de Internet la mayor parte del contenido no se distribuye por streaming, ya sea la descarga directa de contenido, a través de P2P o una página web normal, que no muestra su contenido hasta que el archivo HTML no está completamente descargado. De esto último somos poco conscientes, puesto

que las páginas HTML se descargan como archivos temporales y habitualmente se cargan de forma instantánea, dándonos la sensación de no descargar contenido.

El desarrollo del Streaming comienza en los años 20, cuando se crearon una serie de patentes para un sistema transmisión y distribución de señales a través de líneas eléctricas. Esto fue la base técnica para la creación de Muzak, una tecnología Streaming que permitía transmitir música de forma continua. Este tecnología se utilizó sobretodo en ascensores y centros comerciales.

Durante varias décadas se realizaron pocos avances en este tipo de tecnologías. Desde finales de los 80 los ordenadores al alcance de los usuarios ya tenían suficiente potencia como para soportar un servicio de streaming. La potencia de CPU ya era adecuada y se crearon mecanismos de interrupción en los sistemas operativos para evitar la sobrecarga del buffer. Pero las redes eran bastante limitadas y el contenido normalmente se distribuía por medios no streaming, normalmente descargando el archivo a través de un servidor remoto [46].

Durante la década de los 90 y principios de los 2000 las mejoras en los servicios de Internet fueron considerables, como por ejemplo un mayor ancho de banda o la estandarización de protocolos y formatos como TCP/IP, HTTP y HTML. Esto permitió la comercialización de Internet. En 1993 se transmitió por primera vez un concierto en directo a través de Internet.

En 1995 Microsoft desarrolló el reproductor multimedia ActiveMovie, que permitía la transmisión de streaming y tenía un formato de archivo para streaming. Esto sirvió de base para el desarrollo de las funciones de streaming de Windows Media Player 6.4 en 1999. En 1999 también Apple introdujo un formato de archivo para streaming para QuickTime 4. Este formato fue utilizado en muchos sitios web junto con los formatos de streaming de Real Player y Windows Media. La competencia entre los sitios web y los formatos obligaba a los usuarios a tener tres aplicaciones distintas para poder reproducir todos los formatos.

Desde 2002 debido al interés en un formato de streaming unificado y a la gran difusión de Adobe Flash, se impulsó el desarrollo de un formato de vídeo para streaming a través de Flash. Este formato se utiliza en múltiples reproductores basados en Flash, en sitios tan populares como YouTube. Con el tiempo el streaming en directo ha tenido mucho éxito y YouTube implementó un servicio de streaming en directo para sus usuarios.

Gracias al aumento de potencia de los ordenadores de los usuarios comunes, al avance de las redes de ordenadores y la mejora de los sistemas operativos, hoy en día el streaming se ha convertido en un medio de comunicación práctico y al alcance de cualquier usuario de Internet. La popularidad de los servicios de streaming ha crecido con el tiempo y ha permitido la emisión de radio y televisión por Internet, además cada vez se emiten por streaming más eventos deportivos o conciertos, entre otros.

Habitualmente los archivos multimedia tienen un gran tamaño y son difíciles de almacenar y transmitir, por lo que este tipo de contenido se suele comprimir tanto para su almacenamiento como para su transmisión por streaming. El aumento de la transmisión de streaming en alta definición (HD) ha llevado al desarrollo de tecnologías como Wireless HD o ITU-T G.hn.

El contenido en streaming se puede transmitir en vivo o bajo demanda. Normalmente el contenido en vivo se conoce como streaming verdadero o auténtico streaming. Este tipo de streaming envía el contenido al ordenador o el dispositivo correspondiente sin guardar el archivo en el disco duro. El streaming bajo demanda también se conoce como streaming progresivo o descarga progresiva, que almacena el archivo en el disco duro y lo reproduce desde la localización correspondiente.

Los stream de audio se comprimen con un códec de audio como MP3, Vorbis o AAC, y los streams de vídeo se comprimen con un códec de vídeo como H.264 o VP8. Los streams de audio y vídeo codificados se ensamblan en un flujo de bits contenedor como MP4, FLV, WebM, ASF o ISMA. Este flujo de bits se proporciona desde un servidor de streaming a un cliente de streaming utilizando un protocolo de transporte como MMS o RTP.

Además de los usos anteriormente mencionados, se ha popularizado el uso de streaming en un ámbito más social. Por ejemplo, YouTube fomenta la interacción social en webcast a través de herramientas como chats en vivo o encuestas en línea. También se ha extendido el uso del streaming por parte de empresas sociales o para e-learning.

2.3.1.2. Códec

El uso de códecs es fundamental para el funcionamiento de un mecanismo de distribución de contenido como el streaming, que anteriormente hemos explicado. Los códecs están implicados en todas las etapas del tratamiento de vídeo, desde la grabación del vídeo, su edición, la codificación para su envío y su posterior decodificación para su uso. De estas dos últimas funciones surge el nombre códec, se trata de una abreviatura de codificador-decodificador [47].

Los códecs pueden estar desarrollados con software, hardware o con ambos, y son capaces de transformar el contenido de un stream (flujo de datos) para poderlo transmitir, cifrar o almacenar y posteriormente decodificarlo para que pueda ser reproducido. Los códecs son tecnologías de compresión que como anteriormente hemos mencionado consta de dos partes, un codificador para comprimir los archivos y un decodificador para descomprimirlo. Hay códecs para múltiples tipos de archivos: datos (PKZIP), imágenes (JPEG, PNG, GIF), audio (MP3, Vorbis, AAC) y vídeo (Cinepak, MPEG-2, VP8, H.264).

Hay dos tipos de códecs: los códecs con pérdidas de información y sin pérdidas de información. Los códecs sin pérdida de información (lossless) como PKZIP o PNG reproducen el contenido

original después de ser descomprimidos. Hay algunos códecs de vídeo sin pérdida como Apple Animation codec y Lagarith codec, pero no pueden comprimir el contenido lo suficiente como para distribuirlo a través de streaming.

Los códecs que utilizan una tecnología de compresión con pérdida de información (lossy) utilizan dos tipos de compresión: la compresión espacial (intraframe) y la compresión temporal (interframe). La compresión intraframe es básicamente la compresión de imagen aplicada al vídeo, se comprime cada fotograma por separado. Por ejemplo, Motion-JPEG utiliza solo compresión intraframe y codifica cada fotograma por separado como una imagen JPEG.

Por el contrario la compresión interframe utiliza la redundancia entre los fotogramas para comprimir el vídeo. Con la técnica interframe se almacenan solo algunos fotogramas completos, y para los fotogramas posteriores solo se almacena la información modificada respecto a los anteriores fotogramas. La compresión interframe es más eficiente que la compresión intraframe, por lo que la mayoría de los códecs están optimizados para buscar y aprovechar la información redundante de los fotogramas.

Los primeros códecs en utilizar estos métodos de compresión dividían los fotogramas en dos tipos: fotogramas clave y fotogramas delta. Los fotogramas clave se almacenan completos y se codifican con la compresión intraframe. Durante la compresión, los píxeles de los fotogramas deltas se comparan con los píxeles de los fotogramas anteriores y la información redundante se elimina. Los datos restantes de cada fotograma delta se comprimen con la técnica intraframe.

Las técnicas de compresión interframe han avanzado y la mayoría de los códecs, incluyendo MPEG-2, H.264 y VC-1, utilizan actualmente tres tipos de fotogramas para la compresión: fotogramas I, fotogramas B y fotogramas P. Los fotogramas I son equivalentes a los fotogramas clave. Se comprimen únicamente con técnicas intraframe y son los fotogramas de mayor tamaño y menor eficiencia. Los fotogramas B y P pertenecen a los fotogramas delta. Los fotogramas P son los más simples y pueden utilizar la información redundante en cualquier fotograma I o P anterior. Los fotogramas B son más complejos y pueden utilizar la información redundante de cualquier fotograma I, B o P tanto anterior como posterior. Esto hace a los fotogramas B los más eficientes de los tres tipos.

Estos tres tipos de fotograma se almacenan en grupos de imágenes (GOP), comenzando con un fotograma I e incluyendo todos los fotogramas B y P hasta el siguiente fotograma I, sin incluir este fotograma. Los códecs que utilizan los tres tipos de fotograma se suelen llamar formatos Long GOP. Con el aumento del uso de videocámaras es más habitual el uso de formatos de este tipo como MPEG-2 o H.264. Aunque algunos códecs como Cineform, Apple ProRes y Avid DNxHD utilizan únicamente técnicas de compresión intraframe para intentar mantener un alto nivel de calidad.

Es importante distinguir entre los códecs y los formatos contenedores, aunque en ocasiones comparten el mismo nombre, como en el caso del formato MPEG-4. Los formatos contenedores son formatos de archivos que pueden contener audio, vídeo, texto subtulado o metadatos asociados. Aunque hay formatos contenedores de uso general como QuickTime, la mayor parte de los formatos contenedores tienen un uso concreto.

Históricamente los códecs de vídeo han evolucionado del múltiples formas, aunque la mayoría de ellas han conducido a H.264. La Organización Internacional de Estándar y la Unión Internacional de Telecomunicaciones publicaron durante años distintos estándar, aunque desde hace años su estándar de códec de vídeo es H.264. Desde hace tiempo también las videocámaras utilizan formatos basados en H.264. Aunque la mayor parte de la emisión de televisión por cable aún se basa en MPEG-2, H.264 está aumentando su cuota de uso y en televisión por satélite es ampliamente utilizada. En cuanto al streaming, inicialmente los códecs propietarios dominaron el mercado, pero con el tiempo se ha extendido H.264.

2.3.1.3. Real Time Streaming Protocol

Real Time Streaming Protocol (RTSP) es un protocolo de control de red diseñado para su uso en sistemas de entretenimiento y comunicación, buscando controlar los servidores de contenido a través de streaming [48]. El protocolo se utiliza para establecer y controlar sesiones multimedia que permiten el intercambio de contenido entre distintos puntos. Los clientes de los servidores multimedia utilizan comandos como *play* y *pause* para facilitar el control en tiempo real de la reproducción de archivos multimedia desde el servidor.

RTSP fue desarrollado por RealNetworks, Netscape y la Universidad de Columbia. El primer proyecto se presentó a la IETF (Internet Engineering Task Force) en 1996. Fue estandarizado por MMUSIC WG (Multiparty Multimedia Session Control Working Group) de IETF y publicado como RFC 2326 en 1998. RTSP 2.0 se está desarrollando como un sustituto para RTSP 1.0. RTSP 2.0 se basa en la versión 1.0 pero no es compatible con las versiones anteriores, salvo en algunos mecanismos básicos [49].

Aunque en algunos aspectos es similar a HTTP, RTSP define secuencias de control útiles para controlar la reproducción de contenido multimedia. Al contrario que HTTP, RTSP tiene estado y utiliza un identificador cuando necesita realizar el seguimiento de varias sesiones concurrentes. Al igual que HTTP, RTSP utiliza TCP para mantener una conexión de extremo a extremo. La mayoría de mensajes de control en RTSP son enviados desde el cliente al servidor, aunque algunos mensajes son enviados en la dirección contraria.

El protocolo RTSP es no orientado a conexión, aunque, como anteriormente hemos mencionado, tiene estado y asocia un identificador para cada sesión con el servidor. Además de utilizar el

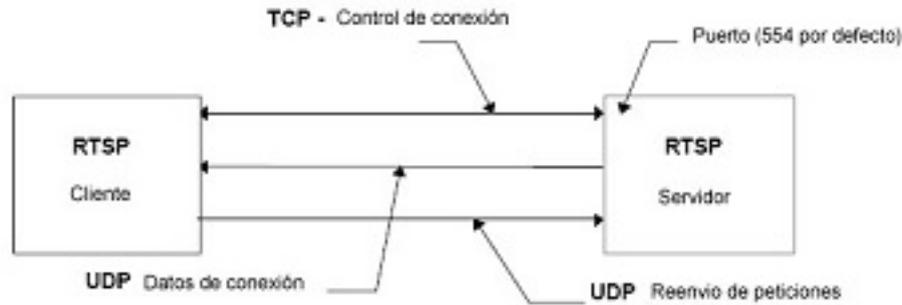


Figura 2.2: Protocolos en RTSP.

protocolo TCP para el control de la reproducción de contenido, se utiliza el protocolo UDP para el envío de los datos de audio y vídeo, aunque se podría utilizar TCP en caso de necesitar mayor fiabilidad en el envío de paquetes.

Las peticiones RTSP están basadas en las de HTTP y suelen ser enviadas desde el cliente al servidor, aunque al contrario que con el protocolo HTTP, el servidor también puede lanzar peticiones. Las cuatro peticiones básicas de RTSP son:

- **Setup:** esta petición especifica cómo se transportarán los datos. Se incluye un puerto para recibir los datos RTP de audio y vídeo y otro para los datos RTCP (metadatos). Se debe configurar cada flujo de datos con Setup antes de una petición Play.
- **Play:** con esta petición comienza el envío de datos del servidor al cliente según los parámetros fijados anteriormente con Setup.
- **Pause:** se detiene el envío de datos y se puede restablecer posteriormente con Play.
- **Teardown:** detiene la entrega de datos para la URL correspondiente y libera los recursos asociados.

Otras peticiones que RTSP puede implementar son:

- **Describe:** con esta petición se obtiene una descripción del objeto multimedia asociado a la URL RTSP situada en el servidor. El servidor responde con una descripción del recurso solicitado. Esto supone la fase de inicio de RTSP.
- **Options:** esta petición es realizada por parte del cliente, y el servidor le informa del tipo de peticiones que acepta.
- **Record:** este método permite iniciar la grabación de una serie de datos multimedia según los parámetros estipulados en la petición. El servidor decide dónde guardar los datos grabados.
- **Announce:** esta petición tiene dos propósitos. Cuando se envía del cliente al servidor, muestra la descripción de un objeto multimedia. Cuando se envía del servidor al cliente, actualiza la descripción de la sesión en tiempo real.

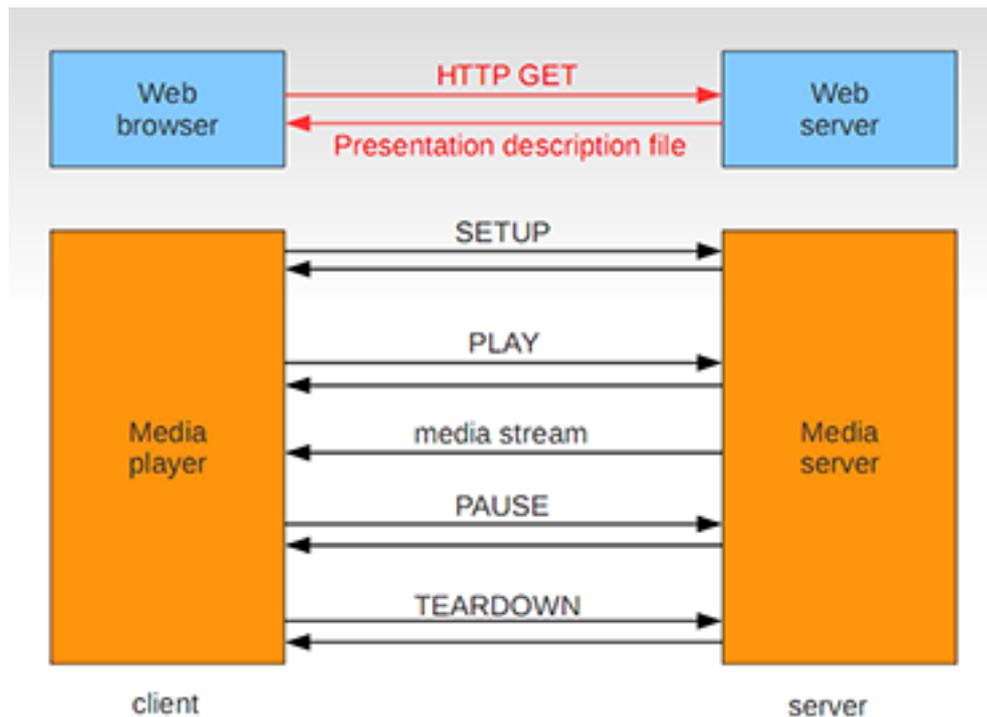


Figura 2.3: Comunicación entre el navegador y un servidor RTSP.

- **Get_Parameter:** esta solicitud recupera el valor de un parámetro de una transmisión específica. Esta petición se puede enviar vacía para probar el funcionamiento los clientes o los servidores.
- **Set_Parameter:** este método permite modificar el valor de un parámetro para una transmisión específica.
- **Redirect:** esta petición informa al cliente de que debe conectarse a otra ubicación del servidor. Contiene la cabecera obligatoria Location, que indica al cliente la nueva localización a la que debe enviar solicitudes.

La transmisión de datos a través de streaming no es por sí sola una tarea del protocolo RTSP. La mayoría de los servidores RTSP utilizan el protocolo de transporte en tiempo real (RTP) [50] junto con el protocolo de control en tiempo real (RTCP) para la entrega del contenido multimedia. Sin embargo, algunos proveedores de contenido implementan sus propios protocolos de transporte.

RTP tiene algunas características de protocolo de nivel de transporte, pero utiliza UDP para el transporte. Los paquetes RTP forman parte del flujo RTSP y se encapsulan dentro de datagramas UDP. En la siguiente imagen se puede ver la estructura de la cabecera de un paquete RTP.

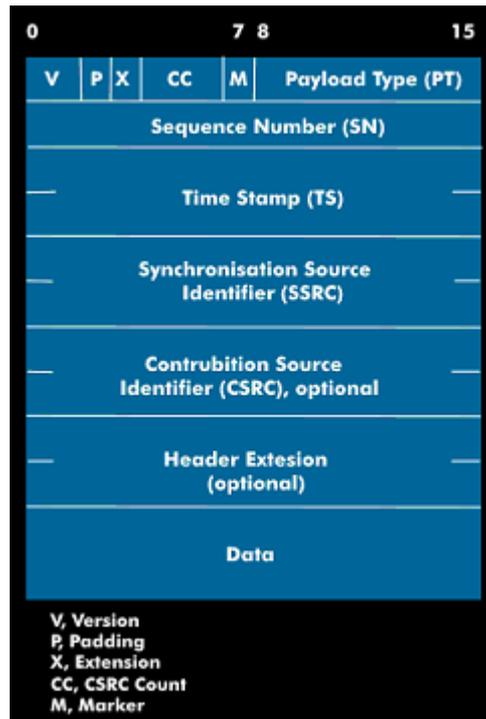


Figura 2.4: Cabecera de un paquete RTP.

Como hemos mencionado previamente, RTSP mantiene un estado que se actualiza en función de las peticiones emitidas. En la siguiente imagen podemos ver el diagrama de los posibles estados por lo que puede pasar un servidor o un cliente RTSP:



Figura 2.5: Posibles estados del servidor y el cliente RTSP.

- **Init:** es el estado inicial tanto del cliente como del servidor antes de que se realicen peticiones. También es el estado al que vuelven el cliente y el servidor después de una petición TEARDOWN.
- **Ready:** el servidor pasa a este estado al recibir una petición SETUP si su estado es INIT, o una petición PAUSE si su estado es PLAYING o RECORDING. El cliente pasa a este estado después de recibir la respuesta del servidor a la petición SETUP o PAUSE en los estados previamente mencionados.
- **Playing/Recording:** el servidor pasa a este estado después de recibir una petición PLAY o una petición RECORD. El cliente pasa a éste después de recibir la respuesta a la petición PLAY o RECORD enviada anteriormente.

Gracias a que RTSP mantiene un estado y a la comunicación del cliente y el servidor a través de las peticiones previamente detalladas, se puede realizar un control de la reproducción del contenido multimedia en tiempo real.

2.3.2. Captura y descarga de vídeo.

Hay varias formas de copiar o descargar el contenido multimedia (vídeo o música) de las páginas web. Existen complementos para navegadores, programas y páginas web que permiten la descarga de este contenido con solo la URL de la página que provee este contenido [51]. Otra alternativa para la copia de un vídeo son los programas que realizan una captura de vídeo de nuestra pantalla, lo que permitiría obtener una copia de cualquier vídeo que reproduzcamos.

En ocasiones no es necesario la ayuda de estas tecnologías para obtener un archivo multimedia. Podemos conseguir un archivo de audio o de vídeo a través de los archivos temporales de nuestro navegador. Este método no es igual de efectivo con todas las páginas web y todos los navegadores, pero por ejemplo en Windows es bastante sencillo acceder a estos archivos. En las siguientes subsecciones detallaremos el funcionamiento de los distintos métodos previamente detallados.

2.3.2.1. Complementos para navegadores.

- **Video Download profesional:** es un complemento para Mozilla Firefox y una aplicación en Google Chrome que permite descargar a nuestro ordenador un vídeo que estemos reproduciendo desde una página web. Además permite crear una lista de vídeos para poder reproducirlos sin la necesidad de volver a la página web original. En el caso de Google Chrome incluso permite ver algunos de estos vídeos en la televisión a través de Google Chromecast. El propio complemento advierte que no puede descargar vídeos protegidos que usen el protocolo RTMP.
- **Download Flash and Video:** este complemento para Firefox ofrece la posibilidad de descargar vídeos y además cualquier archivo de tipo Flash, por ejemplo juegos de Facebook o de otras páginas web con juegos basados en Flash. Se trata de un asistente para la descarga que en su página de descarga recomienda complementar con otras herramientas como DownThemAll y NoScript. Entre las múltiples páginas desde las que se puede descargar vídeo están YouTube y Facebook entre otras muchas.
- **Flash Video Downloader:** este complemento ayuda a descargar vídeo y audio desde prácticamente cualquier web. El funcionamiento de este complemento consiste en proporcionar enlaces a los archivos del contenido que estemos visualizando en la web.

En muchos casos el funcionamiento de estos complementos es parcial o cuestionable y además afecta de forma significativa al rendimiento de nuestro navegador.

2.3.2.2. Programas.

- **VLC:** este reproductor, además de ofrecer las funcionalidades que anteriormente hemos mencionado, como reproductor multimedia, plugin NPAPI o servicio de streaming, también nos permite copiar vídeo desde la web a nuestro ordenador. Desde el menú *Medio* del reproductor podemos utilizar la opción *Abrir Ubicación de Red*, que nos permite visualizar en el reproductor un vídeo de una página web a través de su dirección. Después, a través de los controles avanzados del reproductor, nos permite grabar cualquier vídeo que se esté reproduciendo en VLC.
- **RealPlayer:** si tenemos RealPlayer instalado en nuestro ordenador, al reproducir un vídeo en una página web desde el navegador (Internet Explorer, Firefox o Chrome) nos aparecerá un botón desde el que podemos descargar el vídeo. Este programa ofrece esta funcionalidad para múltiples formatos e incluso ofrece la posibilidad de descargar varios vídeos de forma simultánea.
- **Vdownloader:** este programa ofrece la posibilidad de descargar y convertir vídeos en cualquier formato a nuestro ordenador desde múltiples páginas, entre ellas YouTube, Facebook o MegaVideo. Otra herramienta similar es ClipNabber.
- **Programas de captura de pantalla:** otra forma de obtener el contenido que estamos reproduciendo desde nuestro navegador en una página web es capturar la imagen de nuestra pantalla a través de un programa. Algunos de estos programas son Hypercam o Camstudio, que permiten grabar lo que se muestra en nuestra pantalla y guardarlo en un archivo en formato AVI o WMV.

2.3.2.3. Páginas web.

- **Force-download:** desde esta página podemos realizar tres acciones: en primer lugar nos permite buscar vídeos a través de su motor de búsqueda; además nos permite la descarga de vídeos a través de su URL; y en último lugar nos permite transformar el vídeo al formato que elijamos. Otras web similares son *savefrom.net*, *bajaryoutube.com* o *VideoGetting*, entre muchas otras.

2.3.2.4. Archivos temporales.

Como ya hemos mencionado anteriormente, es posible obtener el contenido multimedia que reproducimos en nuestro navegador a través de los archivos temporales del mismo.

Cuando abrimos una página web con el navegador, se generan una serie de archivos temporales, que quedan almacenados en nuestro ordenador. En el caso de Windows estos archivos son bastante fáciles de localizar sin importar qué navegador estemos utilizando. Una vez localizado este archivo solo habría que copiarlo en otra carpeta y añadirle la extensión que nos interese.

Para poder descargar un vídeo completo es necesario que se cargue por completo en el navegador. Este método es muy sencillo y no necesita de software adicional, aunque no funciona igual con todos los navegadores y las páginas web. En algunos casos el contenido que nos interesa podría dividirse en varios archivos temporales que posteriormente se deberían fusionar con algún programa de edición de vídeo. Como ya se ha mencionado anteriormente este método no siempre es efectivo, pero es destacable que con un método tan sencillo se pueda obtener contenido multimedia a través de la web.

2.3.3. Mecanismos de cifrado de vídeo.

2.3.3.1. HTTPS

HTTPS (Hypertext Transfer Protocol Secure) es un protocolo de la capa de aplicación que permite la comunicación a través de redes de ordenadores, se utiliza especialmente en Internet [52]. Técnicamente no es un protocolo por sí mismo, es el resultado de incluir el protocolo HTTP en la parte superior del protocolo SSL o del protocolo TLS. Así se añaden las capacidades de seguridad de SSL/TLS a la comunicación estándar de HTTP. La principal motivación de HTTPS es proporcionar autenticación al sitio web y proteger la privacidad y la integridad de los datos intercambiados.

En su habitual despliegue en Internet, HTTPS proporciona autenticación al sitio web y al servidor web asociado con el que se está comunicando. Esto protege de los ataques del tipo man in the middle (MitM). Esta vulnerabilidad permite leer, insertar y modificar los mensajes entre dos partes sin que las partes lo detecten.

Además HTTPS provee encriptación bidireccional de la comunicación entre el cliente y el servidor, que protege de accesos ilícitos y de la manipulación o falsificación del contenido de la comunicación. En la práctica esto proporciona una garantía razonable de que la comunicación con el sitio web es precisa (y no se realiza con un impostor). Además asegura que de esta forma el contenido de la comunicación entre el usuario y el sitio web no puede ser leído ni modificado por terceras partes.

Históricamente las conexiones HTTPS se han utilizado principalmente para las operaciones de pago a través de la web, el correo electrónico y las transacciones sensibles en los sistemas de información corporativos. Desde hace unos años el uso de HTTPS se ha extendido para proteger la autenticidad de todo tipo de páginas web, mantener la seguridad de cuentas de usuario, proteger las comunicaciones o la identidad de los usuarios y permitir una mayor privacidad para navegar en la web.

Las URLs de HTTPS comienzan con *https://* y utilizan por defecto el puerto 443, mientras que las URLs de HTTP comienzan con *http://* y utilizan por defecto el puerto 80.

HTTPS permite crear un canal seguro a través de una red insegura, ya que permite proporcionar protección aunque solo un lado de la comunicación esté autenticado (normalmente el servidor). Debido a que HTTPS incrusta el protocolo HTTP completo en la parte superior de TLS, todo el protocolo HTTP puede ser encriptado. Esto incluye la URL solicitada, los parámetros de consulta, encabezados y las cookies. Sin embargo, los sitios web no pueden proteger la divulgación de las direcciones y los números de puerto, puesto que forman parte de los protocolos TCP/IP subyacentes.

Esto significa que en la práctica, aunque el servidor web esté correctamente configurado, un intruso puede obtener la dirección IP y el número de puerto de un servidor web e incluso el nombre de dominio. También puede obtener la duración de la sesión durante una sesión y la cantidad de datos transmitidos durante la misma, aunque no puede obtener el contenido de la comunicación. HTTPS es especialmente importante en redes inseguras, como WiFis de acceso público, ya que cualquiera conectado a esa misma red puede descubrir la información no protegida por el protocolo HTTPS.

La popularidad de HTTPS ha crecido debido a su uso en la red anónima Tor, y sobretudoo por la creciente necesidad de proteger nuestra información en la red. Los usuarios cada vez comparten más información en la red y realizan más transacciones a través de la misma. Además los ataques y los robos de información son más habituales y más sofisticados. Esto ha aumentado la necesidad de HTTPS en un mayor número de sitios web.

Firefox utiliza HTTPS para las búsquedas en Google desde la versión 14 e incluso EFF (Electronic Frontier Foundation) ha creado un complemento para Firefox llamado HTTPS Everywhere. Este complemento habilita HTTPS por defecto para cientos de webs de uso habitual, además existe una versión beta de este plugin para Google Chrome y Chorium.

La seguridad de HTTPS es la proporcionada por TLS. Se utilizan claves públicas y secretas para intercambiar la clave de una sesión para cifrar los datos entre el cliente y el servidor. Se utilizan los certificados X.509 para garantizar que la comunicación se realiza con el cliente o el servidor con el que pretendemos comunicarnos. Como consecuencia de esto se necesita el certificado y la clave pública para verificar la relación entre el titular del certificado y el certificado.

Una web debe estar completamente alojada a través de HTTPS, sin que parte de su contenido se cargue a través de HTTP o será vulnerable a los ataques. Cada vez que se accede con HTTP en lugar de con HTTPS a la información de un sitio sensible, el usuario y la sesión estarán expuestos a ataques. Además, en las web alojadas a través de HTTPS, las cookies tienen que tener el atributo seguro activado.

La protección que se obtiene con HTTPS depende en gran parte en cómo se hayan implemen-

tado los algoritmos de cifrado, el navegador web y el software del propio servidor. A pesar de la protección que ofrece HTTPS tiene ciertas limitaciones y vulnerabilidades para algunos tipos de contenido y algunas formas de ataque, como los ataques criptográficos. Pero en términos generales HTTPS ofrece una buena protección, tanto para la autenticación como para el contenido de las comunicaciones a través de la red.

2.3.3.2. TLS/SSL

SSL (Secure Sockets Layer) y su sucesor TLS (Transport Layer Security) son protocolos que proporcionan comunicación segura a través de una red, generalmente Internet, a través de la criptografía [53]. En estos protocolos se utilizan certificados X.509 y criptografía asimétrica para autenticar la otra parte de la comunicación y acordar una clave simétrica. Esta clave de sesión se utiliza para el cifrado de los datos intercambiados en la comunicación. Con esto se consigue que los datos intercambiados sean confidenciales. El uso de estos protocolos se ha generalizado en múltiples tecnologías como la mensajería instantánea, los navegadores web o el correo electrónico.

Según el modelo TCP/IP los protocolos TLS y SSL encriptan los datos de las conexiones de red en la capa de presentación. Por otra parte, en el modelo OSI, TLS/SSL se inicia en la capa de sesión y trabaja en la capa de presentación. La capa de sesión utiliza un cifrado asimétrico para establecer la configuración de cifrado y una clave compartida de sesión. Posteriormente la capa de presentación encripta el resto de la comunicación utilizando un algoritmo de cifrado simétrico y la clave de sesión anteriormente establecida. Tanto TLS como SSL realizan parte del trabajo de la capa de transporte subyacente al transportar segmentos con los datos encriptados.

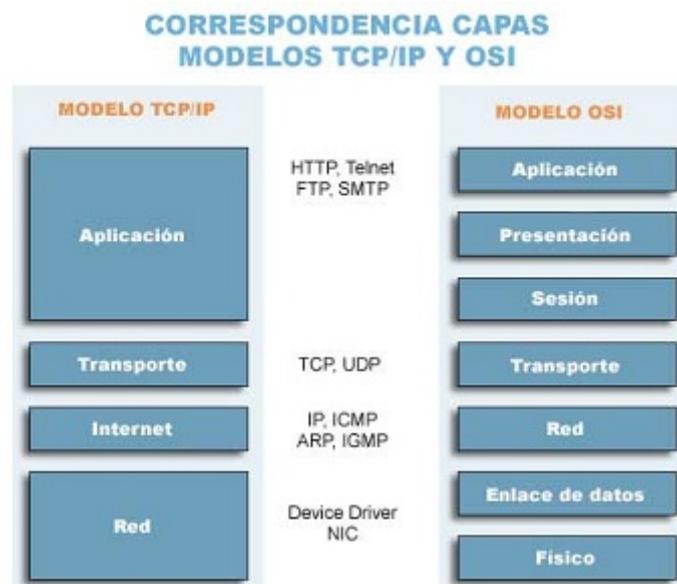


Figura 2.6: Modelos de red de comunicación.

TLS es un protocolo estándar de Internet Engineering Task Force (IETF) definido por primera vez en 1999. Se ha actualizado posteriormente en RFC 5246 y RFC 6176. Este protocolo se ha

basado en las especificaciones anteriores de SSL, que fue desarrollado por Netscape Communications a mediados de los 90 para incluir el protocolo HTTPS en su navegador web.

El protocolo TLS permite a las aplicaciones de tipo cliente-servidor comunicarse a través de una red, de forma que no se pueda ver ni manipular el contenido de la comunicación. Puesto que un protocolo de comunicación puede funcionar tanto con o sin TLS (o SSL) es necesario que el cliente indique al servidor el establecimiento de una conexión TLS. Hay dos formas de hacerlo: una opción es utilizar un puerto diferente para las conexiones TLS, como el puerto 443 para HTTPS; y la otra forma es que el cliente solicite al servidor cambiar la conexión a TLS, a través de un mecanismo específico del protocolo como STARTTLS, utilizado en protocolos de correo y noticias.

Cuando el cliente y el servidor han acordado el uso de TLS, se negocia una conexión con estado utilizando un procedimiento de toma de contacto conocido como *handshaking* o *apretón de manos*. Durante este procedimiento el cliente y el servidor acuerdan una serie de parámetros para establecer la seguridad de la conexión, siguiendo estos pasos:

- El *handshake* comienza cuando el cliente se conecta con un servidor habilitado para ofrecer TLS, el cliente solicita una conexión segura y presenta una lista de las funciones de cifrado y funciones hash que soporta.
- De esta lista el servidor escoge una función de cifrado para la que tenga soporte y se lo notifica al cliente.
- El servidor normalmente envía su identificación en forma de certificado digital. Este certificado suele contener el nombre del servidor, la autoridad certificadora de confianza y la clave pública de cifrado del servidor.
- El cliente puede contactar con el servidor que emitió el certificado y confirmar la validez del mismo antes de continuar.
- Para generar las claves de sesión utilizadas para la conexión segura, el cliente encripta un número aleatorio con la clave pública del servidor y envía el resultado al propio servidor. Solo el servidor debería poderlo descifrar con su clave privada.
- A partir del número aleatorio ambas partes generan una clave secreta simétrica denominada *secreto maestro* (*master secret*) y después negocian una clave de sesión para el cifrado y el descifrado.

Con esto termina el *handshake* y comienza la conexión segura, que encripta y desencripta el contenido hasta que la conexión se cierra. Si alguno de los pasos falla, el *handshake* se considera fallido y no se crea la conexión.

Hoy en día las últimas versiones de los grandes navegadores soportan TLS 1.0, 1.1 y 1.2 y lo tienen habilitado por defecto. Pueden surgir algunos problemas con algunos navegadores,

sobretudo si no están actualizados a la última versión. Además los protocolos SSL y TLS se han implementado en múltiples proyectos de software libre y abierto. Se pueden utilizar librerías como OpenSSL o GnuTLS para obtener la funcionalidad de SSL/TLS. Muchas aplicaciones o sistemas operativos utilizan bibliotecas similares o tienen su propia implementación de estos protocolos, pero un uso incorrecto de estas librerías puede generar vulnerabilidades.

2.3.4. Autenticación de usuarios

Con la autenticación se busca evitar el acceso ilícito a un lugar o a un sistema. En nuestro caso para este proyecto, sería de interés la autenticación para conseguir que solo los usuarios autorizados puedan acceder al contenido. Se utiliza en muchos sistemas para controlar el acceso y para realizar un seguimiento de la actividad de los usuarios.

Podemos definir la identificación de un usuario en sistema como el momento en que dicho usuario se muestra en el sistema y la autenticación puede definirse como la verificación que el sistema realiza sobre la identificación del usuario [54]. Existen cuatro formas de realizar la autenticación de la identidad del usuario, y se pueden utilizar de forma individual o combinada:

- Algo que el usuario conoce: una clave criptográfica, una clave secreta o un número de identificación personal.
- Algo que el usuario tiene: una tarjeta magnética o una llave.
- Algo que identifica al usuario de forma única: la voz o las huellas digitales.
- Algo que el usuario es capaz de realizar: un patrón de escritura.

Cada vez se utiliza más la autenticación en dos pasos que combina algo que el usuario tiene con algo que el usuario sabe. Normalmente la autenticación a través de la red se realiza a través de algo que el usuario sabe (login y password), pero en algunos casos no ofrece el nivel de seguridad deseado y por eso se utiliza de forma más frecuente la autenticación en dos pasos. En el sistema bancario lleva muchos años utilizándose este tipo de autenticación con las tarjetas (algo que el usuario tiene) y el PIN (algo que el usuario sabe). Un tipo de autenticación en dos pasos más reciente se realiza a través del teléfono móvil, por ejemplo enviando un código de verificación al móvil del usuario para que confirme su identidad con ese código.

Hay otra opción para la autenticación en dos pasos que resulta más barata al no tener que enviar mensajes al móvil. Se conoce como contraseña de uso único basada en el tiempo o TOTP (Time based One Time Password). Se trata de una contraseña que se utiliza una única vez y que cambia con el tiempo. El funcionamiento es el siguiente:

- El servidor escoge un número como base para las contraseñas que se generen posteriormente, y se transmite a la aplicación del teléfono donde se guarda.

- Cuando se quiere acceder al servicio, el servidor solicita un segundo código de autenticación.
- Para obtener este código hay que acceder a la aplicación con el móvil, y se genera un código en función de la clave, la fecha y la hora.

De esta forma, solo el propietario del móvil, y por lo tanto un usuario autorizado, puede acceder al sistema o al contenido que se busca proteger.

2.4. Conclusiones y valoraciones

Después de este detallado estudio sobre los navegadores, el funcionamiento de los plugins en los mismos, de los Frameworks para el desarrollo de plugins, de las tecnologías utilizadas tanto en la reproducción, como de la descarga y el cifrado del vídeo, y por último de los métodos de cifrado de vídeo y de autenticación de usuarios, se ha llegado a la conclusión de que la mejor alternativa para este proyecto es **la implementación de un cliente y un servidor RTSP, que permita realizar un control de los procesos en ejecución del lado del cliente.**

Hemos escogido esta alternativa porque nos permite realizar, a partir del cliente, un control de los procesos en ejecución en la máquina del usuario durante la reproducción del contenido. De esta forma, podemos pausar la reproducción del contenido ante la ejecución de procesos de captura de pantalla.

Las otras alternativas no nos permitían realizar este control de los procesos. Además de este estudio hemos realizado algunas pruebas con las tecnologías estudiadas y se han descartado por los siguiente motivos:

- Se desarrolló un pequeño plugin y se realizaron pruebas con varios plugins de vídeo [55], pero los plugins solo proporcionan control sobre la pestaña del navegador en la que se está utilizando.
- También se probó a listar los procesos en ejecución en el lado del cliente, pero con los lenguajes de desarrollo web solo puede hacerse del lado del servidor. Se consiguió listar los procesos del lado del servidor con varios lenguajes como PHP o Python, pero esto no tenía utilidad.
- Aunque utilizando un plugin o HTML5 junto con los mecanismos de cifrado de vídeo y de autenticación de usuario se podría evitar la descarga del contenido a partir de complementos del navegador o páginas web, no podríamos evitar la copia del contenido a través de la captura de pantalla.

Para poder realizar un control de los proceso en ejecución es necesario tener nuestro software instalado en la máquina del usuario. Para poder justificar que el usuario tenga que instalar software en su equipo, el cliente RTSP será el encargado de recibir y reproducir el contenido, además

de controlar la ejecución de procesos. Si el contenido se reproduce en el navegador es difícil de justificar que el usuario tenga que instalar en su equipo una aplicación para que realice el control de procesos.

Con este control de procesos podemos evitar la captura de vídeo por parte de la mayoría de los usuarios. Además los métodos de descarga a través de complementos o páginas web no tienen ningún efecto sobre una aplicación Cliente-Servidor RTSP que no se relaciona con el navegador.

Por lo tanto se trata de implementar un cliente RTSP que permita recibir y reproducir el contenido multimedia en el equipo del usuario, y que realice un control de los procesos en ejecución y pause la reproducción de contenido ante la ejecución. Y por otro lado, implementar un servidor RTSP que permita enviar el contenido multimedia al cliente RTSP desde otro equipo, porque nuestro objetivo es que el usuario pueda ver el contenido sin que éste esté en su equipo.

Capítulo 3

Análisis

3.1. Alcance del proyecto.

Podemos describir este trabajo como el estudio de las tecnologías de reproducción, difusión y copia de vídeo a través de la red y el desarrollo de una aplicación que se encarga de retransmitir el contenido desde el servidor hasta el cliente, en el que se reproduce el propio contenido. Además durante la reproducción del contenido el sistema implementado se encarga de listar los procesos en ejecución en el equipo del cliente, y si alguno de estos procesos es sospechoso de realizar una captura o grabación de la pantalla se pausa la reproducción del contenido.

Con el desarrollo de este sistema hemos buscado conseguir una forma segura de transmitir vídeo a través de la red. Al utilizar un sistema Cliente-Servidor que se comunica a través de los protocolos RTP y RTSP evitamos la descarga del contenido que puede realizarse con múltiples tecnologías (páginas web, complementos del navegador o aplicaciones), como cuando se accede al contenido a través de una página web con el navegador.

Para que nuestro sistema sea más seguro, además de la descarga del vídeo debemos evitar su copia (o captura), y por eso durante la ejecución del proceso se comprueban cada cinco segundos los procesos en ejecución. Esto evita o ,en gran medida, dificulta la copia del vídeo, al menos con software incluido en este proyecto como sospechoso de realizar captura o grabación de la pantalla.

A pesar de todo sería posible capturar el vídeo a través de determinados tipos de hardware o con software no incluido en este proyecto. La parte del hardware sería un estudio interesante con el que ampliar este trabajo. En la parte del software sería sencillo incluir nuevo software como sospechoso sin que el usuario tenga que realizar cambios o actualizaciones.

Aunque se pueda realizar la captura del contenido a través de hardware o, conociendo el funcionamiento del sistema, con software, con esta aplicación conseguimos dificultar la copia del vídeo y en muchos casos evitarla, ya que la mayor parte de los usuarios no dispone ni de las herramientas, ni de los conocimientos necesarios para copiar el vídeo. Trataremos más este tema

en el capítulo de pruebas y en el de trabajos futuros.

3.2. Requisitos.

3.2.1. Requisitos funcionales.

- [RF.1] **Reproducción de vídeo:** el reproductor debe permitir la reproducción del vídeo.
- [RF.2] **Pausado de vídeo:** el reproductor debe permitir pausar la reproducción del vídeo.
- [RF.3] **Opacidad de los protocolos:** el sistema debe permitir al usuario interactuar con la aplicación sin percibir los protocolos subyacentes .
- [RF.4] **Control de procesos:** el vigilante debe controlar los procesos en ejecución.
- [RF.5] **Pausa ante procesos sospechosos :** el cliente debe pausar la reproducción del vídeo ante la ejecución de procesos sospechosos.
- [RF.6] **Envío del contenido:** el servidor debe proporcionar el contenido multimedia al cliente.
- [RF.7] **Listado de procesos sospechosos:** el servidor debe proporcionar la lista de procesos sospechosos.
- [RF.8] **Salida del sistema:** el sistema debe permitir al usuario abandonar el sistema cuando lo desee.
- [RF.9] **Interacción Cliente-Servidor:** el sistema debe permitir la interacción con un único cliente de forma simultánea.

3.2.2. Requisitos no funcionales

- [RNF.1] **Tiempo de reproducción:** el tiempo de respuesta para reproducir un vídeo menor a 5 segundos.
- [RNF.2] **Interfaz sencilla:** el sistema debe tener una interfaz fácil de utilizar e intuitiva.
- [RNF.3] **Multiplataforma:** la aplicación debe ser multiplataforma (al menos Windows y Linux).
- [RNF.4] **Formatos de vídeo:** el formato de vídeo soportado será MJPEG.
- [RNF.5] **Lenguajes de implementación:** la aplicación será implementada utilizando Java.
- [RNF.6] **Documentación:** la documentación del proyecto será desarrollada con LaTeX.

- [RNF.7] **Protocolo de comunicación:** la comunicación entre el cliente y servidor se realizará a través del protocolo RTSP.
- [RNF.8] **Protocolo de transporte:** el contenido se provee a través del protocolo RTP.
- [RNF.9] **Manual de usuario:** se proporcionará un manual de Usuario e instalación de la aplicación.
- [RNF.10] **Implementación de los protocolos:** la implementación de los protocolos se basará en varias clases incompletas proporcionadas por UMBC [56]. Este requisito influye en los dos siguientes.
- [RNF.11] **Archivo único:** se reproduce un único archivo de vídeo.
- [RNF.12] **Archivo sin audio:** el archivo de vídeo no contiene audio y se transmite como una serie de imágenes.

3.3. Modelo de dominio.

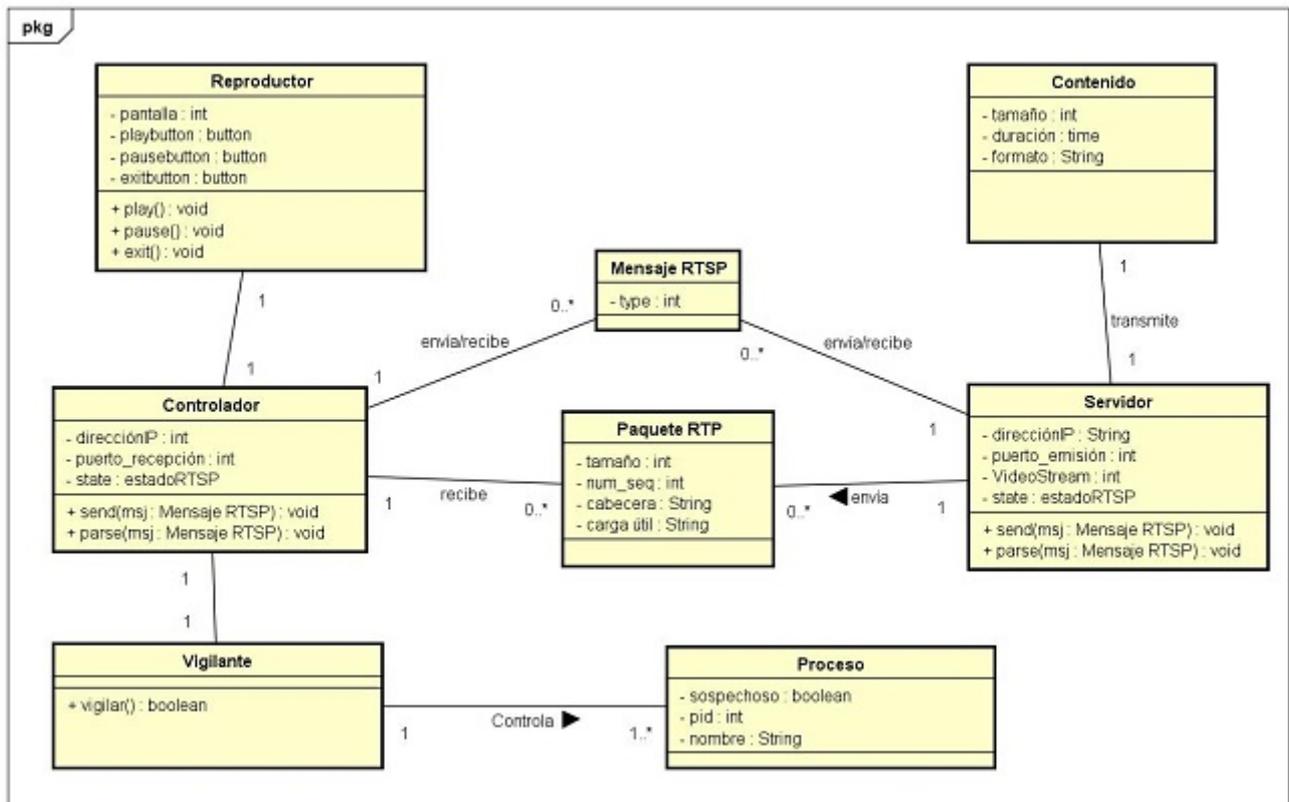


Figura 3.1: Modelo de dominio.

Es difícil realizar un modelo de dominio de nuestro sistema, puesto que se trata de un modelo conceptual que representa de mejor forma entidades con datos que la interacción a través de pro-

tocolos subyacentes que caracteriza nuestra aplicación.

El modelo de dominio muestra una idea conceptual de nuestra aplicación y no se corresponde con la arquitectura que posteriormente tiene el software, aunque tiene similitudes. Aunque en el siguiente capítulo mostraremos en detalle la arquitectura de la aplicación, hemos querido mostrar, aunque sea de una forma más abstracta, los principales conceptos de la aplicación.

3.4. Diagrama de casos de uso.

En este diagrama mostramos los principales casos de uso a realizar por el usuario.

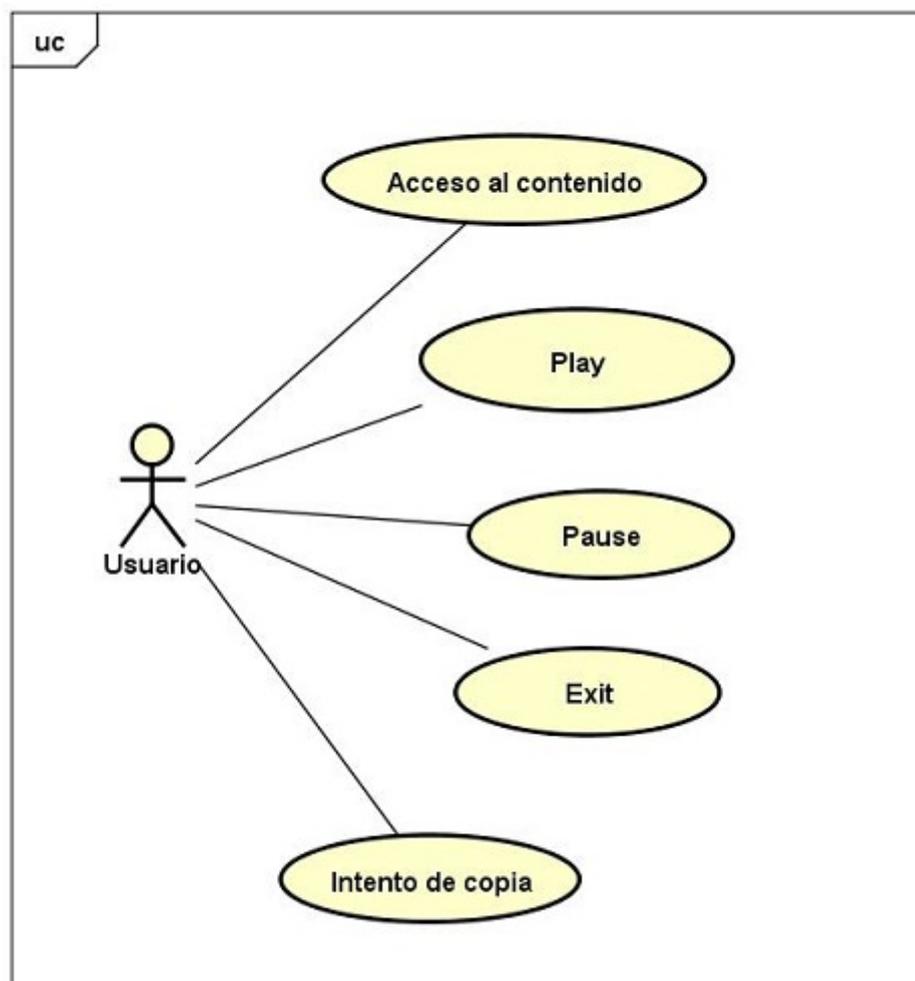


Figura 3.2: Diagrama de casos de uso.

3.5. Casos de uso.

En esta sección detallamos los casos de uso nombrados en el apartado previo, mostrando los pasos durante su realización. Hay que destacar que en nuestro sistema la interacción con el usuario es reducida. Todos los casos de uso se componen de una única acción del usuario y de la correspondiente reacción del sistema. Esto se debe a que el sistema no pide al usuario que introduzca datos o que escoja entre distintas opciones.

Hemos simplificado al máximo la interfaz de nuestro sistema para que el usuario únicamente tenga que acceder al contenido, reproducirlo, pausarlo o salir del sistema. Un caso de uso especial es el intento de copia: aunque también se compone de una acción del usuario y la respuesta del sistema, no se trata de un operación habitual del usuario. Sin embargo, que nuestro sistema reaccione de forma oportuna ante un intento de copia es uno de los objetivos principales de este proyecto.

En esta sección solo mostraremos la reacción del sistema ante las acciones del usuario. Las operaciones que realiza el sistema para responder a las acciones del usuario serán detalladas en los diagramas de secuencia del siguiente capítulo. En estos diagramas mostraremos la interacción de las clases en lo que en este apartado mostramos como un único paso.

Para la realización de los otros cuatro casos de uso es necesario la realización del caso de uso 1, por lo que las precondiciones necesarias para este caso de uso también lo son para los demás. Pero para simplificar las tablas de los casos de uso, solo se incluyen estas precondiciones en el primer caso de uso.

Caso de uso 1	Acceso al contenido
Descripción	Se realiza la conexión con el servidor y se da acceso al contenido para el usuario.
Precondición	El reproductor se ha instalado satisfactoriamente en el equipo del usuario y la aplicación se ha iniciado.
Secuencia normal	
Paso 1	El usuario selecciona acceder al contenido.
Paso 2	El sistema inicia la conexión RTSP con el servidor, recibe la lista de procesos sospechosos y muestra la pantalla de reproducción del contenido.
Postcondición	El sistema muestra la pantalla de reproducción del contenido.
Excepciones	2' Si el servidor no está conectado, está conectado con otro cliente o la dirección del servidor no es correcta, se muestra un mensaje de error y se cierra el sistema.
Frecuencia	Alta
Importancia	Alta
Estabilidad	Alta

Tabla 3.1: Caso de uso 1

Caso de uso 2	Play
Descripción	Se inicia la reproducción del contenido.
Precondición	Se ha accedido de forma satisfactoria al contenido.
Secuencia normal	
Paso 1	El usuario selecciona la reproducción del contenido.
Paso 2	El sistema comienza la recepción del contenido desde el servidor y lo muestra.
Postcondición	El sistema realiza la reproducción del contenido.
Excepciones	2' Si el servidor se cierra de forma inesperada, ante la operación del usuario se muestra una mensaje de error y se cierra el sistema.
Frecuencia	Alta
Importancia	Alta
Estabilidad	Alta

Tabla 3.2: Caso de uso 2

Caso de uso 3	Pause
Descripción	Se pausa la reproducción del contenido.
Precondición	El sistema está reproduciendo el contenido.
Secuencia normal	
Paso 1	El usuario selecciona pausar la reproducción.
Paso 2	El sistema detiene la transmisión del contenido desde el servidor y el reproductor se detiene en la última imagen recibida.
Postcondición	El sistema pausa la reproducción del contenido.
Excepciones	2' Si el servidor se cierra de forma inesperada, ante la operación del usuario se muestra una mensaje de error y se cierra el sistema.
Frecuencia	Alta
Importancia	Alta
Estabilidad	Alta

Tabla 3.3: Caso de uso 3

Caso de uso 4	Exit
Descripción	Se realiza un cierre ordenado del reproductor.
Precondición	El sistema ha accedido al contenido. Se pueden dar tres situaciones: el contenido se está reproduciendo, la reproducción está pausada o aún no se ha empezado a reproducir el contenido.
Secuencia normal	
Paso 1	El usuario selecciona cerrar el reproductor.
Paso 2	El sistema cierra la conexión con el servidor y posteriormente se cierra el propio sistema.
Postcondición	El sistema se cierra sin mostrar mensajes de error.
Excepciones	2' Si el servidor se cierra de forma inesperada, ante la operación del usuario se muestra una mensaje de error y se cierra el sistema.
Frecuencia	Alta
Importancia	Alta
Estabilidad	Alta

Tabla 3.4: Caso de uso 4

Caso de uso 5	Intento de copia
Descripción	Una vez iniciada la reproducción del contenido, el usuario abre un programa incluido en la lista de procesos sospechosos.
Precondición	El sistema ha iniciado la reproducción del contenido, aunque se encuentre pausado. Se ha recibido la lista de procesos desde el servidor
Secuencia normal	
Paso 1	El usuario abre un proceso sospechoso de realizar una copia o grabación del contenido.
Paso 2	El sistema detiene la transmisión del contenido desde el servidor y no permite que se continúe con dicha transmisión hasta que se cierre el proceso o procesos sospechosos.
Postcondición	El sistema pausa la reproducción del contenido ante la aparición del proceso sospechoso.
Excepciones	2' Si la transmisión del contenido ya está pausada o no se ha comenzado, no se permite la continuación o el inicio de la transmisión hasta que no se cierre el proceso o procesos sospechosos.
Frecuencia	Alta
Importancia	Alta
Estabilidad	Alta

Tabla 3.5: Caso de uso 5

Capítulo 4

Gestión del proyecto

4.1. Estudio de los riesgos.

A continuación se muestra una identificación de los diferentes riesgos (eventos que si ocurren tienen un efecto positivo o negativo sobre los objetivos del proyecto) que podemos encontrar, además de su análisis, planificación y monitorización. La gestión de riesgos nos va servir como una inversión de futuro, ahorrándonos costes de corrección de problemas que ya se han producido y mejorando el control del proyecto.

Existen principalmente tres causas de riesgos en el ámbito de proyectos software:

- **Riesgos de Proyecto:** restricciones de recursos, interfaces externas, relaciones con los proveedores, etc.
- **Riesgos de Proceso:** proceso software no documentado, proceso de diseño pobre, planificación ineficaz, etc.
- **Riesgos de Producto:** diseño complejo, requisitos incompletos, etc.

Sobre cada uno de los riesgos identificados detallaremos la siguiente información:

- **Tipo de riesgo:** si se trata de un riesgo de Producto, Proceso o Proyecto.
- **Enunciado:** breve explicación del riesgo a analizar.
- **Probabilidad:** la posibilidad de ocurrencia del riesgo expresada en %.
- **Magnitud:** se da una estimación de la importancia que tiene la ocurrencia del riesgo. La escala utilizada es la siguiente (de mayor a menor importancia): catastrófica, crítica, media, marginal y despreciable.
- **Consecuencias:** descripción de los efectos que tiene sobre el proyecto la ocurrencia del riesgo.

- **Indicadores:** son hechos cuya ocurrencia aumenta la posibilidad de aparición del riesgo.
- **Plan de acción:** medidas planificadas a lo largo del proyecto para intentar evitar los riesgos o minimizar sus consecuencias.
- **Plan de contingencia:** medidas planificadas para cuando el riesgo ha ocurrido intentar minimizar los daños.
- **Riesgo 1:** No se cumple con los plazos, se retrasa el proyecto.
 - **Tipo de riesgo:** Riesgo de Proyecto.
 - **Enunciado:** se alarga el desarrollo del proyecto, incumpléndose los plazos previstos en la planificación.
 - **Probabilidad:** 50 %.
 - **Magnitud:** Crítica.
 - **Consecuencias:** el retraso del proyecto impediría presentar el Trabajo de Fin de Grado en la convocatoria ordinaria. Esto demoraría la obtención del título.
 - **Indicadores:** no se está cumpliendo la planificación; no se consigue una versión estable del software.
 - **Plan de acción:** se intentará seguir la planificación de la forma más rigurosa posible.
 - **Plan de contingencia:** si se incumple la planificación o se tienen dificultades, se modificará la planificación aumentando la carga de trabajo. En última instancia, retrasar la presentación del Trabajo de Fin de Grado.
- **Riesgo 2:** No se cumplen los objetivos del proyecto.
 - **Tipo de riesgo:** Riesgo de Producto.
 - **Enunciado:** el proyecto pretende conseguir una serie de objetivos, pero dada la magnitud del problema planteado, no sabemos si todos los objetivos serán alcanzables.
 - **Probabilidad:** 40 %.
 - **Magnitud:** Catastrófica.
 - **Consecuencias:** se tendrían que modificar los objetivos del proyecto.
 - **Indicadores:** después de varias versiones del software no conseguimos alguna de las funciones que teníamos previstas.
 - **Plan de acción:** intentar que la aplicación obtenida se adapte de la mejor forma posible a los objetivos iniciales.
 - **Plan de contingencia:** modificar los objetivos iniciales que no sean alcanzables.
- **Riesgo 3:** Las pruebas realizadas son pocas o insuficientes.

- **Tipo de riesgo:** Riesgo de Proceso.
 - **Enunciado:** la batería de pruebas no tiene en cuenta alguno de los posibles fallos o de los casos críticos.
 - **Probabilidad:** 30 %.
 - **Magnitud:** Marginal.
 - **Consecuencias:** el software puede realizar funciones inesperadas o incorrectas en los casos no contemplados en las pruebas.
 - **Indicadores:** Resultados inesperados en el funcionamiento del programa.
 - **Plan de acción:** hacer un plan de pruebas detallado utilizando técnicas formales, poniendo especial atención a los casos más complejos y problemáticos.
 - **Plan de contingencia:** rehacer el plan de pruebas teniendo en cuenta los casos no contemplados
- **Riesgo 4:** La planificación temporal es inadecuada.
- **Tipo de riesgo:** Riesgo de Proceso.
 - **Enunciado:** la planificación no es adecuada para las necesidades del proyecto.
 - **Probabilidad:** 60 %.
 - **Magnitud:** Media.
 - **Consecuencias:** reorganización y modificación de la planificación, se podría retrasar el proyecto.
 - **Indicadores:** las tareas se prolongan más de lo previsto y no se cumplen los plazos establecidos.
 - **Plan de acción:** realizar un seguimiento de la planificación para ver si se adecúa al avance del proyecto.
 - **Plan de contingencia:** rehacer la planificación adaptándola al progreso real del proyecto. Si es necesario se aumentará la carga de trabajo.
- **Riesgo 5:** Falta de conocimientos para el desarrollo.
- **Tipo de riesgo:** Riesgo de Producto.
 - **Enunciado:** no se ha trabajado previamente en ningún ámbito similar al de este proyecto, por lo que se puede desconocer alguna de las tecnologías para el desarrollo del trabajo.
 - **Probabilidad:** 40 %.
 - **Magnitud:** Media.
 - **Consecuencias:** Se requiere un mayor tiempo de aprendizaje, lo que implica modificar la planificación o un retraso en el proyecto.

- **Indicadores:** en la etapa de implementación se dedica demasiado tiempo a la consulta de documentación.
- **Plan de acción:** realizar un fase de documentación previa a la implementación.
- **Plan de contingencia:** posponer la implementación para realizar una fase de documentación más larga.
- **Riesgo 6:** Pérdida de información del proyecto.
 - **Tipo de riesgo:** Riesgo de Proceso.
 - **Enunciado:** pérdida de alguna parte del proyecto, ya sea código o documentación.
 - **Probabilidad:** 10 %.
 - **Magnitud:** Crítica.
 - **Consecuencias:** volver a crear o buscar la parte correspondiente, generando un retraso en el proyecto.
 - **Indicadores:** no se localiza alguno de los elementos del proyecto.
 - **Plan de acción:** almacenar varias copias de todos los elementos del proyecto utilizando varios medios de almacenamiento.
 - **Plan de contingencia:** volver a desarrollar los elementos que se hayan perdido.
- **Riesgo 7:** Cantidad de información excesiva, dificulta escoger la información adecuada.
 - **Tipo de riesgo:** Riesgo de Proceso.
 - **Enunciado:** en este TFG la fase de documentación y la búsqueda de la misma requiere muchas horas de esfuerzo, y además abarca un gran número de tecnologías distintas. Por lo que se tiene demasiada información, mucha de ella, inadecuada.
 - **Probabilidad:** 80 %.
 - **Magnitud:** Marginal.
 - **Consecuencias:** obtener documentación concreta sobre la temática del proyecto conlleva un tiempo excesivo.
 - **Indicadores:** una parte de considerable de la documentación encontrada no es adecuada para el ámbito del proyecto.
 - **Plan de acción:** durante la búsqueda de información, dedicar parte del tiempo a seleccionar y clasificar la documentación en función del tema y la fuente de información.
 - **Plan de contingencia:** dedicar más tiempo a la fase de documentación para descartar la información inadecuada.
- **Riesgo 8:** Problemas con el hardware.
 - **Tipo de riesgo:** Riesgo de Proceso.

- **Enunciado:** el trabajo se está realizando con un único ordenador. Algún defecto de hardware o de software podría dejarnos varios días sin nuestro ordenador de trabajo.
 - **Probabilidad:** 10 %.
 - **Magnitud:** Crítica.
 - **Consecuencias:** pérdida de tiempo al no poder utilizar el ordenador habitual o tener que configurar otro distinto.
 - **Indicadores:** el funcionamiento del ordenador no es el adecuado.
 - **Plan de acción:** realizar el mantenimiento adecuado para el ordenador de trabajo.
 - **Plan de contingencia:** instalar todos las tecnologías con las que estamos trabajando en otro ordenador.
- **Riesgo 9:** Problemas al desplegar el software en el equipo del usuario.
- **Tipo de riesgo:** Riesgo de Producto.
 - **Enunciado:** una vez desarrollada nuestra aplicación, no funciona correctamente o no es compatible con el equipo del usuario .
 - **Probabilidad:** 25 %.
 - **Magnitud:** Media.
 - **Consecuencias:** el funcionamiento del software puede ser inadecuado e incluso no funcionar.
 - **Indicadores:** no se permite instalar la aplicación o hay dificultades para la instalación.
 - **Plan de acción:** intentar crear una aplicación multiplataforma, para que sea compatible con la mayor cantidad de equipos posible.
 - **Plan de contingencia:** si no se puede crear una herramienta multiplataforma, entonces especificar claramente en el manual de instalación las características necesarias para instalar el software.
- **Riesgo 10:** Cambio de tecnología.
- **Tipo de riesgo:** Riesgo de Proceso.
 - **Enunciado:** Durante el desarrollo del proyecto, alguna de las tecnologías escogidas para desarrollar el proyecto o para su uso durante el mismo no cumple con el rendimiento o las funcionalidades previstas.
 - **Probabilidad:** 20 %.
 - **Magnitud:** Crítica.
 - **Consecuencias:** habría que escoger una tecnología que sustituya a la que inicialmente teníamos prevista, retrasando el proyecto.

- **Indicadores:** alguna de las tecnologías utilizadas no cumple con el papel que teníamos previsto.
- **Plan de acción:** realizar un estudio de las tecnologías disponibles para seleccionar las más adecuadas para la realización del proyecto.
- **Plan de contingencia:** seleccionar una tecnología alternativa para que realice la funciones previstas para la tecnología que necesitamos cambiar.

4.2. Planificación temporal.

Con esta sección se ha buscado proporcionar una guía sobre el desarrollo de este Trabajo de Fin de Grado. Como se puede ver en esta planificación temporal, se ha dedicado mucho tiempo a la búsqueda de información y a la documentación. El gran número de horas, tanto para el trabajo global como para la documentación, se debe a la singularidad del proyecto.

El desarrollo de un software para la visualización de vídeo como el que se ha desarrollado en este proyecto abarca muchos ámbitos (navegadores, desarrollo de plugins, tecnologías de copia y reproducción de vídeo, cifrado y autenticación). Esto conlleva una gran cantidad de documentación relacionada con el proyecto, sobretodo al comienzo del mismo. Además para sacar adelante un proyecto concreto era necesario realizar un estudio y escoger entre las distintas tecnologías disponibles.

Además de la gran cantidad de documentación y de la necesidad de realizar un estudio para conseguir un proyecto concreto, otro factor que ha influido en la planificación temporal es el tipo de documentación relacionada con los plugins para navegadores. Este tipo de plugins, y más en concreto los de reproducción de vídeo, han sido desarrollados solo por algunas compañías o proyectos, por lo que es difícil encontrar documentación. En muchas ocasiones, la búsqueda de esta información nos ha conducido a documentación sobre otros complementos para navegadores o plugins para Wordpress. Mucha de la documentación encontrada ha sido antigua, obsoleta o poco concreta. Por lo que la búsqueda, lectura y selección de documentación ha sido larga y tediosa.

Por otra parte, el estudio realizado en el Estado del Arte para la selección de las tecnologías utilizadas en este proyecto ha supuesto un tiempo adicional para documentarse acerca de tecnologías que, salvo en este estudio previamente mencionado, no se utilizan en el proyecto. Además hemos incluido en esta fase de documentación alguna pequeña implementación de prueba para ver las posibilidades que nos ofrecían algunas de las tecnologías estudiadas.

Por último, hay que destacar que para el desarrollo del software se ha utilizado un tipo de desarrollo en cascada[57]. Consiguiendo en primer lugar la implementación del protocolo con una arquitectura cliente-servidor y añadiéndole posteriormente más funcionalidades.

No	Nombre de la tarea	Duración	Fecha inicio	Fecha fin	Dependencias
1	Inicio				
2	Fase de análisis		19/01/2015	01/02/2015	
3	Ámbito y límites del proyecto	5	19/01/2015	25/01/2015	
4	Planificación temporal	10	19/01/2015	25/01/2015	
5	Especificación de requisitos	15	19/01/2015	25/01/2015	
6	Estudio de los riesgos	15	26/01/2015	01/02/2015	
7	Implementación de un plugin inicial	10	26/01/2015	01/02/2015	
8	Pruebas del plugin inicial	5	26/01/2015	01/02/2015	
9	Fase de documentación		02/02/2015	14/06/2015	
10	Búsqueda de información	30	02/02/2015	08/02/2015	
11	Documentación navegadores	50	09/02/2015	22/02/2015	
12	Documentación tecnologías	50	23/02/2015	08/03/2015	
13	Documentación desarrollo plugins	30	09/03/2015	15/03/2015	
14	Documentación vídeo	50	16/03/2015	29/03/2015	
15	Documentación cifrado	30	30/03/2015	05/04/2015	
16	Documentación autenticación	30	06/04/2015	12/04/2015	
17	Revisión y corrección de la documentación	50	13/04/2015	26/04/2015	
18	Revisión código FBVLC	50	27/04/2015	06/05/2015	
19	Implementación de un plugin de vídeo	80	07/05/2015	20/05/2015	
20	Control de procesos desde la web	50	21/05/2015	31/05/2015	
21	Revisión protocolos RTSP y RTP	80	01/06/2015	14/06/2015	
22	Fase de diseño		15/06/2015	02/07/2015	
23	Modelo de desarrollo	20	15/06/2015	19/06/2015	9
24	Arquitectura	20	20/06/2015	25/06/2015	9
25	Interfaz de usuario	30	26/06/2015	02/07/2015	
26	Fase de implementación		03/07/2015	02/08/2015	
27	Implementación cliente y servidor RTSP	40	03/07/2015	10/07/2015	21
28	Implementación control de procesos	40	10/07/2015	18/07/2015	20
29	Implementación reproductor	30	20/07/2015	26/07/2015	
30	Implementación interfaz de usuario	20	27/07/2015	02/08/2015	25
31	Implementación web descarga	10	27/07/2015	02/08/2015	
32	Fase de pruebas		03/08/2015	09/08/2015	
33	Pruebas protocolos	5	03/08/2015	09/08/2015	27
34	Pruebas control de procesos	5	03/08/2015	09/08/2015	28
35	Pruebas reproducción de vídeo	10	03/08/2015	09/08/2015	29
36	Otras tareas		19/01/2015	23/08/2015	
37	Memoria TFG	80	19/01/2015	23/08/2015	
38	Cambios y correcciones	30	19/01/2015	23/08/2015	
39	Manual de usuario	30	10/08/2015	16/08/2015	
40	Fin				
	Total	1000	19/01/2015	23/08/2015	

Tabla 4.1: Planificación temporal final

4.3. Plan de fases.

Como podemos ver en la imagen inferior, este Trabajo de Fin de Grado se divide en seis fases. Podemos considerarlo como cinco fases principales y una serie de tareas realizadas a lo largo del proyecto, que son difíciles de situar en una planificación temporal, porque se realizan a lo largo de todo el proyecto o porque es complicado relacionarlas con las tareas de las otras fases.

Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
★	<Inicio>	1 día	lun 19/01/15	lun 19/01/15	
➡	> <Fase de análisis>	10 días	lun 19/01/15	dom 01/02/15	
➡	> <Fase de documentación>	95 días	lun 02/02/15	dom 14/06/15	
➡	> <Fase de diseño>	14 días	lun 15/06/15	jue 02/07/15	
➡	> <Fase de Implementación>	21 días	vie 03/07/15	dom 02/08/15	
➡	> <Fase de pruebas>	5 días	lun 03/08/15	dom 09/08/15	
➡	> Otras tareas	155 días?	lun 19/01/15	dom 23/08/15	
★	<Fin>	1 día	dom 23/08/15	dom 23/08/15	

Figura 4.1: Fases planificación temporal.

Dentro de las fases de este proyecto cabe destacar la fase de documentación, porque tanto en duración temporal como en horas de esfuerzo supone más de la mitad de este Trabajo de Fin de Grado. Esto se debe a la singularidad de este proyecto, lo que ha supuesto en la práctica una dedicación mayor al estudio y la selección de tecnologías, que a la implementación de la tecnología escogida.

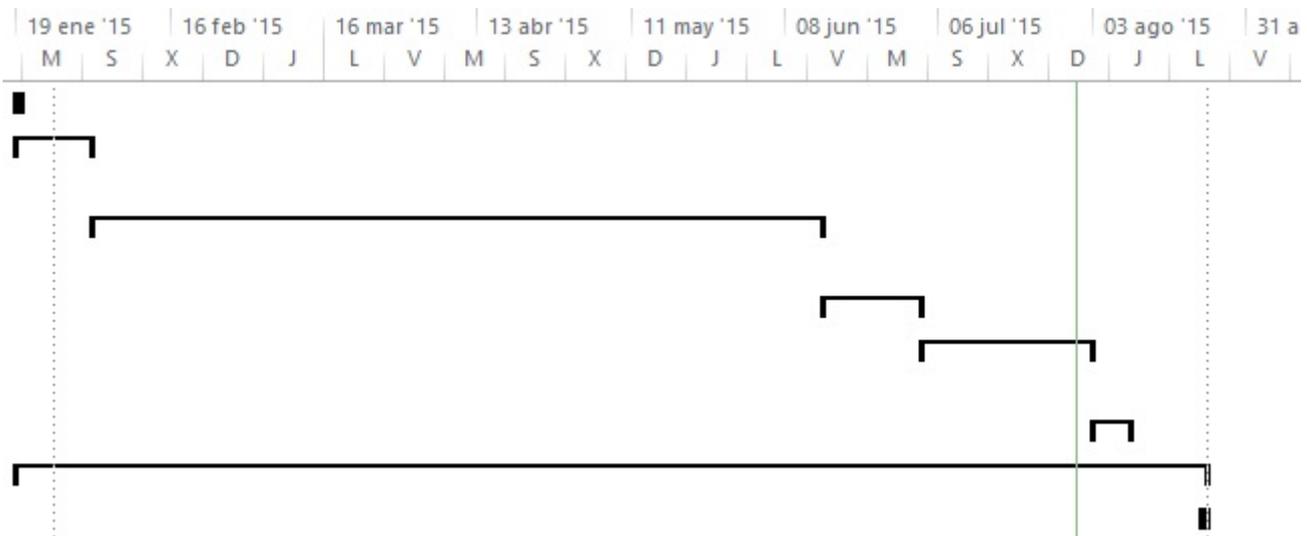


Figura 4.2: Diagrama fases planificación temporal.

4.3.1. Fase de análisis

En esta fase realizamos un planteamiento inicial del TFG. Cabe destacar que tanto la planificación como los requisitos del proyecto se han ido modificando durante el avance del trabajo. Esto se debe a que durante el proyecto se han incluido tecnologías que inicialmente no estaban previstas y esto ha alargado la duración del proyecto. La planificación que se muestra en la sección anterior es la planificación final que incluye cambios con respecto a versiones anteriores. En el estudio de los riesgos planificamos medidas ante estos eventos, como se puede ver en los Riesgos 1, 4, 5, 7 y 10, en los que se preveía un aumento de la duración del proyecto por su singularidad. Esto nos ha permitido seguir adelante con el trabajo a pesar del aumento de la duración del mismo.

<Fase de análisis>	10 días	lun 19/01/15	dom 01/02/15
Ámbito y límites del proyecto	6 días	lun 19/01/15	dom 25/01/15
Planificación temporal	6 días	lun 19/01/15	dom 25/01/15
Especificación de requisitos	6 días	lun 19/01/15	dom 25/01/15
Estudio de los riesgos	6 días	lun 26/01/15	dom 01/02/15
Implementación de un plugin inicial	6 días	lun 26/01/15	dom 01/02/15
Pruebas del plugin inicial	6 días	lun 26/01/15	dom 01/02/15

Figura 4.3: Fase de análisis.

Cabe destacar en esta fase de análisis la implementación de un plugin inicial, para conocer mejor una de las tecnologías más utilizadas hasta hace poco para la visualización de contenido multimedia a través de la red.

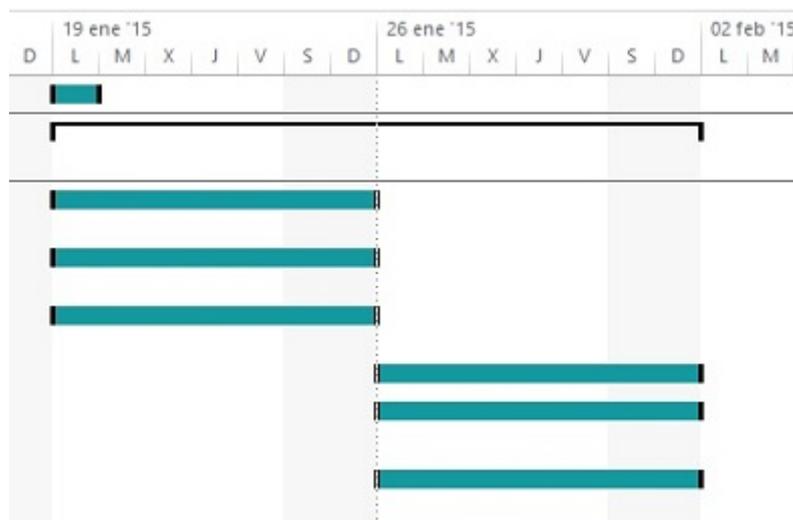


Figura 4.4: Diagrama fase de análisis.

4.3.2. Fase de documentación

Esta fase ha supuesto una gran cantidad de tiempo y de esfuerzo dentro del desarrollo de este Trabajo de Fin de Grado. Esto se debe a que en esta fase hemos realizado un estudio acerca de las tecnologías relacionadas con este proyecto para poder elegir y posteriormente implementar la tecnología que mejor se adaptase a nuestros objetivos.

Durante el estudio de algunas de las tecnologías hemos descubierto otras relacionadas con ellas o de interés para el proyecto, por lo que se han incluido en el estado del arte. Esto ha supuesto múltiples modificaciones de la planificación y el aumento de la duración de esta fase de documentación. Hemos preferido realizar por separado el estudio de cada una de las tecnologías y dedicarle un tiempo específico a cada una, para así poder tener una visión detallada de cada una de las tecnologías.

Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
	<Fase de documentación>	95 días	lun 02/02/15	dom 14/06/15	
★	Búsqueda de información	6 días	lun 02/02/15	dom 08/02/15	
★	Documentación navegadores	11 días	lun 09/02/15	dom 22/02/15	
★	Documentación tecnologías	11 días	lun 23/02/15	dom 08/03/15	
★	Documentación desarrollo plugins	6 días	lun 09/03/15	dom 15/03/15	
★	Documentación vídeo	11 días	lun 16/03/15	dom 29/03/15	
★	Documentación cifrado	6 días	lun 30/03/15	dom 05/04/15	
★	Documentación autenticación	6 días	lun 06/04/15	dom 12/04/15	
★	Revisión y corrección de la documentación	11 días	lun 13/04/15	dom 26/04/15	
★	Revisión código FBVLC	8 días	lun 27/04/15	mié 06/05/15	
★	Implementación de un plugin de vídeo	10 días	jue 07/05/15	mié 20/05/15	
★	Control de procesos desde la web	8 días	jue 21/05/15	dom 31/05/15	
★	Revisión protocolos RTSP y RTP	11 días	lun 01/06/15	dom 14/06/15	

Figura 4.5: Fase de documentación.

Hemos incluido en esta fase algunas tareas más prácticas, como el uso y la prueba de plugins de vídeo para la reproducción del mismo, y el uso y la investigación de lenguajes como PHP o

Python para tratar de controlar los procesos desde el navegador. Hemos decidido dar poco peso a estas tareas dentro de esta memoria porque no se consiguieron resultados satisfactorios. El único fin de estas tareas ha sido descartar el uso de determinadas tecnologías, ya que no nos permitían cumplir con los objetivos establecidos.

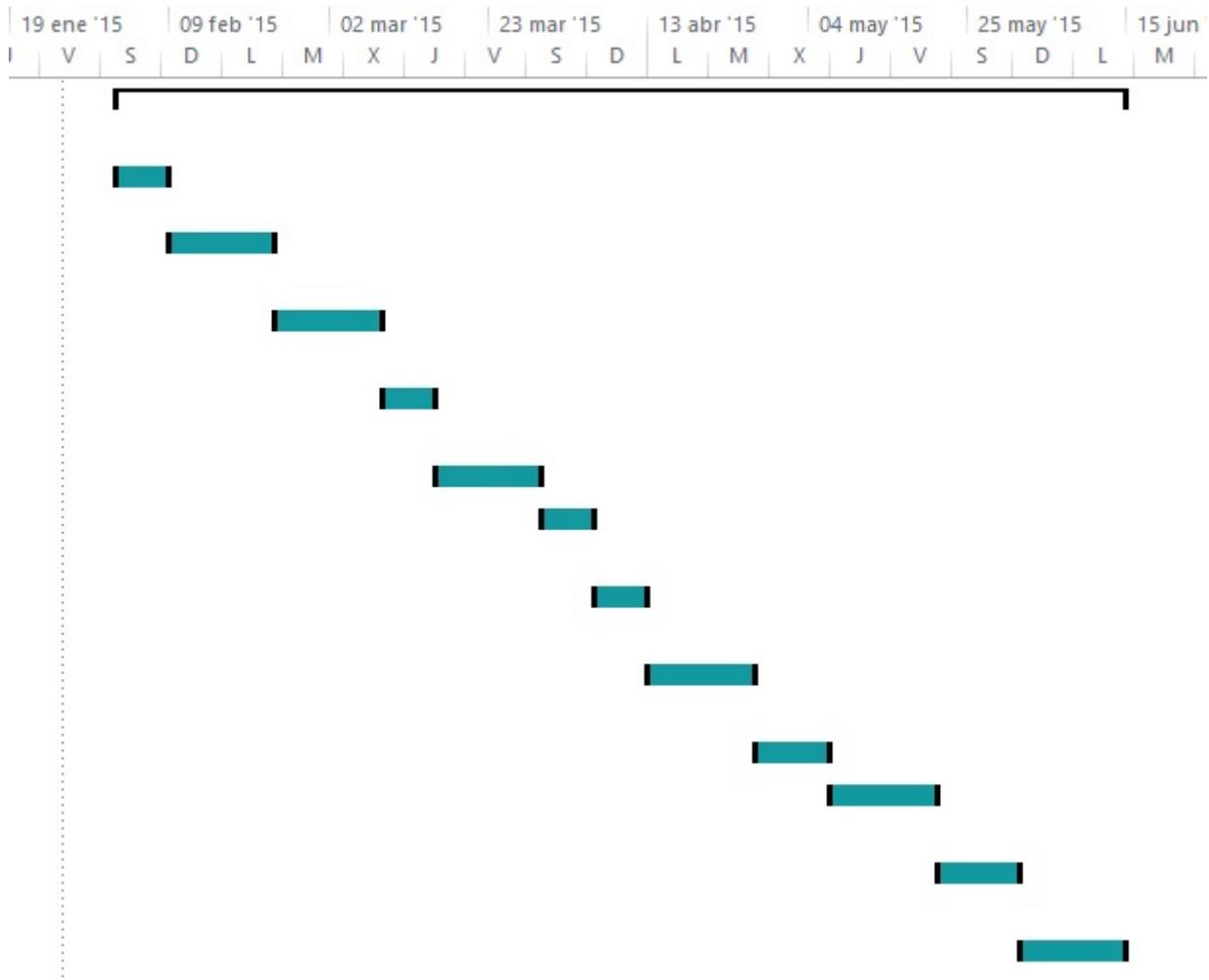


Figura 4.6: Diagrama fase de documentación.

4.3.3. Fase de diseño

En esta fase se ha realizado el planteamiento y la descripción del modelo de desarrollo utilizado, de la arquitectura de la aplicación implementada y de la interfaz de usuario de la misma.

La fase de diseño depende totalmente de las fase de análisis y sobretodo de la fase de documentación, ya que por ejemplo el modelo de desarrollo y la arquitectura se han tenido que plantear de acuerdo a las tecnologías escogidas en la fase previa de documentación. Y la interfaz de usuario se ha diseñado en función de los objetivos y requisitos previamente establecidos.

<Fase de diseño>	14 días	lun 15/06/15 jue 02/07/15		
Modelo de desarrollo	5 días	lun 15/06/15	vie 19/06/15	9
Arquitectura	5 días	sáb 20/06/15	jue 25/06/15	9
Interfaz de usuario	5 días	vie 26/06/15	jue 02/07/15	

Figura 4.7: Fase de diseño.

Se ha escogido un modelo de desarrollo en cascada, que nos permite ir añadiendo funcionalidad de forma progresiva a nuestro software. La arquitectura utilizada se corresponde con un modelo Cliente-Servidor que utiliza los protocolos RTSP y RTP. Con la interfaz se ha buscado facilitar al usuario el uso de la aplicación ocultando los protocolos y procesos subyacentes.

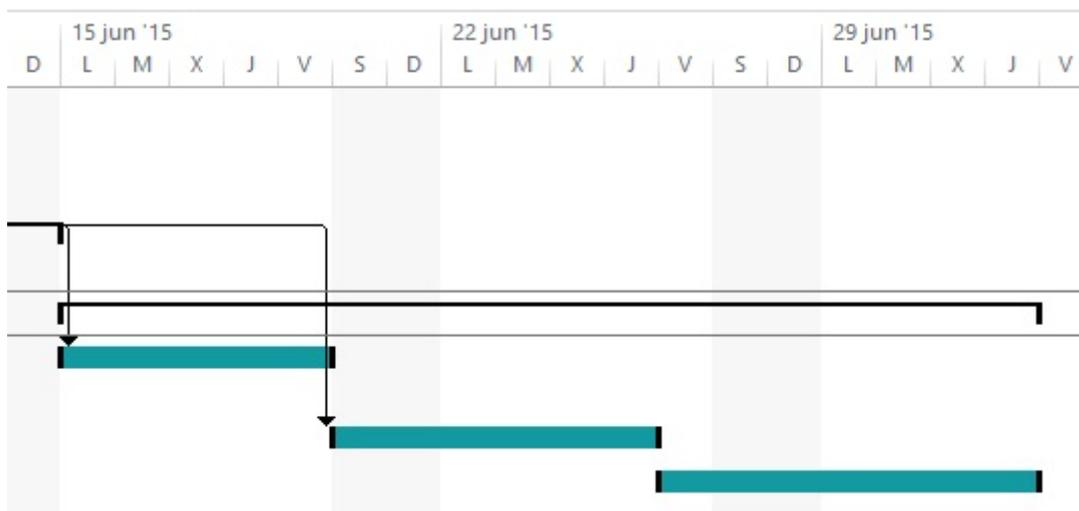


Figura 4.8: Diagrama fase de diseño.

4.3.4. Fase de implementación

La fase de implementación depende en gran medida de las decisiones tomadas durante la fase documentación, en la que decidimos qué implementar, y la fase diseño la arquitectura de la aplicación a implementar y la forma en que se desarrollará la implementación.

Las tareas de esta fase se han realizado por separado, aunque posteriormente se hayan ensamblado las distintas partes de código, porque dentro de la aplicación desarrollada podemos distinguir claramente varias partes:

- Implementar un cliente y un servidor que permitan la comunicación a través del protocolo RTSP y el envío del contenido a través del protocolo RTP.
- Desarrollo de un vigilante que permita realizar un listado los procesos en ejecución en el equipo del usuario e informe de procesos sospechosos.

<Fase de Implementación>	21 días	vie 03/07/15	dom 02/08/15	
Implementación cliente y servidor RTSP	6 días	vie 03/07/15	vie 10/07/15	21
Implementación control de procesos	7 días	vie 10/07/15	sáb 18/07/15	20
Implementación reproductor	6 días	lun 20/07/15	dom 26/07/15	
Implementación interfaz de usuario	6 días	lun 27/07/15	dom 02/08/15	25
Implementación web descarga	6 días	lun 27/07/15	dom 02/08/15	

Figura 4.9: Fase de implementación.

- Implementación de un reproductor que permita mostrar el contenido que el cliente recibe desde el servidor.
- Implementar de un interfaz de usuario que permita al usuario acceder al contenido e interactuar con el mismo.
- Crear un página web que permita al usuario la descarga del software que necesita para la visualización del contenido, es decir el cliente, el reproductor y la interfaz. Aunque para el usuario serán una misma aplicación.

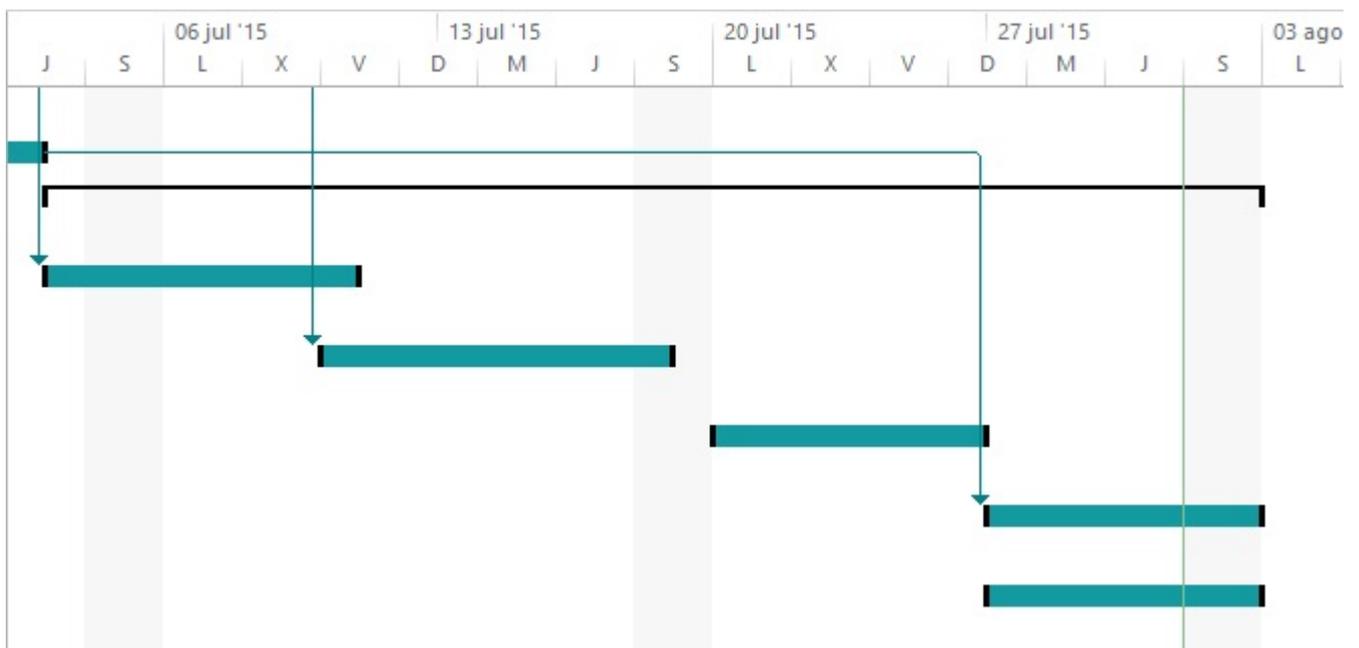


Figura 4.10: Diagrama fase de implementación.

4.3.5. Fase de pruebas

En esta fase realizamos las pruebas pertinentes para comprobar el correcto funcionamiento y la corrección de la aplicación implementada en la fase anterior. Esta es la fase más pequeña del trabajo, tanto en duración como en horas de esfuerzo. Es razonable en este trabajo, dado que incluso la fase de implementación es reducida en comparación con el tiempo y el esfuerzo dedicado a la investigación de tecnologías y a la documentación. En parte, la duración de esta fase se ve reducida porque durante la fase de implementación se ha ido probando el funcionamiento de la aplicación.

▲ <Fase de pruebas>	5 días	lun 03/08/15 dom 09/08/15		
Pruebas protocolos	6 días	lun 03/08/15	dom 09/08/15	27
Pruebas control de procesos	6 días	lun 03/08/15	dom 09/08/15	28
Pruebas reproducción de vídeo	6 días	lun 03/08/15	dom 09/08/15	29

Figura 4.11: Fase de pruebas.

Dividimos nuestra fase de pruebas en tres tareas principales, ya que realizamos por separado las pruebas para ver el correcto funcionamiento de la comunicación a través de los protocolos, del control de los procesos en ejecución y por último de la reproducción del contenido. En esta última tarea de pruebas podemos incluir las pruebas del funcionamiento de la interfaz.

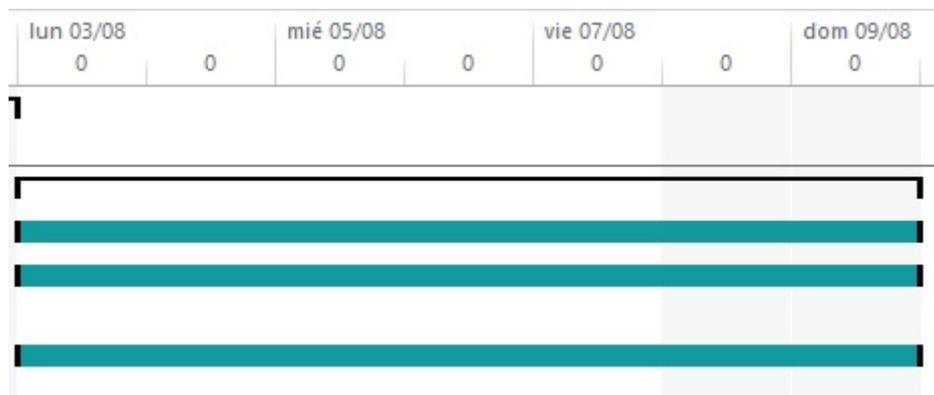


Figura 4.12: Diagrama fase de pruebas.

4.3.6. Otras tareas

En este apartado incluimos tareas difíciles de ubicar en las otras fases, y en concreto la memoria y las correcciones son difíciles de situar en la propia planificación, porque son tareas a las que se ha dedicado tiempo de forma esporádica a lo largo de la realización de todo el Trabajo de Fin de Grado.

▾ Otras tareas	155 días?	lun 19/01/15	dom 23/08/15
Memoria TFG	156 días	lun 19/01/15	dom 23/08/15
Correcciones y cambi	156 días?	lun 19/01/15	dom 23/08/15
Manual de usuario	6 días?	lun 10/08/15	dom 16/08/15

Figura 4.13: Otras tareas.

Como hemos mencionado previamente, es muy difícil situar las tareas dedicadas a la memoria y a los cambios y correcciones realizados dentro de alguna fase, ya que estas tareas se han realizado durante todo el proyecto, con una distribución temporal muy distinta a la de las demás tareas. Estas tareas, junto con la realización del manual de usuario e instalación, podrían verse como una fase de transición dentro de un modelo de planificación iterativo, pero hemos preferido reflejarlo de este modo debido a que la singularidad de nuestro proyecto no nos ha permitido adaptarnos de forma precisa a un modelo de planificación.

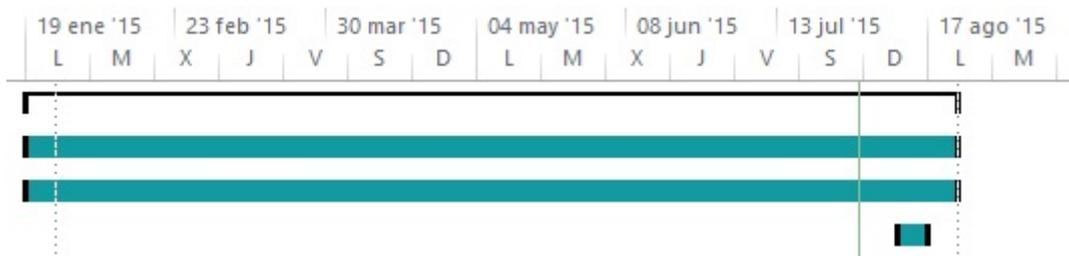


Figura 4.14: Diagrama de otras tareas.

4.4. Presupuesto

Es aconsejable calcular un presupuesto del proyecto, para estimar los gastos realizados o para asignar precio al software de cara a su comercialización. El coste de este proyecto de desarrollo de software se calcula en función del coste de estos tres factores:

- Esfuerzo (horas de trabajo invertidas en el proyecto).
- Hardware y software necesario para el desarrollo del proyecto.
- Comunicaciones y desplazamientos.

La mayor parte de los costes del proyecto corresponden a las horas de trabajo invertidas (esfuerzo), ya que se han dedicado un gran número de horas para la realización del proyecto. Consideraremos como coste de hardware parte del coste del ordenador utilizado para el desarrollo. No tenemos coste de software al utilizar mayoritariamente software gratuito, y en el caso de los programas de pago, no han tenido coste al ser obtenidos con licencia de estudiante. Los costes de viajes y comunicación han sido bastante reducidos, al realizar gran parte de la comunicación por

correo electrónico. El único gasto de viajes ha sido el autobús entre Palencia y Valladolid para la reunión semanal con el tutor.

El coste del esfuerzo lo vamos a calcular con una fórmula bastante sencilla: número de horas x precio de la hora. El precio por hora puede ser un tema controvertido, porque depende normalmente del tipo de puesto, de la empresa, de los años de experiencia y de muchos otros factores. Pero para este proyecto, hemos considerado oportuno un precio de 12 €/hora. Por lo tanto el coste del esfuerzo en este proyecto ha sido:

$$1000 \times 12 = 12000 \text{ € de coste de esfuerzo para el proyecto.}$$

De cara a este presupuesto no contabilizaremos el coste completo del hardware. En primer lugar, porque uno de los ordenadores utilizados (Acer Aspire) tiene más de cuatro años, por lo que le consideramos amortizado. Además, una vez finalizado el proyecto, el otro ordenador (Lenovo) seguirá teniendo vida útil. Calculamos el coste del hardware para el proyecto de la siguiente forma: dividimos el precio de compra del ordenador (600 €) entre el periodo de duración normal de un ordenador (cuatro años) y los multiplicamos por la duración del proyecto (aprox. siete meses). Por lo tanto:

$$(600 / 4) \times 0.6 = 90 \text{ € de coste de hardware para el proyecto.}$$

El software utilizado ha sido mayoritariamente gratuito, pero cabe destacar que hemos utilizado tres programas de pago como son: Microsoft Project Professional 2013, Visual Studio Professional 2013 y Astah Professional. Gracias a un convenio de la Universidad de Valladolid con Microsoft hemos podido obtener estos programas de forma gratuita, lo que supone un ahorro considerable, ya que el precio de Microsoft Project y Visual Studio es respectivamente de 1369 y 1553 €, y el de Astah Professional de 229 €. Tampoco contabilizamos el coste de los Sistemas Operativos utilizados, puesto que Ubuntu es gratuito y el coste de Windows está incluido en el precio del ordenador Lenovo.

Si este proyecto se realizase en el ámbito de la empresa deberíamos tener en cuenta otros factores que generan coste, como costes de electricidad o el coste del personal de la empresa que no está directamente relacionado con el proyecto. Así como gastos de seguridad social, seguros de la empresa o gastos de mantenimiento de la misma. Pero dado que este proyecto es de ámbito académico y que se ha desarrollado mayoritariamente en la universidad o en bibliotecas públicas, no tendremos en cuenta este tipo de gastos para el presupuesto.

Concepto	Cantidad	Precio (€)
Software		
TeXworks	1	0,0
TeXLive	1	0,0
Visual Studio 2013	1	0,0
FireBreath	1	0,0
Microsoft Project 2013	1	0,0
Cmake 3.1.0	1	0,0
Python	1	0,0
Xampp	1	0,0
Handbrake	1	0,0
Java SE Development Kit	1	0,0
Notepad	1	0,0
Astah Professional	1	0,0
NetBeans IDE 8.0.2	1	0,0
Jar2Exe	1	0,0
Ubuntu 14.04 LTS 32 bits	1	0,0
Windows 8.1 pro 64 bits	1	0,0
<i>Subtotal</i>		0,0
Hardware		
Acer Aspire-5715Z Intel Pentium Dual CPU T2390	1	0,0
Lenovo G50 i7-4500U 8 GB RAM	1	90,0
<i>Subtotal</i>		90,0
Esfuerzo		
Horas de trabajo	1000	12
<i>Subtotal</i>		12000,0
Desplazamientos		
Bono de autobús Palencia-Valladolid 40 viajes	1	100,0
<i>Subtotal</i>		100,0
Total		12190,0

Tabla 4.2: Costes del proyecto

Capítulo 5

Diseño

5.1. Descripción de la arquitectura

En esta sección describimos el diseño arquitectónico del sistema desarrollado. Se trata de un sistema compuesto por dos aplicaciones, un reproductor (o cliente) y un servidor. El servidor se compone de un único paquete y el reproductor de dos: controlador e interfaz. Las clases de cada paquete se mostrarán en las próximas secciones.

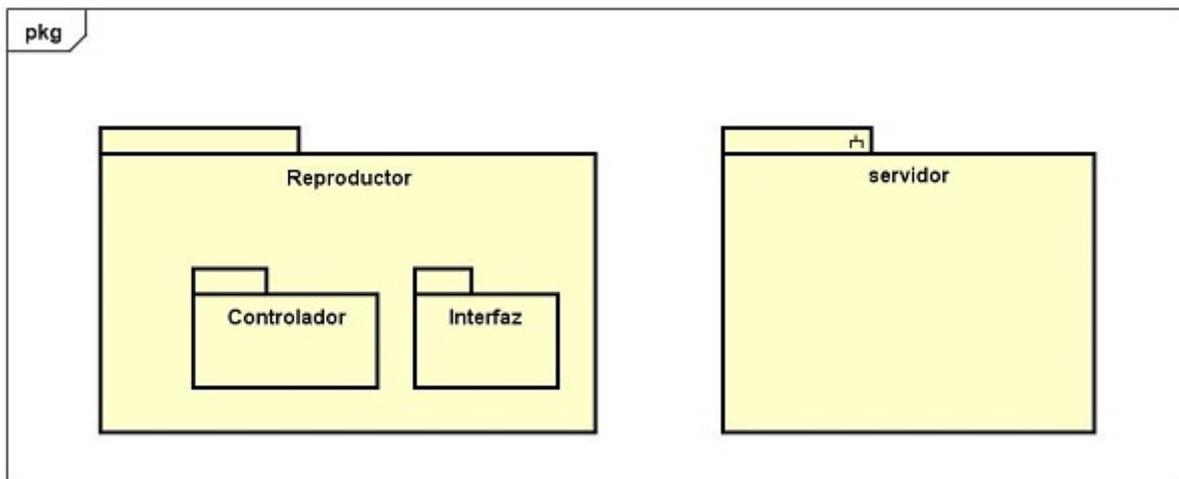


Figura 5.1: Arquitectura del sistema.

5.2. Patrones de arquitectura

En esta sección mostramos los patrones de diseño que se han tenido en cuenta para crear nuestro sistema[58].

5.2.1. Experto

Se trata de un principio básico de asignación de responsabilidades. Este patrón designa la responsabilidad correspondiente a la clase que tiene la información para realizar las acciones. De esta forma, cada clase solo realiza las acciones para las que se ha diseñado.

En nuestro caso, este patrón se aplica tanto en el lado del cliente como en el lado del servidor, *RTPpacket*, *VideoStream* o *Vigilante* son ejemplos de clases destinadas a realizar una acción concreta para la que disponen información.

5.2.2. Controlador

Este patrón sirve como intermediario entre una interfaz y algoritmo que la implementa, de forma que es esta clase la que recibe los datos y realiza las acciones. De esta forma, se separa la lógica de negocios de la capa de presentación. Así, la interfaz únicamente recoge los eventos del usuario y muestra los resultados de las acciones realizadas por el controlador.

En nuestro sistema se aplica este patrón entre las clases *Reproductor* y *Controlador*, de forma que la primera recoge los eventos y muestra el contenido, y la segunda se encarga de la comunicación con el servidor y sus correspondientes acciones.

5.2.3. Estado

Este patrón se utiliza cuando, dependiendo del estado de un objeto, su comportamiento cambia. En nuestro caso tanto en el lado del cliente, como en el lado del servidor, dependiendo del estado RTSP, se realizan distintas funciones.

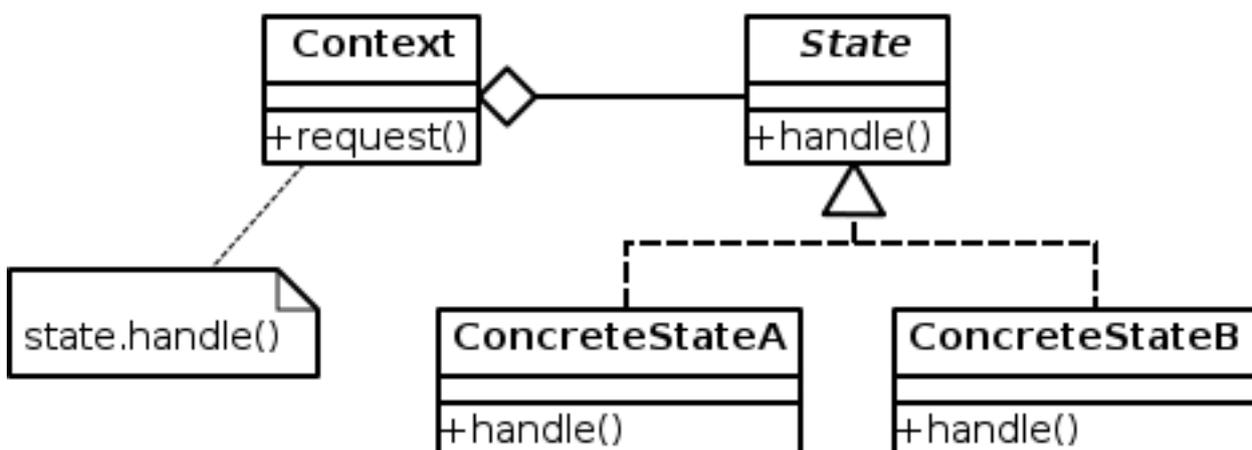


Figura 5.2: Patrón estado.

5.3. Modelo estructural

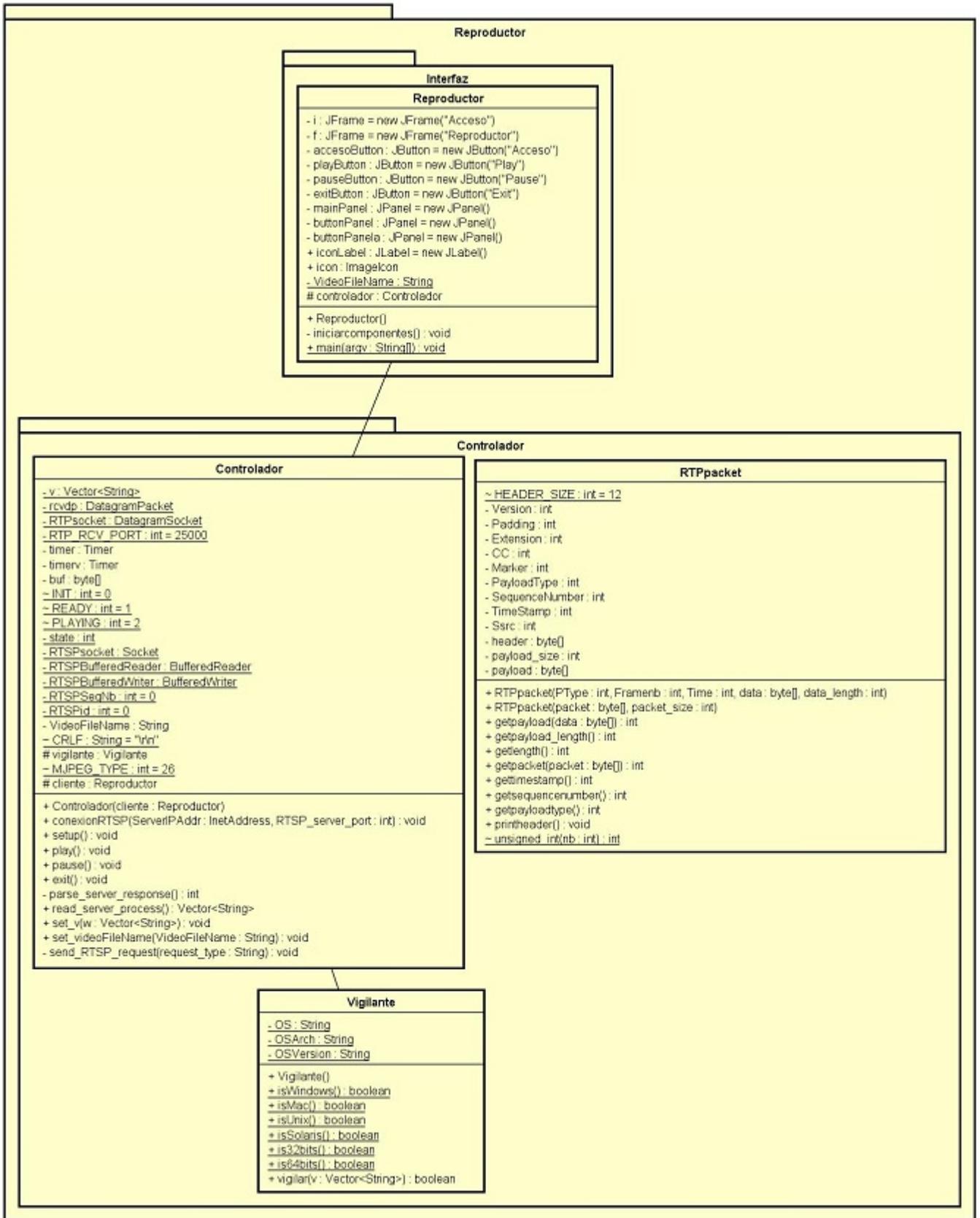


Figura 5.3: Diagrama de clases del lado del cliente.

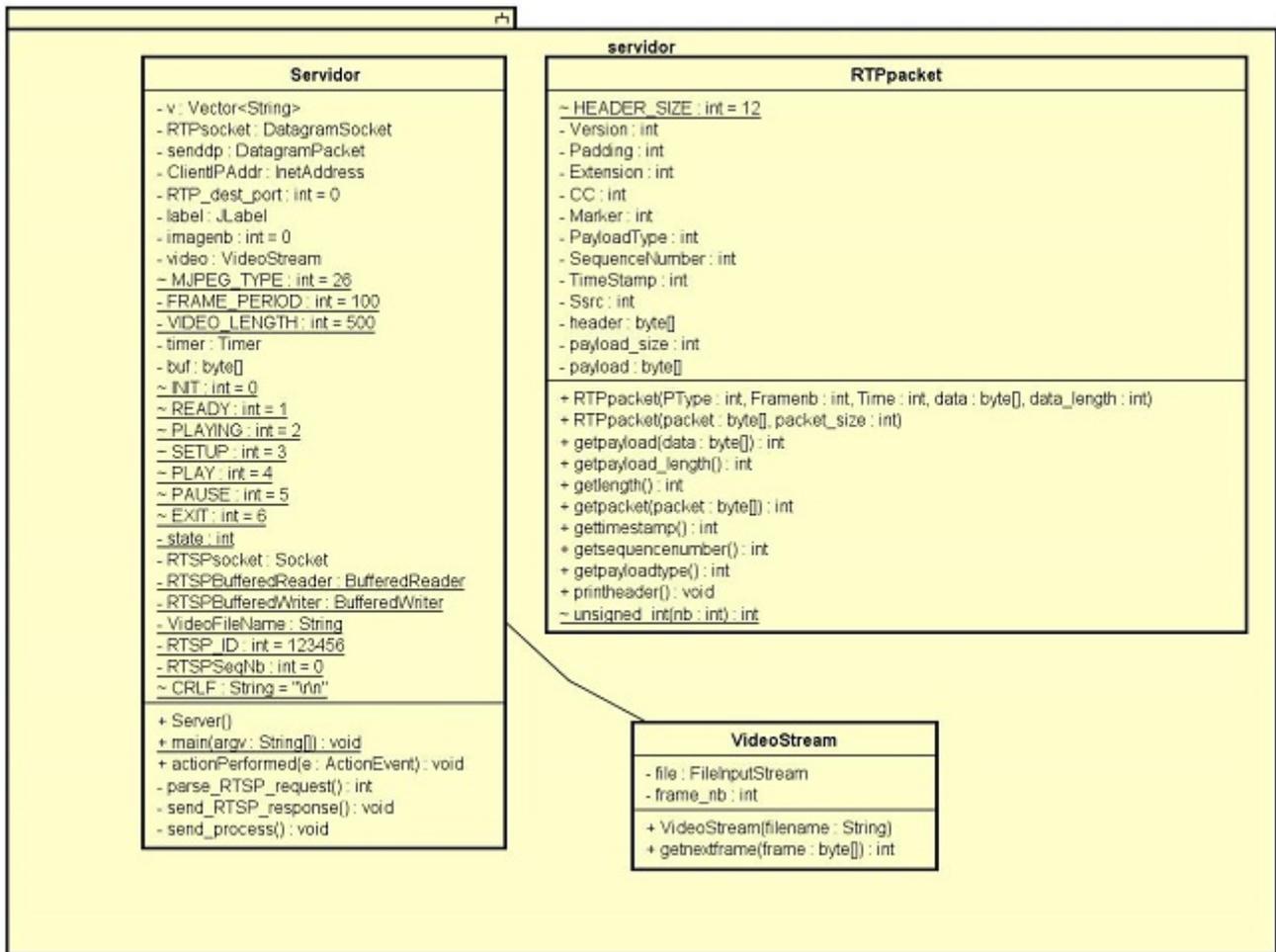


Figura 5.4: Diagrama de clases del lado del servidor.

5.4. Diseño de la interfaz de usuario

A continuación mostramos las interfaces diseñadas para las distintas partes de nuestra aplicación. Hemos buscado que la interfaz sea sencilla e intuitiva.

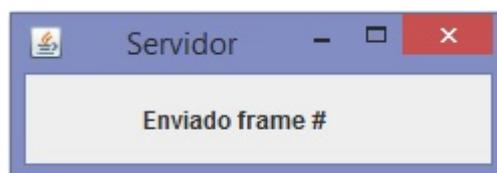


Figura 5.5: Interfaz del servidor.

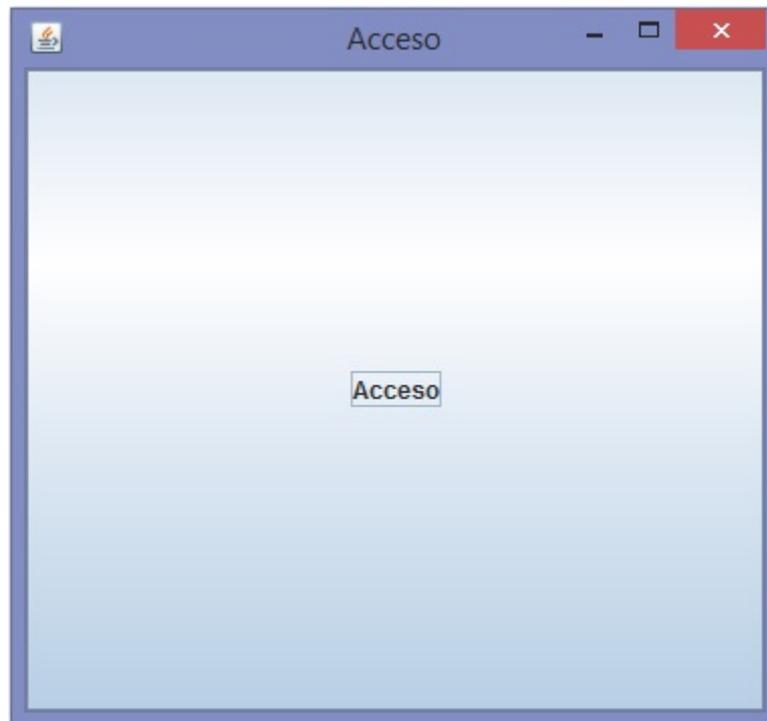


Figura 5.6: Interfaz de acceso.



Figura 5.7: Interfaz del reproductor.

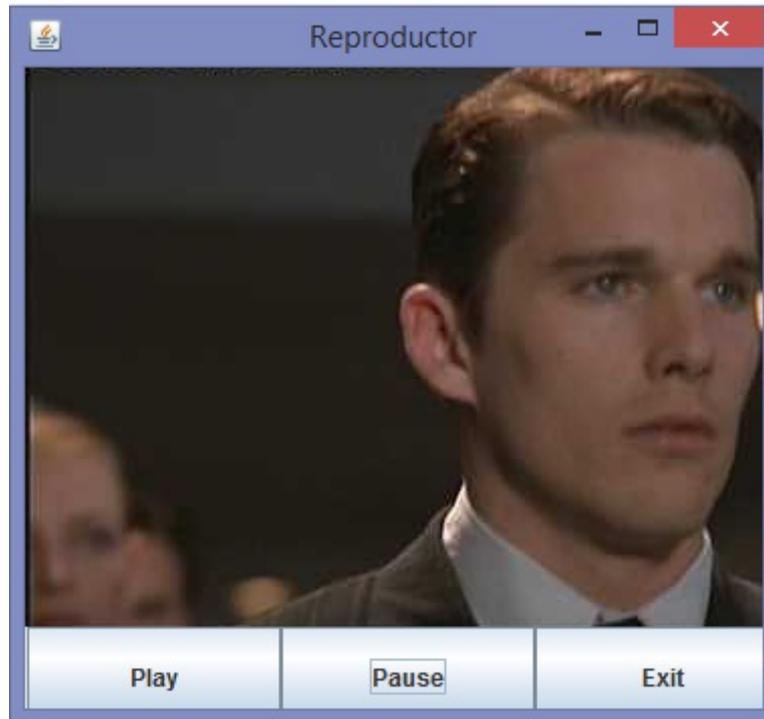


Figura 5.8: Interfaz del reproductor mostrando el contenido.

5.5. Diagramas de secuencia

Para el caso de uso intento de copia no se ha realizado diagrama de secuencia, puesto que la apertura de un proceso sospechoso produce acciones por parte de nuestro sistema pero de forma indirecta. Y la parte del envío del listado de procesos y de la vigilancia de procesos se muestra en los casos de uso de acceso y play.

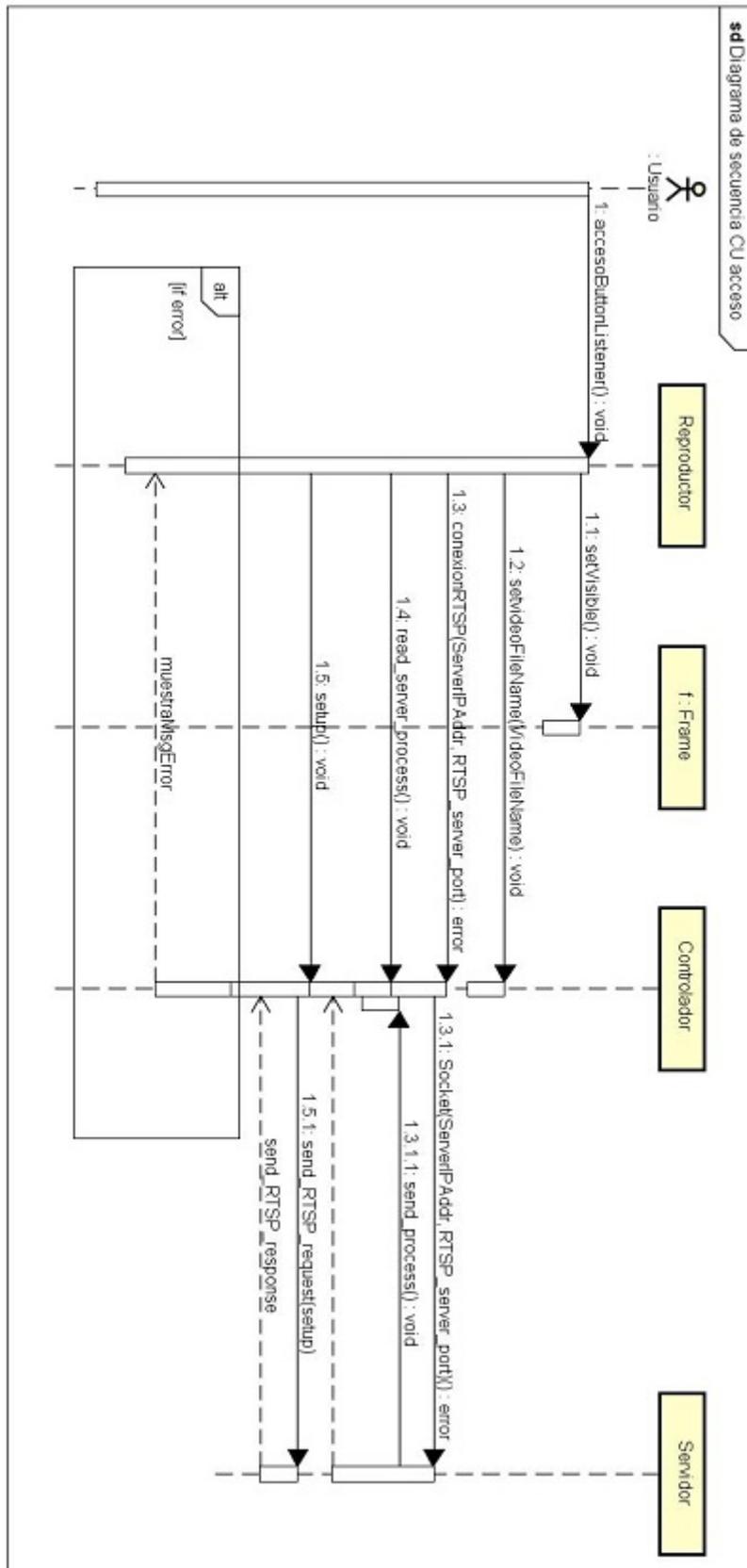


Figura 5.9: Diagrama de secuencia CU acceso.

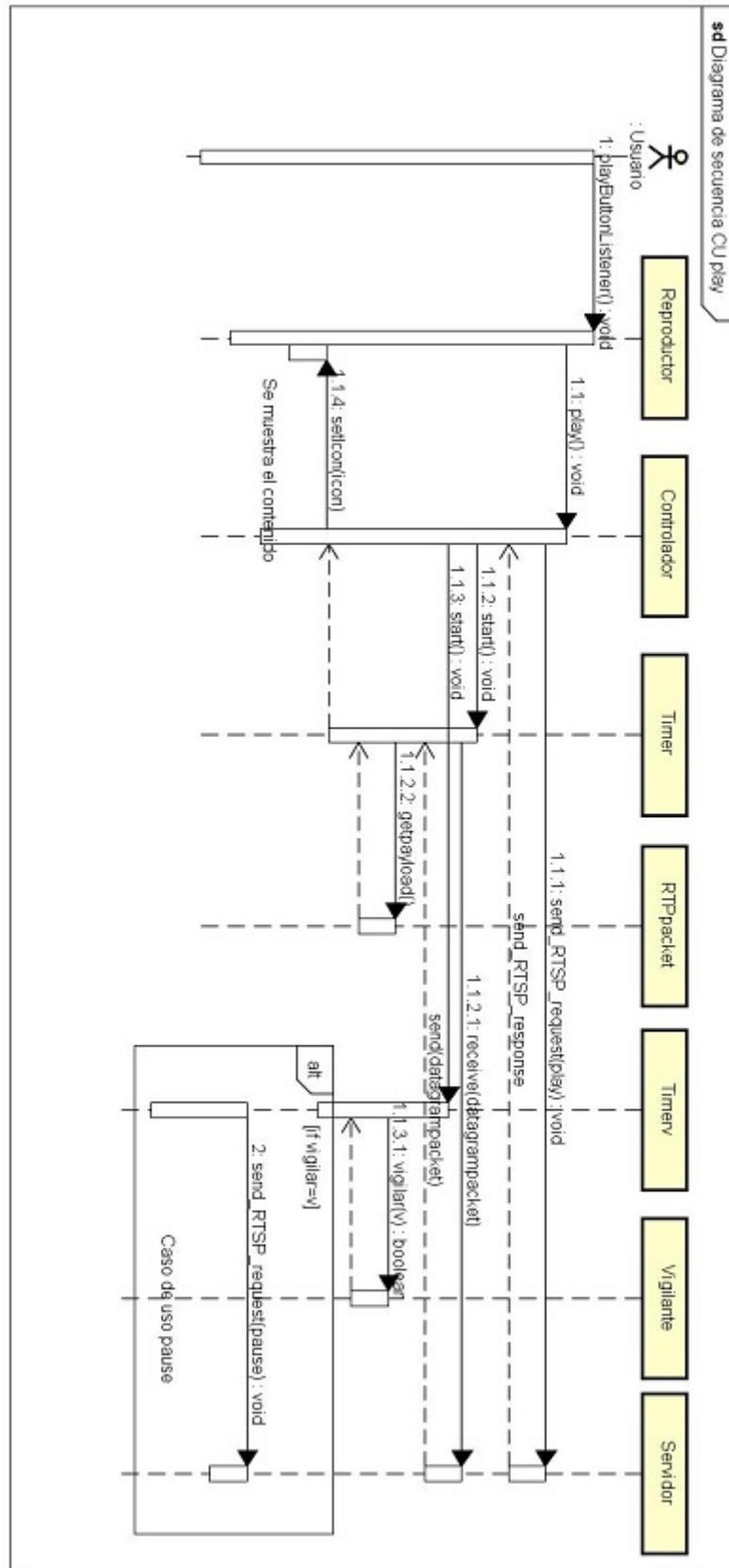


Figura 5.10: Diagrama de secuencia CU play.

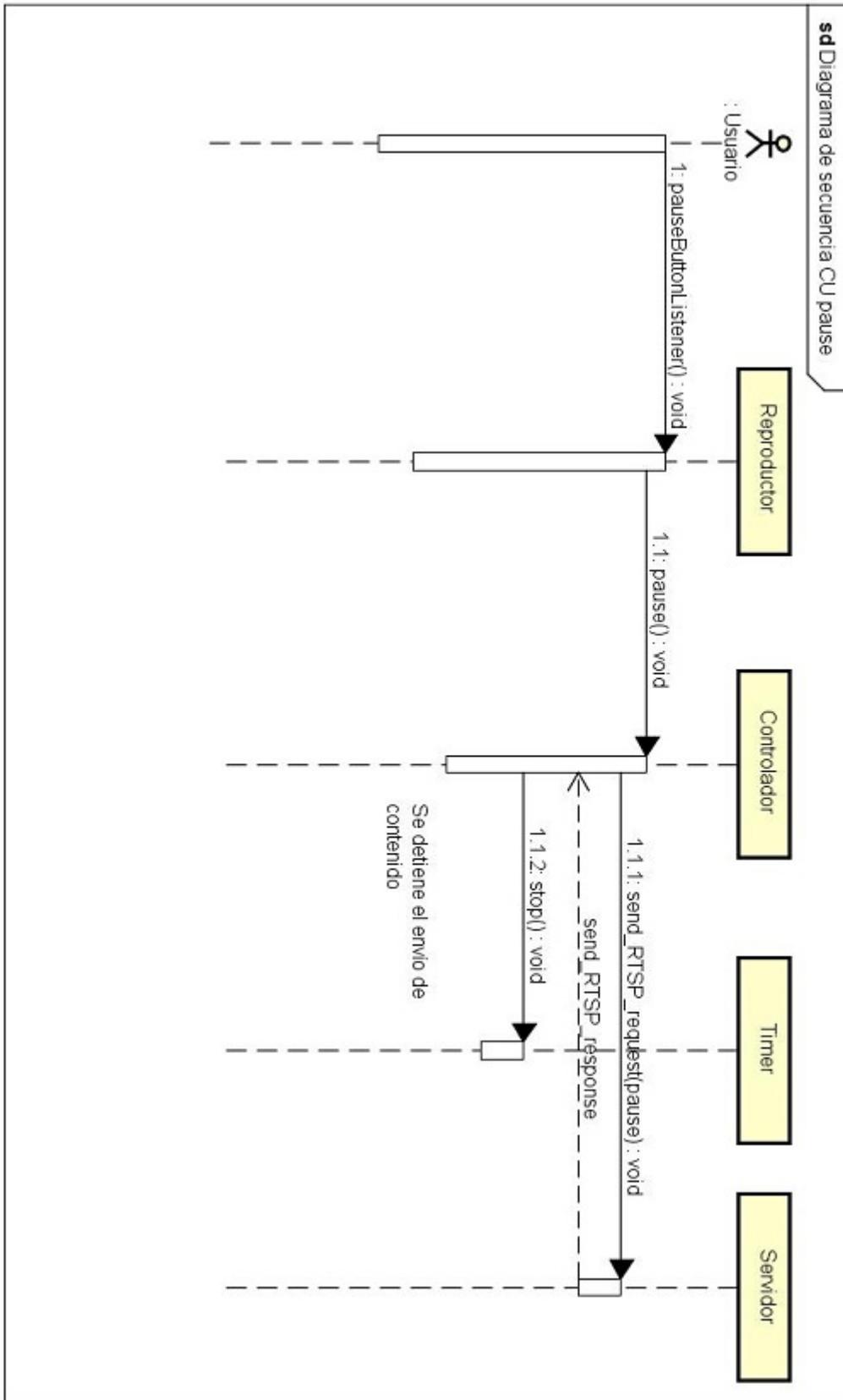


Figura 5.11: Diagrama de secuencia CU pause.

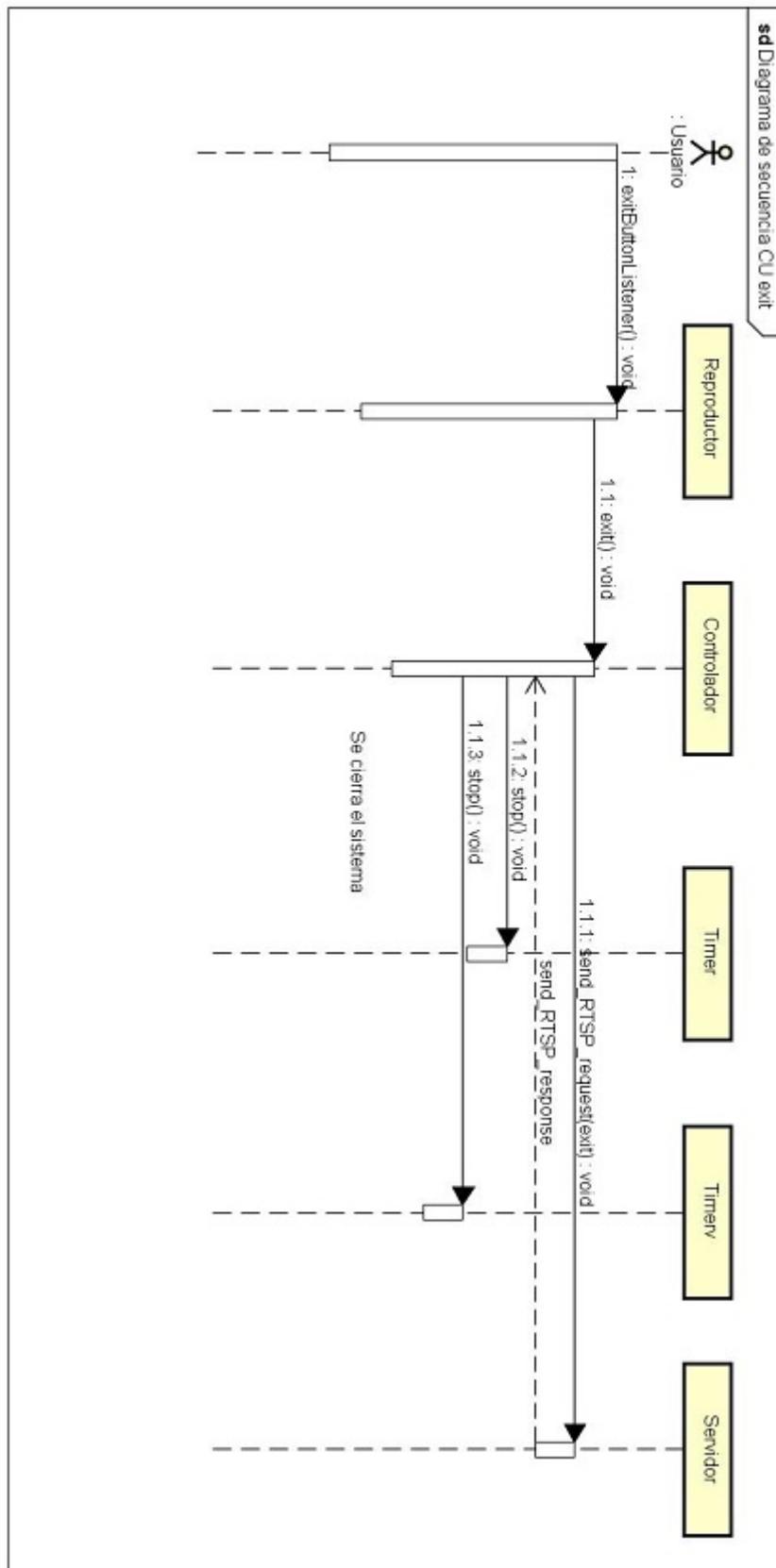


Figura 5.12: Diagrama de secuencia CU exit.

Capítulo 6

Implementación

6.1. Modelo de desarrollo

La implementación de nuestro sistema se ha realizado mediante desarrollo en cascada. Podemos dividir la implementación en las siguientes actividades:

- Implementación del cliente y el servidor RTSP, además de la correspondiente comunicación entre ellos.
- Desarrollo de un vigilante para el control de procesos en distintos sistemas operativos.
- Creación del reproductor del contenido e implementación del protocolo RTP que se encarga del envío de dicho contenido.
- Implementación de la interfaz de usuario.
- Implementación de la web de descarga para la descarga de nuestro sistema.

La implementación inicial del cliente y el servidor RTSP así como del reproductor, se ha basado varias clases incompletas creadas por UMBC (University of Maryland Baltimore County) [56]. En nuestro sistema se mantiene parte de ese código, aunque la mayor parte de este código se ha modificado o completado para adaptarlo a nuestras necesidades.

6.2. Entorno de desarrollo

Para el desarrollo de este Trabajo de Fin de Grado se han utilizado los siguientes entornos de desarrollo:

- **NetBeans IDE 8.0.2:** inicialmente NetBeans iba a ser nuestro único entorno de desarrollo, pero la ejecución de nuestro código en este entorno de desarrollo era incompleta. No nos permitía la reproducción del contenido, ni tampoco la ejecución de los archivos tasklist.exe para el listado de procesos en Windows. A pesar de estos problemas, hemos aprovechado las utilidades de NetBeans para realizar correcciones en nuestro código.

- **Notepad++:** hemos utilizado esta herramienta de edición de texto y código para escribir nuestro sistema. Para compilar y ejecutar hemos utilizado otra herramienta.
- **Java SE Development Kit:** este software proporciona herramientas de desarrollo para poder crear programas en Java [59]. Hemos utilizado estas herramientas, junto con el Procesador de comandos de Windows y el Terminal de Ubuntu, para la compilación y ejecución de nuestro código.
- **Visual Studio 2013:** este entorno de desarrollo integrado ha sido utilizado para la creación de plugins para navegadores durante el estudio de tecnologías realizado en el capítulo 2.

6.3. Lenguajes de programación

Toda la implementación de nuestro sistema se ha realizado con Java, aunque hemos utilizado otros lenguajes para la creación de una sencilla página web para la descarga de nuestro software y para la pruebas realizadas durante el estudio de las tecnologías realizado en el capítulo 2. Describimos brevemente los lenguajes utilizados:

- **Java:** es un lenguaje de programación de propósito general, orientado a objetos y multiplataforma, desarrollado por Sun Microsystems durante los 90. Java tiene mucha sintaxis de C y C++, pero elimina herramientas de bajo nivel y simplifica el modelo de objetos. Es un lenguaje con un uso muy extendido, particularmente aplicaciones cliente-servidor en la web, por lo que resulta adecuado para el desarrollo de este proyecto. Como anteriormente hemos mencionado, todo nuestro sistema se ha implementado con Java[60], tanto los protocolos de comunicación y transmisión como la interfaz y el control de procesos.
- **HTML:** se trata de un lenguaje de marcado utilizado para la creación de páginas web. Es un estándar de W3C que permite definir la estructura y el contenido de una página web. Su uso en este proyecto ha tenido dos fines: la creación de una web sencilla para la descarga del software del proyecto y la implementación de páginas web para probar el funcionamiento de los plugins para navegadores web.
- **CSS:** es un lenguaje utilizado para definir la presentación de un documento HTML y de esta forma separar la presentación y la estructura de dicho documento. En este proyecto se ha utilizado únicamente para dar estilo a la página web de descarga del software.
- **PHP:** se trata de un lenguaje del lado del servidor de uso general, diseñado para el desarrollo de contenido dinámico en la web. Se ha utilizado para intentar listar los procesos en ejecución desde el navegador.
- **C++:** este lenguaje se creó como una extensión de C. Se trata de un lenguaje multiparadigma y orientado a objetos. Su uso en este proyecto ha sido para el desarrollo de plugins para navegadores web.

- **Python:** es un lenguaje interpretado, multiplataforma y multiparadigma. Se ha utilizado en las pruebas realizadas durante el capítulo 2, tanto para el desarrollo de plugins para navegadores como para el control de procesos desde la web.

6.4. Protocolos de comunicación

En el capítulo 2 ya hemos realizado una descripción en detalle del protocolo RTSP (Real Time Streaming Protocol) y del uso que hace RTSP de otros protocolos, como son RTP, TCP y UDP. Pero en este apartado queremos especificar el uso que hace nuestro sistema de estos protocolos:

- **RTSP:** se trata de un protocolo de nivel de aplicación, no orientado a conexión y que mantiene un estado con un identificador asociado a cada sesión. En nuestro sistema se ha implementado este protocolo para la comunicación entre el cliente y el servidor mediante el envío de mensajes. Se realiza la comunicación entre las clases *Servidor* y *Controlador* a través del uso de sockets TCP.
- **RTP:** este protocolo de nivel de sesión se utiliza para transmitir información, como audio y vídeo, en tiempo real. Aunque se trata de un protocolo de nivel de sesión tiene algunas características propias de un protocolo de nivel de transporte, pero la información se transporta utilizando UDP. En nuestro sistema *Servidor* se encarga de enviar los paquetes RTP como datagramas a través un socket UDP. *Controlador* se encarga de recibir estos datagramas a través de un socket UDP. Además obtiene la carga útil de estos paquetes y lo transforma en las imágenes que podemos ver en el reproductor.
- **TCP:** es uno de los protocolos fundamentales de internet. Se trata de un protocolo a nivel de transporte que permite un flujo de bits fiable entre aplicaciones. Como anteriormente hemos mencionado, en nuestro sistema los mensajes RTSP se envían mediante el uso de sockets TCP.
- **UDP:** se trata de un protocolo a nivel de transporte que se basa en el intercambio de datagramas. Aunque no ofrece las garantías en el transporte que ofrece TCP, este protocolo es muy utilizado para el envío de audio y vídeo. En nuestro sistema los paquetes RTP son enviados como datagramas a través de sockets UDP.

Aunque en nuestro código aparecen comentados, hemos querido mantener los *println*, que si se descomentan permiten mostrar el intercambio de mensajes entre el cliente y el servidor. Hemos preferido mantener ocultos estos mensajes, ya que el usuario no necesita conocer el funcionamiento de los protocolos subyacentes. Además estos mensajes ralentizan la reproducción del contenido.

6.5. Control de procesos

El control de los procesos en ejecución en el equipo del usuario es una parte importante de nuestro sistema. Se trata de realizar un listado de los procesos a través de los ejecutables *tas-*

klist32.exe o *tasklist64.exe* si se trata respectivamente de un sistema operativo Windows de 32 o 64 bit, o del comando *ps -e* en el caso de los demás sistemas operativos. Para poder listar de forma adecuada los procesos se realiza de forma previa una comprobación del sistema operativo utilizado en el equipo del usuario.

Una vez listados los procesos se comprueba si alguno de los procesos listados coincide con los procesos sospechosos de realizar la grabación de la pantalla. Este listado de procesos sospechosos se envía desde *Servidor* a *Controlador*. Hemos querido que este listado se envíe desde el servidor para poder modificar dicho listado de procesos en el servidor sin que esto implique actualizar el software del lado del cliente. Si alguno de los procesos en ejecución coincide con algún proceso sospechoso, se detiene la reproducción del contenido, o en caso de estar pausada, se impide su reanudación.

A continuación mostramos cómo se añaden los procesos a la lista de sospechosos:

```
//Creamos un vector sin tamaño fijo para poder añadir los procesos a vigilar
v = new Vector<String>();

//Añadimos los procesos
v.add("vlc");
v.add("camstudio");
v.add("jing");
v.add("webineria");
v.add("wink");
v.add("ultravnc");
v.add("windows media encoder");
v.add("bb flashback");
v.add("capturefox");
v.add("utipu tipcam");
v.add("krut");
v.add("hypercam");
v.add("expression encoder");
v.add("ezvid");
v.add("freezeesr");
v.add("istanbul");
v.add("camtasia");
v.add("snaglt");
v.add("recordmydesktop");
v.add("activepresenter");
v.add("cropper");
v.add("tasksi");
v.add("xfire");
v.add("virtualdub");
v.add("ffmpeg");
v.add("ocam");
v.add("smrecorder");
v.add("gamemaster");
v.add("atube catcher");
v.add("screenpresso");
v.add("screen");
v.add("recorder");
v.add("encoder");
```

Figura 6.1: Creación del listado de procesos sospechosos.

Hemos creado el listado a través de un vector, ya que consideramos que para el número de procesos en la lista, almacenar esta información en un archivo o en una base datos hubiese supuesto una complicación innecesaria.

A continuación mostramos el envío por parte de *Servidor* del listado de procesos y de la recepción de dicho listado por *Controlador*:

```
//send_process
private void send_process() { // envía los procesos sospechosos al cliente.
    try {

        RTSPBufferedWriter.write(v.size()+CRLF);
        for(int i=0;i<v.size();i++){
            RTSPBufferedWriter.write(v.elementAt(i)+CRLF);
        }
        RTSPBufferedWriter.flush();
        //System.out.println("El servidor RTSP envia los procesos al cliente.");
    }
    catch(Exception ex) {
        System.out.println("Excepcion: "+ex);
        System.exit(0);
    }
}
}
```

Figura 6.2: Envío del listado de procesos sospechosos.

```
//read_server_process
public Vector<String> read_server_process()
{ // analiza mensaje del servidor con los procesos
    Vector<String> w;
    w = new Vector<String>();
    try{
        String sizev = RTSPBufferedReader.readLine();
        int size=Integer.parseInt(sizev);
        for (int i=0;i<size;i++){
            w.addElement(RTSPBufferedReader.readLine());
        }
    }
    catch(Exception ex)
    {
        System.out.println("Excepcion : "+ex);
        System.exit(-1);
    }
    return w;
}
}
```

Figura 6.3: Recepción del listado de procesos sospechosos.

Para terminar este apartado de Control de procesos, mostramos la parte de la función *vigilar* de la clase *Vigilante* en la que se realiza el listado de los procesos en ejecución y la comprobación de estos procesos con los de la lista de sospechosos. De forma previa a esta comprobación de procesos, se han realizado comprobaciones sobre el sistema operativo para escoger el comando a ejecutar:

```

try{
    // Ejecución del Comando
    Process proceso = Runtime.getRuntime().exec(comando);

    //Leemos la salida del comando
    InputStreamReader entrada = new InputStreamReader(proceso.getInputStream());
    BufferedReader stdInput = new BufferedReader(entrada);

    //Comprobamos la salida del comando.
    if((salida=stdInput.readLine().toLowerCase()) != null)
    {
        //System.out.println("Comando ejecutado correctamente");
        while ((salida=stdInput.readLine()) != null)
        {
            for(int i=0;i<v.size();i++){
                if (salida.indexOf(v.elementAt(i)) >= 0)
                {
                    proc=true;
                }
            }
        }
    }else
    {
        //System.out.println("No se ha producido salida");
    }
}catch (IOException ioe) {
    System.out.println("Excepcion: ");
    ioe.printStackTrace();
}
return(proc); //El valor devuelto por la función determina si hay
//o no procesos sospechosos de capturar el vídeo.

```

Figura 6.4: Comprobación del listado de procesos sospechosos.

6.6. Reproducción y transmisión del contenido

El contenido se transforma en múltiples arrays de bytes en el lado del servidor, para poder ser enviados en paquetes RTP. Esto se realiza con la clase *VideoStream* que, a partir del nombre del archivo con el contenido multimedia, realiza la lectura de dicho archivo y la transformación de cada frame del contenido en un array de bytes. En la clase *Servidor* se crea un objeto de tipo *VideoStream* para realizar la transformación del contenido.

Después, estos paquetes RTP son recibidos en el lado del cliente. Se obtiene la carga útil de estos paquetes y por último se transforma en imágenes que se muestran en el reproductor.

A continuación mostramos la función *getNextframe* de la clase *VideoStream* utilizada para transformar el contenido:

```

// getNextframe
//devuelve el siguiente frame como un array de bytes y devuelve el tamaño del frame

public int getNextframe(byte[] frame) throws Exception
{
    int length = 0;
    String length_string;
    byte[] frame_length = new byte[5];

    //lee el tamaño actual del frame
    file.read(frame_length,0,5);

    //transforma el tamaño del frame en un entero
    length_string = new String(frame_length);
    length = Integer.parseInt(length_string);

    return(file.read(frame,0,length));
}

```

Figura 6.5: Transformación del contenido.

La clase *Controlador* se encarga de recibir los paquetes RTP, de transformarlos en imágenes y de mostrar estas imágenes en *Reproductor*. Para que estas acciones se realicen de forma periódica cada pocos milisegundos se realizan con Timer:

```

class timerListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        timer.start(); // Arrancamos timer
        //Construye un DatagramPacket para recibir datos del socket UDP
        rcvdp = new DatagramPacket(buf, buf.length);

        try{
            //recibe el DatagramPacket desde el socket :
            RTPsocket.receive(rcvdp);
            //crea un objeto RTPpacket a partir del DatagramPacket
            RTPpacket rtp_packet = new RTPpacket(rcvdp.getData(), rcvdp.getLength());
            //get the payload bitstream from the RTPpacket object
            int payload_length = rtp_packet.getpayload_length();
            byte [] payload = new byte[payload_length];
            rtp_packet.getpayload(payload);

            //obtiene un objeto imagen de the payload bitstream
            Toolkit toolkit = Toolkit.getDefaultToolkit();
            Image image = toolkit.createImage(payload, 0, payload_length);

            //muestra la imagen como un objeto ImageIcon
            cliente.icon = new ImageIcon(image);
            cliente.iconLabel.setIcon(cliente.icon);
        }
    }
}

```

Figura 6.6: Transformación de los paquetes RTP en imágenes.

6.7. Estructura del código

En este apartado vamos a listar y describir brevemente las clases de las que se compone nuestro sistema:

■ Lado del servidor:

- **Servidor.java:** esta clase se encarga de recibir las solicitudes RTSP enviadas desde el lado del cliente. Además realiza el envío de las correspondientes respuestas a estas solicitudes y de enviar los paquetes RTP con el contenido. Por último, esta clase envía la lista de procesos sospechosos.
- **RTPpacket.java:** se utiliza tanto en el cliente como en el servidor. En este caso, en el servidor se utiliza para la creación de los paquetes RTP.
- **VideoStream.java:** es la clase encargada de leer el contenido y transformarlo para su envío.

■ Lado del cliente:

- **Controlador.java:** es la clase más importante en el lado del cliente. Se encarga de enviar las solicitudes RTSP al servidor. Además recibe las correspondientes respuestas, así como los paquetes RTP con el contenido y la lista con los procesos sospechosos. También se encarga de mostrar en el reproductor las imágenes del contenido, y a través de una instancia de la clase *Vigilante* realiza el control de los procesos.
- **Reproductor.java:** es la parte visual del lado del cliente. En ella se muestra el contenido, aunque el encargado de mostrarlo sea *Controlador*. Al separar la interfaz de las acciones, cuando se pulsan los botones de *Reproductor*, se realiza una llamada a *Controlador* para que realice las acciones oportunas.
- **Vigilante.java:** esta clase realiza a través de su función *vigilar* el control de procesos en ejecución en el equipo del usuario. Además realiza comprobaciones sobre el sistema operativo para escoger qué comando se ejecuta para listar los procesos. Para la comprobación de los procesos recibe como parámetro el listado de los procesos sospechosos que *Controlador* ha recibido previamente.
- **RTPpacket.java:** como anteriormente hemos mencionado, esta clase se utiliza tanto en el lado del cliente como en el lado del servidor. En el lado del cliente se encarga de obtener la carga útil, el tamaño y el flujo de bits de los paquetes RTP.

Capítulo 7

Pruebas

Con la realización de pruebas buscamos detectar la mayor cantidad de defectos del sistema que estamos probando. Los defectos en el software pueden ser muy costosos, tanto en tiempo de trabajo como en coste económico, y cuanto más tarde se detecten estos defectos, mayor es su coste.

A lo largo de este proyecto se han realizado múltiples pruebas, algunas de ellas durante el estudio realizado en el capítulo 2 para ver el funcionamiento de determinadas tecnologías y poder escoger la más adecuada. Una vez escogida la tecnología a implementar se han ido realizando pruebas durante su desarrollo. La mayoría de esas pruebas se han repetido con el software finalizado, y son esas pruebas y sus correspondientes resultados lo que se documenta en este capítulo.

7.1. Plan de pruebas

- **Pruebas de rendimiento:** comprobamos la cantidad de tiempo y recursos que emplea nuestro sistema para la realización de los principales casos de uso. Para considerar adecuada la cantidad de recursos consumidos por nuestro sistema, tendrá que cumplir con los requisitos fijados en el capítulo 4.
- **Pruebas de funcionalidad:** se trata de comprobar que nuestro sistema cumple con los requisitos exigidos, que su funcionalidad es buena. Para estas pruebas nos basaremos en los casos de uso, para comprobar el funcionamiento del sistema durante la realización de sus principales acciones. Con el fin de crear situaciones límite para poner a prueba nuestro sistema y encontrar el mayor número de defectos posibles, realizaremos estas pruebas tanto con datos válidos como con datos no válidos.
- **Pruebas de interfaz de usuario:** comprobamos si la interfaz cumple con su cometido y su diseño es intuitivo. Por la simplicidad de la interfaz de nuestro sistema, estas pruebas serán las menos numerosas.

7.2. Casos de prueba

Caso de prueba 1	Conexión con el servidor I
Descripción	Con la aplicación correctamente instalada y el servidor arrancado y libre, se intenta acceder al contenido.
Acción	Se inicia el reproductor y se intenta acceder al contenido.
Resultado esperado	El reproductor está listo para mostrar el contenido y no se ha mostrado ningún mensaje de error.
Resultado obtenido	OK

Tabla 7.1: Caso de prueba 1

Se cumplen los requisitos funcionales 3 y 7. Al acceder al contenido, el servidor ya ha realizado el envío de los procesos sospechosos y toda la comunicación se ha realizado con opacidad sin que el usuario sea consciente de los protocolos subyacentes.

Caso de prueba 2	Conexión con el servidor II
Descripción	Con la aplicación correctamente instalada y el servidor sin iniciar, se intenta acceder al contenido.
Acción	Se inicia el reproductor y se intenta acceder al contenido.
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error <i>El servidor no está disponible</i> .
Resultado obtenido	OK

Tabla 7.2: Caso de prueba 2

Este caso de prueba no se corresponde con ningún requisito funcional, pero es una situación que puede darse con facilidad en nuestro sistema y es mejor tenerla prevista.

Caso de prueba 3	Conexión con el servidor III
Descripción	Con la aplicación correctamente instalada y el servidor arrancado y ocupado, se intenta acceder al contenido.
Acción	Se inicia el reproductor y se intenta acceder al contenido.
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error <i>El servidor no está disponible</i> .
Resultado obtenido	OK

Tabla 7.3: Caso de prueba 3

Se cumple el requisito funcional 9, ya que solo se permite que un cliente esté conectado con el servidor. Aunque se muestre un mensaje de error al cliente que se intenta conectar, el cliente que ya está conectado no sufre ninguna consecuencia.

Caso de prueba 4	Conexión con el servidor IV
Descripción	Con la aplicación correctamente instalada y el servidor arrancado y ocupado, se intenta acceder al contenido, pero indicando un puerto no válido.
Acción	Se inicia el reproductor y se intenta acceder al contenido.
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error <i>El puerto no está disponible.</i>
Resultado obtenido	OK

Tabla 7.4: Caso de prueba 4

El usuario no tiene la opción de modificar el puerto del servidor al que se intenta conectar, pero hemos realizado esta prueba para añadir robustez a nuestro sistema.

Caso de prueba 5	Conexión con el servidor V
Descripción	Con la aplicación correctamente instalada y el servidor arrancado y ocupado, se intenta acceder al contenido, pero indicando una dirección IP no válida.
Acción	Se inicia el reproductor y se intenta acceder al contenido.
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error.
Resultado obtenido	OK

Tabla 7.5: Caso de prueba 5

El usuario no tiene la opción de modificar la dirección IP a la que se intenta conectar, pero hemos realizado esta prueba para añadir robustez a nuestro sistema.

Caso de prueba 6	Reproducción del contenido I
Descripción	Después de acceder correctamente al contenido, se intenta reproducir el contenido.
Acción	Se accede al contenido y se escoge la opción <i>Play</i> .
Resultado esperado	Se inicia la reproducción del contenido sin incidentes.
Resultado obtenido	OK

Tabla 7.6: Caso de prueba 6

Caso de prueba 7	Reproducción del contenido II
Descripción	Después de acceder correctamente al contenido, se intenta reproducir el mismo, pero indicando un nombre de archivo no válido.
Acción	Se accede al contenido y se escoge la opción <i>Play</i> .
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error. En el lado del servidor se muestra el mensaje de error <i>Archivo no encontrado</i> .
Resultado obtenido	OK

Tabla 7.7: Caso de prueba 7

Caso de prueba 8	Reproducción del contenido III
Descripción	Después de acceder correctamente al contenido, se intenta reproducir el mismo, pero indicando un nombre de archivo válido, pero con un formato no válido.
Acción	Se accede al contenido y se escoge la opción <i>Play</i> .
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error. En el lado del servidor también se muestra el mensaje de error.
Resultado obtenido	OK

Tabla 7.8: Caso de prueba 8

Caso de prueba 9	Reproducción del contenido IV
Descripción	Después de acceder correctamente al contenido, se intenta reproducir el mismo, pero después de acceder al contenido se ha cerrado el servidor.
Acción	Se accede al contenido y se escoge la opción <i>Play</i> .
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error.
Resultado obtenido	OK

Tabla 7.9: Caso de prueba 9

Caso de prueba 10	Pausa del contenido I
Descripción	Después de acceder correctamente al contenido y de iniciar su reproducción, se intenta pausar el contenido.
Acción	Se accede al contenido, se reproduce y se escoge la opción <i>Pause</i> .
Resultado esperado	Se pausa la reproducción del contenido sin incidentes.
Resultado obtenido	OK

Tabla 7.10: Caso de prueba 10

Caso de prueba 11	Pausa del contenido II
Descripción	Después de acceder correctamente al contenido y de iniciar su reproducción, se intenta pausar el contenido, pero después de iniciar la reproducción del contenido se ha cerrado el servidor.
Acción	Se accede al contenido, se reproduce y se escoge la opción <i>Pause</i> .
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error.
Resultado obtenido	OK

Tabla 7.11: Caso de prueba 11

Caso de prueba 12	Salida del sistema
Descripción	Después de acceder correctamente al contenido, se intenta salir del sistema.
Acción	Se accede al contenido y se escoge la opción <i>Exit</i> .
Resultado esperado	Se cierra el reproductor y se muestra el mensaje de error en el servidor.
Resultado obtenido	OK

Tabla 7.12: Caso de prueba 12

Caso de prueba 13	Intento de copia I
Descripción	Después de acceder correctamente al contenido y de iniciar su reproducción, se abre un proceso de la lista de procesos sospechosos.
Acción	Se accede al contenido, se escoge la opción <i>Play</i> y se inicia un proceso sospechoso.
Resultado esperado	Cuando se realice la comprobación de los procesos (cada 5 segundos) se pausa el contenido.
Resultado obtenido	OK

Tabla 7.13: Caso de prueba 13

Caso de prueba 14	Intento de copia II
Descripción	Después de acceder correctamente al contenido y de iniciar su reproducción, se pausa el contenido y se abre un proceso de la lista de procesos sospechosos.
Acción	Se accede al contenido, se inicia su reproducción, se pausa el contenido, se inicia un proceso sospechoso y se escoge la opción <i>Play</i> .
Resultado esperado	Mientras el proceso sospechoso siga en ejecución no se permite la reanudación de la reproducción del contenido.
Resultado obtenido	OK

Tabla 7.14: Caso de prueba 14

Capítulo 8

Manual de usuario e instalación

En este manual de usuario explicamos de forma breve cómo obtener y cómo utilizar la parte del cliente de nuestro sistema. Damos por supuesto que el usuario no tiene por qué instalar ni utilizar el servidor. Aunque tanto en los ejecutables como en el código fuente proporcionados, el cliente se conecta con un servidor que está en el mismo equipo *localhost*, por lo que para probar el sistema es necesario ejecutar tanto el servidor como el cliente proporcionados en *Ejecutables*. En los anexos se detalla el contenido proporcionado. También se incluye en el contenido una sencilla página web que permite la descarga del reproductor tanto para Windows como para Linux.

Durante este manual de usuario daremos por supuesto que el cliente se conecta con un servidor remoto del que el usuario no necesita tener conocimiento para la ejecución del cliente. Para poder ejecutar nuestra aplicación es necesario que el usuario tenga instalado Java 8. Se puede descargar e instalar fácilmente desde la página de Java.

8.1. Descarga e inicio del cliente

Se accede a la página web de descarga del cliente RTSP. Una vez dentro de la web se selecciona a través de los links si se descarga el cliente para Windows o para Linux, y se obtiene el .zip correspondiente.

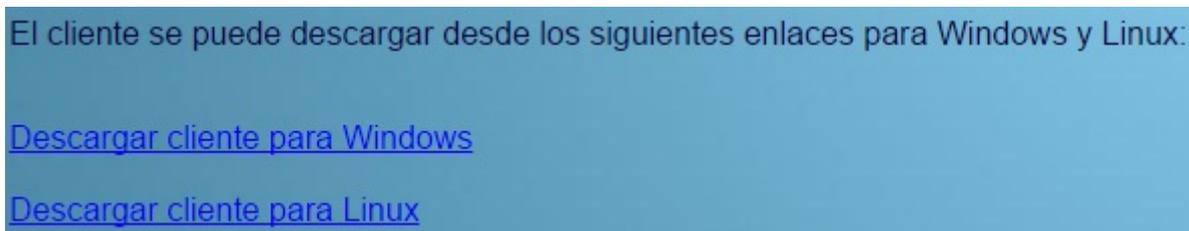


Figura 8.1: Enlaces de descarga del cliente RTSP.

Una vez descargado el .zip se descomprime en la carpeta que desee el usuario (no es necesario realizar ningún tipo de instalación). Con la carpeta descomprimida se deben realizar estas acciones para ejecutar el cliente dependiendo de si el sistema operativo es Windows o es Linux:

- **Windows:** solo es necesario hacer doble click sobre el archivo *Reproductor.exe* para iniciar la ejecución del cliente RTSP.
- **Linux:** se abre un terminal, se accede a la carpeta descomprimida del .zip utilizando el comando *cd* y la correspondiente dirección de la carpeta. Para iniciar la ejecución del cliente RTSP se introduce el siguiente comando *sh reproductor.sh*.

Si cuando se inicia el cliente, el servidor no se ha conectado o está ocupado con otro cliente se mostrará el siguiente mensaje de error y se cierra el cliente:



Figura 8.2: Mensaje de error: Servidor no disponible.

En el siguiente apartado se muestra cómo utilizar el cliente una vez iniciado.

8.2. Acceso al contenido

Si se ha iniciado correctamente el cliente y el servidor se encuentra disponible, el sistema nos mostrará la siguiente pantalla:

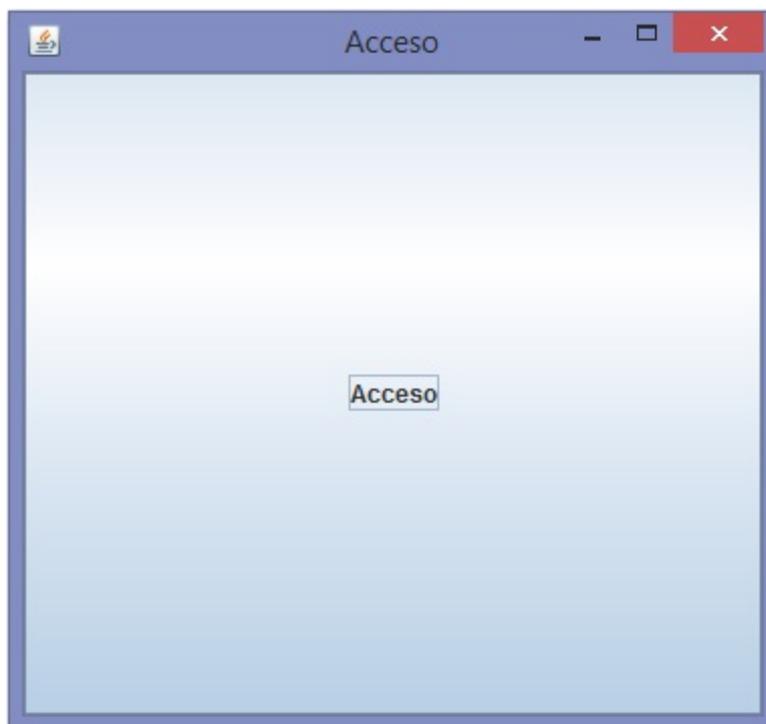


Figura 8.3: Pantalla de acceso.

Se pulsa el botón *Acceso* y pasa a la pantalla de reproducción del contenido. En el siguiente apartado se muestra cómo interactuar con el contenido.

8.3. Reproducción del contenido y controles

Una vez que se ha accedido correctamente al contenido, el sistema nos mostrará la siguiente pantalla:



Figura 8.4: Pantalla de reproducción del contenido.

Para iniciar la reproducción del contenido se pulsa *Play* y comienza la transmisión del mismo. En la siguiente imagen mostramos la apariencia del cliente durante la reproducción:

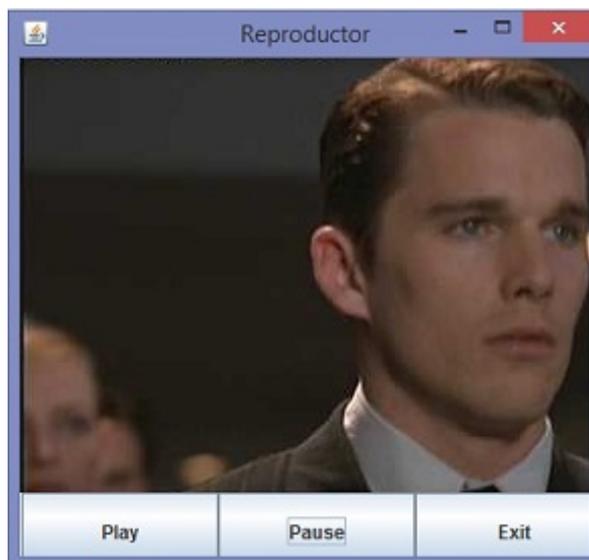


Figura 8.5: Cliente durante la reproducción del contenido.

Una vez iniciada la reproducción se puede detener la misma con *Pause*, y con el contenido pausado se puede reanudar la reproducción con *Play*. Si en algún momento el servidor tiene problemas se cierra el cliente y se muestra un mensaje de error.

En cualquier momento durante la ejecución del cliente, se puede cerrar la aplicación, tanto con *Exit* como con la forma habitual de cerrar una ventana. Pero es preferible hacerlo a través de *Exit*, ya que realiza un cierre más ordenado del sistema.

Capítulo 9

Conclusiones y trabajo futuro

9.1. Consecución de objetivos

- Se han mostrado los objetivos y motivaciones que nos han llevado a la realización de este TFG y a la implementación del cliente y el servidor RTSP.
- Se ha realizado un estudio sobre las tecnologías de reproducción, descarga, copia y cifrado de vídeo a través de la red, así como de otras tecnologías relacionadas con el vídeo a través de la red, como son los navegadores, los plugins y el desarrollo de los mismos. Además se han realizado pruebas para ver el funcionamiento y las prestaciones de estas tecnologías antes de escoger la tecnología a desarrollar.
- Se ha realizado el análisis y el diseño de la aplicación a realizar.
- Se ha implementado la comunicación entre un cliente y un servidor a través de los protocolos RTP y RTSP.
- Se ha realizado la comprobación de los procesos en ejecución en el equipo del usuario a través del listado de estos procesos utilizando el comando correspondiente en función del sistema operativo.
- Se ha conseguido el envío de información adicional a la de los protocolos desde el servidor.
- Se ha creado una interfaz sencilla y se ha separado la lógica de negocio.
- Se han integrado las aplicaciones previas para conseguir el Reproductor y el Servidor definitivos.
- Se han realizado pruebas para comprobar el correcto funcionamiento de nuestro sistema.
- Se ha documentado todo el proceso de realización de este TFG.

9.2. Valoración

Durante el desarrollo de este Trabajo de Fin de Grado se ha buscado aplicar los conocimientos y aptitudes adquiridos durante la realización de este Grado en Ingeniería Informática. Además, a lo largo de este TFG se han ampliado los conocimientos relacionados con:

- Navegadores y plugins para navegadores.
- Tecnologías de vídeo.
- Protocolos de comunicación.
- El lenguaje Java.
- Planificación y gestión de proyectos.

9.3. Trabajos futuros

Aunque nuestro sistema cumple con los objetivos propuestos, vamos a enumerar las posibles mejoras para nuestro sistema:

- Incluir la reproducción de archivos con audio.
- Permitir la reproducción de archivos en múltiples formatos.
- Implementar en el reproductor controles para poder ver el contenido en pantalla completa, así como retroceder y avanzar en la reproducción del contenido.
- Permitir que múltiples clientes se conecten al servidor RTSP.

Apéndice A

Anexos

A.1. Creación de los archivos ejecutables

Para la obtención de los archivos ejecutables, se han seguido una serie de pasos. Para ambos ejecutables se ha seguido el mismo proceso, así que mostraremos como ejemplo los pasos seguidos para crear el ejecutable de *Reproductor*:

- En primer lugar, se han compilado todos los archivos fuente .java. *java archivo.java*.
- Se ejecuta *jar -cf Reproductor.jar Reproductor.class*.
- Se ejecuta *jar cmf temp.mf Reproductor.jar *.class*. El archivo *temp.mf* es un archivo de manifiesto que indica la clase principal del archivo .jar. Ya tenemos generado el archivo .jar [61].
- Con el programa *Jar2Exe* se genera un .exe para Windows [62].
- En el caso de Linux, para tener un ejecutable creamos el archivo *reproductor.sh*, con el que ejecutamos el comando *sudo java -jar Reproductor.jar*.

A.2. Contenido del DVD

En el DVD adjunto se incluye toda la información del proyecto. A continuación detallaremos el contenido y la organización de dicho DVD. La estructura del DVD se divide en cinco carpetas con las distintas partes del proyecto:

- **Ejecutables:** incluye los archivos ejecutables tanto del *Reproductor* como del *Servidor*, para Windows y Linux, así como el contenido a reproducir en la carpeta del servidor, y los ejecutables *tasklist* en la carpeta del reproductor.
- **Fuentes:** contiene los archivos de código fuente del *Reproductor* y del *Servidor*.
- **Memoria:** incluye esta memoria en formato pdf y las imágenes que se muestran en la misma.

- **Otros:** contiene el archivo .asta con los diagramas realizados para el proyecto y el documento de Microsoft Project para la planificación del mismo.
- **Web:** incluye el archivo .html de la web de descarga, el .css de estilo y los .zip de descarga que contienen los ejecutables.

Bibliografía

- [1] Organización Mundial de la Propiedad Intelectual. Derecho de Autor. <http://www.wipo.int/copyright/es/>, 20 enero de 2015.
- [2] w3schools. Browser statistics. http://www.w3schools.com/browsers/browsers_stats.asp, 4 mayo de 2015.
- [3] Mozilla Foundation. History of the Mozilla Project. <https://www.mozilla.org/en-US/about/history/details/>, 9 febrero de 2015.
- [4] Mozilla Foundation. Get started with Firefox - An overview of the main features. <https://support.mozilla.org/en-US/kb/get-started-firefox-overview-main-features>, 9 febrero de 2015.
- [5] Mozilla Foundation. Firefox for Desktop. <https://www.mozilla.org/en-US/firefox/desktop/>, 11 febrero de 2015.
- [6] Mozilla Wiki. Telemetry. <https://wiki.mozilla.org/Telemetry>, 10 febrero de 2015.
- [7] Mozilla Corporation. Scratchpad. <https://developer.mozilla.org/en-US/docs/Tools/Scratchpad>, 10 febrero de 2015.
- [8] Mozilla Foundation. Mozilla Foundation License Policy. <https://www.mozilla.org/MPL/license-policy.html>, 11 febrero de 2015.
- [9] Mozilla Corporation. Plugin Basics. https://developer.mozilla.org/en-US/Add-ons/Plugins/Gecko_Plugin_API_Reference/Plug-in_Basics, 12 febrero de 2015.
- [10] Mozilla Corporation. Plugin Development Overview. https://developer.mozilla.org/en-US/Add-ons/Plugins/Gecko_Plugin_API_Reference/Plug-in_Development_Overview, 13 febrero de 2015.
- [11] Mozilla Corporation. Gecko. <https://developer.mozilla.org/en-US/docs/Mozilla/Gecko>, 13 febrero de 2015.
- [12] Mozilla Corporation. Plugins. <https://developer.mozilla.org/en-US/Add-ons/Plugins>, 12 febrero de 2015.

- [13] Mozilla Corporation. Streams. https://developer.mozilla.org/en-US/Add-ons/Plugins/Gecko_Plugin_API_Reference/Streams#Receiving_a_Stream, 13 febrero de 2015.
- [14] Mozilla Wiki. NPAPI. <https://wiki.mozilla.org/NPAPI>, 14 febrero de 2015.
- [15] Mozilla Corporation. Scripting plugins. https://developer.mozilla.org/en-US/Add-ons/Plugins/Gecko_Plugin_API_Reference/Scripting_plugins, 14 febrero de 2015.
- [16] StateTech. A Visual History of Internet Explorer. <http://www.statetechmagazine.com/misc/2013/08/visual-history-internet-explorer>, 15 febrero de 2015.
- [17] Microsoft Developer Network. Internet Explorer Architecture. <https://msdn.microsoft.com/en-us/library/aa741312.aspx>, 16 febrero de 2015.
- [18] Internet Explorer Dev Center. Internet Security and Privacy. [https://msdn.microsoft.com/en-us/library/ie/hh801956\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ie/hh801956(v=vs.85).aspx), 17 febrero de 2015.
- [19] Secunia. Secunia Vulnerability Review 2014. https://secunia.com/?action=fetch&filename=secunia_vulnerability_review_2014.pdf, 16 febrero de 2015.
- [20] Mozilla Corporation. ActiveX Control for Hosting Netscape Plugins in IE. https://developer.mozilla.org/en-US/docs/ActiveX_Control_for_Hosting_Netscape_Plug-ins_in_IE, 17 febrero de 2015.
- [21] StatCounter. Top 5 desktop, tablet and console browsers. <http://gs.statcounter.com/#browser-ww-monthly-201404-201504-bar>, 4 mayo de 2015.
- [22] Hector Russo. La historia del navegador Google Chrome. <http://geeksroom.com/2012/07/la-historia-del-navegador-google-chrome-infografia/64392/>, 18 febrero de 2015.
- [23] Chrome Developer. Welcome to Native Client. <https://developer.chrome.com/native-client>, 19 febrero de 2015.
- [24] Chrome Help. Plugins. <https://developer.chrome.com/native-client>, 21 febrero de 2015.
- [25] Mozilla Corporation. Html5. <https://developer.mozilla.org/es/docs/HTML/HTML5>, 23 febrero de 2015.
- [26] Libros Web. Breve historia de HTML. http://librosweb.es/libro/xhtml/capitulo_1/breve_historia_de_html.html, 24 febrero de 2015.
- [27] Genbeta. Historia de HTML. <http://www.genbetadev.com/desarrollo-web/historia-de-html-un-lenguaje-de-marca-que-ha-marcado-historia>, 24 febrero de 2015.

- [28] Flash Magazine. The Flash history. https://http://www.flashmagazine.com/news/detail/the_flash_history/, 25 febrero de 2015.
- [29] Flash Player Help. Archived Flash Player versions. <https://helpx.adobe.com/flash-player/kb/archived-flash-player-versions.html>, 26 febrero de 2015.
- [30] Gnash Project. Gnash. <http://www.gnashdev.org/?q=node/78>, 27 febrero de 2015.
- [31] Genbeta. GNU Gnash. <http://www.genbeta.com/imagen-digital/gnu-gnash-flash-player-0-8-10-liberado>, 27 febrero de 2015.
- [32] Lightspark. A free, open source Flash player. <http://lightspark.github.io/>, 28 febrero de 2015.
- [33] freedesktop. Swfdec. <http://swfdec.freedesktop.org/wiki/>, 1 marzo de 2015.
- [34] Fedora Project. Features Swfdec. <http://fedoraproject.org/wiki/Features/Swfdec>, 1 marzo de 2015.
- [35] Techcrunch. Mozilla HTML5 based Flash Player. <http://fedoraproject.org/wiki/Features/Swfdec>, 4 marzo de 2015.
- [36] VideoLAN. VLC media player Web plugin. https://wiki.videolan.org/Web_plugin/, 8 marzo de 2015.
- [37] ColonelPanic. Firebreath ready for testing. <http://colonelpanic.net/2009/12/firebreath-ready-for-testing/>, 12 marzo de 2015.
- [38] Richard Bateman. Firebreath. <http://www.firebreath.org/display/documentation/FireBreath+Home>, 11 marzo de 2015.
- [39] JUCE. Getting Started. <http://www.juce.com/learn/getting-started>, 13 marzo de 2015.
- [40] JUCE. Media player plugin for playing video files. <http://www.juce.com/forum/topic/media-player-plugin-playing-video-files>, 13 marzo de 2015.
- [41] Qt. Developing Plugins. <http://doc.qt.digia.com/solutions/4/qtbrowserplugin/developingplugins.html>, 15 marzo de 2015.
- [42] Qt. Browser Plugin. <http://doc.qt.digia.com/solutions/4/qtbrowserplugin/>, 15 marzo de 2015.
- [43] Qt. Qt Netscape Plugin Extension. <http://doc.qt.digia.com/3.3/netscape-plugin.html>, 15 marzo de 2015.
- [44] Streaming media. What is streaming? <http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=74052>, 20 marzo de 2015.

- [45] Lingfen Sun. *Guide to Voice and Video over IP For Fixed and Mobile Networks*. Springer, 1 edition, 2013. ISBN: 978-1-4471-4905-7.
- [46] Ministerio de Educación. ¿Qué es el streaming? <http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0103.html>, 20 marzo de 2015.
- [47] Streaming media. What is a codec? <http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-a-Codec-74487.aspx>, 22 marzo de 2015.
- [48] Internet Engineering Task Force. Real time streaming protocol. <http://www.ietf.org/rfc/rfc2326.txt>, 17 julio de 2015.
- [49] Internet Engineering Task Force. Real Time Streaming Protocol 2.0. <https://tools.ietf.org/html/draft-ietf-mmusic-rfc2326bis-40>, 15 julio de 2015.
- [50] Internet Engineering Task Force. RTP. <http://www.ietf.org/rfc/rfc3550.txt>, 5 junio de 2015.
- [51] Emezeta. Programas gratis para capturar pantalla en vídeo. <http://www.emezeta.com/articulos/18-programas-gratis-para-capturar-pantalla-en-video>, 29 marzo de 2015.
- [52] Internet Engineering Task Force. HTTP Over TLS. <http://tools.ietf.org/html/rfc2818>, 3 abril de 2015.
- [53] Internet Engineering Task Force. The Transport Layer Security Protocol. <http://tools.ietf.org/html/rfc5246#ref-TLS1.1>, 5 abril de 2015.
- [54] Xataka. Autenticación en dos pasos. <http://www.xataka.com/otros/autenticacion-en-dos-pasos-que-es-como-funciona-y-por-que-deberias-activarla>, 11 abril de 2015.
- [55] VideoLan. Plugins Mozilla. <https://wiki.videolan.org/Plugins/Mozilla/>, 20 mayo de 2015.
- [56] UMBC. Streaming video with rtsp and rtp. <http://www.csee.umbc.edu/~pmundur/courses/CMSC691C/lab5-kurose-ross.html>, 11 agosto de 2015.
- [57] Ian Sommerville. *Ingeniería del Software*. Addison Wesley, 7 edition, 2005. ISBN: 978-84-7829-074-1.
- [58] Craig Larman. *UML y patrones : introducción al análisis y diseño orientado a objetos y al proceso unificado*. Prentice Hall, 2 edition, 2010. ISBN: 978-84-2053-438-1.
- [59] Oracle. Java Platform SE 8. <http://www.ietf.org/rfc/rfc2326.txt>, 20 agosto de 2015.

- [60] Bogdan Ciubotaru. *Advanced Network Programming Principles and Techniques: Network Application Programming with Java*. Springer, 1 edition, 2013. ISBN: 978-1-4471-5292-7.
- [61] Como crear un jar con java desde la consola en Windows. <http://gl-epn-programacion-ii.blogspot.com.es/2013/02/como-crear-un-jar-con-java-desde-la.html>, 21 agosto de 2015.
- [62] Convertir archivos ejecutables de Java a exe con jar2exe. <http://monillo007.blogspot.com/2014/01/convertir-archivos-ejecutables-de-java.html>, 22 agosto de 2015.