Índice

Índice de ilustraciones ........................................................................................................... 5

1. Introducción .................................................................................................................. 7
   1.1 Planteamiento inicial ......................................................................................... 7
   1.2 Motivaciones ................................................................................................... 8
   1.3 Objetivos ........................................................................................................ 8

2. Blu-ray .......................................................................................................................... 10
   2.1 Descripción ................................................................................................. 10
   2.2 Funcionamiento ......................................................................................... 10
   2.3 Capacidad de almacenaje y velocidad .................................................... 11
   2.4 Tecnología ................................................................................................. 12
   2.5 Códices ..................................................................................................... 13
   2.6 Protecciones anticopia ............................................................................. 14
   2.7 HD Ready, Full-HD y HDMI ........................................................................ 16
   2.8 Diferencias entre el Blu-ray, el HD DVD y el DVD .................................. 17
   2.9 Estructura de los discos Blu-ray ............................................................... 19
   2.10 BD-J ....................................................................................................... 21
   2.11 BD-Live: Acceso a red ........................................................................... 26

3. Análisis .......................................................................................................................... 27
   3.1 Herramientas y lenguaje de programación .................................................. 27
   3.2 Fases del proyecto ...................................................................................... 30
   3.3 Modelo conceptual de la aplicación ........................................................... 34
   3.4 Actores ....................................................................................................... 37
   3.5 Casos de uso y contratos del cliente ............................................................ 37
3.5.1 Caso de uso “reproducir video” .................................................. 37
3.5.2 Caso de uso “pausar video” ...................................................... 38
3.5.3 Caso de uso “quitar el pause al video” ....................................... 39
3.5.4 Caso de uso “parar la reproducción del video” ........................... 39
3.5.5 Caso de uso “elegir modo de audio” .......................................... 40
3.5.6 Caso de uso “visualizar galería fotográfica” ............................... 41
3.5.7 Caso de uso “visualizar imagen de la galería fotográfica” ........... 42
3.5.8 Caso de uso “petición de evento actualizado” ............................ 43
3.5.9 Caso de uso “comprobar si hay una versión de eventos actualizada” ............................................................... 44
3.5.10 Caso de uso “visualizar la carpeta de archivos descargados” .... 45
3.5.11 Caso de uso “visualizar un archivo de texto descargado” .......... 46
3.5.12 Caso de uso “visualizar una imagen descargada” ..................... 47
3.5.13 Caso de uso “conexión con el servidor” .................................... 48
3.5.14 Caso de uso “navegar entre directorios del servidor” ............... 49
3.5.15 Caso de uso “descargar archivo del servidor” .......................... 50
3.5.16 Caso de uso “cancelar la descarga del archivo” ....................... 51
3.5.17 Caso de uso “desconectar del servidor” .................................... 52
3.6 Estudio económico ........................................................................ 53

4. Diseño ............................................................................................... 55

4.1 Diagramas de colaboración .............................................................. 55

4.1.1 Caso de uso “reproducir video” ............................................... 55
4.1.2 Caso de uso “pausar video” ..................................................... 56
4.1.3 Caso de uso “quitar el pause al video” ...................................... 58
4.1.4 Caso de uso “parar la reproducción del video” ......................... 59
4.1.5 Caso de uso “elegir modo de audio” ................................. 61
4.1.6 Caso de uso “visualizar galería fotográfica” .......................... 62
4.1.7 Caso de uso “visualizar imagen de la galería fotográfica” .... 64
4.1.8 Caso de uso “petición de evento actualizado” ...................... 65
4.1.9 Caso de uso “comprobar si hay una versión de eventos actualizada” ................................................................. 66
4.1.10 Caso de uso “visualizar la carpeta de archivos descargados”. 68
4.1.11 Caso de uso “visualizar un archivo de texto descargado” ..... 69
4.1.12 Caso de uso “visualizar una imagen descargada” ............... 70
4.1.13 Caso de uso “conexión con el servidor” ............................. 71
4.1.14 Caso de uso “navegar entre directorios del servidor” ........... 73
4.1.15 Caso de uso “descargar archivo del servidor” ...................... 74
4.1.16 Caso de uso “cancelar la descarga del archivo” ................. 76
4.1.17 Caso de uso “desconectar del servidor” ............................... 77

4.2 Diseño gráfico ............................................................................. 79
4.2.1 Pantalla del menú inicial .......................................................... 79
4.2.2 Pantalla del menú reproducir película ..................................... 80
4.2.3 Pantalla del menú modo de audio ........................................... 81
4.2.4 Pantalla del menú extras ......................................................... 82
4.2.5 Pantalla de la galería fotográfica ............................................ 83
4.2.6 Pantalla de una imagen ampliada de la galería de imágenes ... 84
4.2.7 Pantalla de la sección de eventos .......................................... 85
4.2.8 Pantalla de la sección de archivos descargados ...................... 86
4.2.9 Pantalla para visualizar un archivo de texto descargado ....... 87
4.2.10 Pantalla para visualizar una imagen descargada .................. 88
Programación en red para Blu-ray

4.2.11 Pantalla del menú descarga de contenidos .............................. 89

4.2.12 Pantalla de descarga de un archivo ......................................... 90

4.3 Diagrama de clases ........................................................................ 91

5. Implementación ................................................................................ 96

5.1 Menú inicial ..................................................................................... 96

5.2 Opciones de reproducción de video ................................................ 98

5.3 Escoger el modo de audio ............................................................... 102

5.4 Menú extras ..................................................................................... 104

5.5 Galería de imágenes ........................................................................ 106

5.6 Eventos ............................................................................................ 109

5.7 Archivos descargados ..................................................................... 117

5.8 Descarga de contenidos ................................................................. 119

6. Juego de pruebas ............................................................................... 132

7. Tutorial .................................................................................................. 162

7.1 Aplicación cliente ........................................................................... 162

7.1.1 Requisitos .................................................................................. 162

7.1.2 Funcionamiento .......................................................................... 164

7.2 Aplicación servidor ........................................................................ 169

7.2.1 Requisitos .................................................................................. 169

7.2.2 Funcionamiento .......................................................................... 171

8. Conclusiones ..................................................................................... 174

8.1 Aprendizaje personal .................................................................... 177

8.2 Futuras mejoras ............................................................................. 177

9. Bibliografía ........................................................................................ 179

ANEXO: Código fuente de la aplicación
Índice de ilustraciones

Fig 1. Logotipo de Blu-ray ................................................................. 10
Fig 2. Disco de Blu-ray ........................................................................ 12
Fig 3. Logos de HD Ready y Full HD .................................................. 16
Fig 4. Conector HDMI ......................................................................... 17
Fig 5. Tabla de las diferencias entre Blu-ray, HD DVD y DVD ............. 17
Fig 6. Estructura de directorios de un disco BDMV .............................. 19
Fig 7. Arquitectura Java ME ................................................................ 22
Fig 8. Diagrama de recursos de un receptor de TV Digital .................. 23
Fig 9. Diagrama de estado de un Xlet .................................................. 25
Fig 10. IDE de Netbeans 6.0 ................................................................ 28
Fig 11. Simulador de Blu-ray WinDVD 8 ............................................. 29
Fig 12. Fases del proyecto .................................................................. 33
Fig 13. Diagrama de Gantt ................................................................. 34
Fig 14. Diagrama conceptual de la aplicación ....................................... 36
Fig 15. Diagrama de colaboración “reproducir video” ......................... 56
Fig 16. Diagrama de colaboración “pausar video” ............................... 57
Fig 17. Diagrama de colaboración “quitar el pause al video” ................. 59
Fig 18. Diagrama de colaboración “parar la reproducción del video” .... 60
Fig 19. Diagrama de colaboración “elegir modo de audio” .................. 62
Fig 20. Diagrama de colaboración “visualizar galería fotográfica” ......... 63
Fig 21. Diagrama de colaboración “visualizar imagen de la galería fotográfica” .... 64
Fig 22. Diagrama de colaboración “petición de evento actualizado” .... 66
Fig 23. Diagrama de colaboración “comprobar evento actualizado” .... 67
Fig 24. Diagrama de colaboración “visualizar la carpeta de archivos descargados” 69
Fig 25. Diagrama de colaboración “visualizar un archivo de texto descargado” ..... 70
Fig 26. Diagrama de colaboración “visualizar una imagen descargada” .... 71
Fig 27. Diagrama de colaboración “conexión con el servidor” ............ 72
Fig 28. Diagrama de colaboración “navegar entre los directorios del servidor” ...... 74
Fig 29. Diagrama de colaboración “ descargar archivo del servidor” ....... 75
Fig 30. Diagrama de colaboración “cancelar la descarga del archivo” .. 77
Fig 31. Diagrama de colaboración “ desconectar del servidor” ............... 78
Fig 32. Diseño del menú inicial................................................................. 79
Fig 33. Diseño del menú de opciones de reproducción............................. 80
Fig 34. Diseño del menú de los modos de audio ...................................... 81
Fig 35. Diseño del menú de extras .............................................................. 82
Fig 36. Diseño de la galería fotográfica ...................................................... 83
Fig 37. Diseño de la imagen seleccionada ampliada .................................. 84
Fig 38. Diseño de la sección de eventos ..................................................... 85
Fig 39. Diseño de la sección de descargas .................................................. 86
Fig 40. Diseño para visualizar un archivo de texto ................................... 87
Fig 41. Diseño para visualizar una imagen ............................................... 88
Fig 42. Diseño del menú descarga de contenidos ..................................... 89
Fig 43. Diseño de la pantalla de descarga de un archivo ......................... 90
Fig 44. Diagrama de clases de la aplicación base ..................................... 91
Fig 45. Diagrama de clases de la aplicación ............................................. 95
Fig 46. Consola de MS-DOS .................................................................... 171
Fig 47. Ejemplo de pantalla después de acceder al directorio de la aplicación... 172
Fig 48. Pantalla de ejecución de la aplicación .......................................... 173
Fig 49. Diagrama de Gantt final ................................................................. 175
1. Introducción

1.1 Planteamiento inicial

Actualmente tras la victoria del Blu-ray por parte de Sony en la guerra de formatos ante Toshiba y la salida de la nueva especificación de Blu-ray denominada BD-Live (Blu-ray Disc Live) o Profile 2.0, observé que había pocos discos de Blu-ray que implementaban esta nueva opción. Al llevar este nuevo Perfil 2.0 tan poco tiempo en funcionamiento, vi la necesidad de crear una aplicación que utilizase esas características y estuviera desarrollado como una de las opciones disponibles dentro de un menú realizado en el lenguaje Java. Esta especificación permite las descargas de contenidos adicionales como pueden ser documentos, imágenes o videos a través de Internet. Esto repercute en una gran mejora en lo que se refiere a interactividad con las películas.

A partir de ahí decidí que también debía añadir las funcionalidades típicas que dispone una película de Blu-ray. En este caso las funciones básicas para la visualización de video como pueden ser: iniciar su reproducción, poder pausarlo, resumirlo y detenerlo. También he pensado que se pueda cambiar el modo de audio en el que se puede escuchar la película. Además debe de tener un conjunto de extras relacionados con la película. Uno de ellos será una galería de imágenes. Otra opción debería ser poder visualizar un video con contenido extra de la película. También he pensado que debería de haber un extra que te permita visualizar noticias actualizadas relacionadas con la película que dispone el Blu-ray. Por último debe haber una función que permita poder visualizar una lista de los archivos descargados, además de poderlos visualizar esos archivos de texto e imágenes.

Una aplicación en Blu-ray consta de varios archivos en lenguaje Java y también de imágenes que puedan utilizarse en los distintos menús. Para realizar la auditoría de Blu-ray se utilizará la base de una aplicación con su conjunto de librerías, con las cuales podré realizar una interfaz gráfica personalizada y aplicar funcionalidades de las que dispone. Además se le añadirá la nueva especificación del nuevo Perfil 2.0 de Sony.
Para finalizar cabe destacar que gracias a esta aplicación conseguimos implementar en una auditoria de Blu-ray las características de la nueva especificación BD-Live. De esta manera proporcionamos una mejora al formato Blu-ray para que la interactividad de estas películas sea la máxima posible.

1.2 Motivaciones

La motivación principal a la hora de realizar este proyecto fue el hecho de poder utilizar una tecnología actual con mucho futuro después de haberse impuesto Sony en la guerra de formatos gracias a su Blu-ray al HD DVD de Toshiba. Esta tecnología revoluciona la interactividad que teníamos vista hasta hoy en día entre el usuario y el contenido del disco Blu-ray, además de la calidad de imagen que nos ofrece.

También me importaba mejorar el nivel de Java que tenía aprendido hasta este momento y aprender esta parte que nos ofrece su lenguaje como es la especificación BD-J. Esto me ayudará a mejorar mi experiencia y conocimientos como programador para estar preparado en el futuro para ser más competitivo en el mundo laboral.

Para acabar tenía como un gran reto poder utilizar las características de la nueva especificación del BD-Live en mi aplicación. Esto es debido a que actualmente hay pocos discos de Blu-ray que lo incorporan como una opción más para el usuario. Esto se traduce en más interactividad.

1.3 Objetivos

Los objetivos que se quieren conseguir tras la realización de este proyecto son:

1. Utilizar las características del profile 2.0 para mi auditoria de Blu-ray.

2. Ofrecer a los usuarios una interfaz gráfica intuitiva para facilitar el poder moverse a través de los distintos menús que nos ofrece una película de Blu-ray.
3. Conseguir que el código implementado en la aplicación quede bien estructurado para que en un futuro cualquier programador pueda añadir mejoras con mucha facilidad.

4. El código de la aplicación debe funcionar en cualquier emulador o reproductor de Blu-ray de ordenador.

5. Mejorar en el uso del lenguaje Java, necesario para realizar cualquier aplicación para una película de Blu-ray.

6. Poner en práctica todos los conocimientos adquiridos de las diferentes asignaturas realizadas en la carrera para realizar la especificación, diseño e implementación de un proyecto.
2. Blu-ray

2.1 Descripción

Blu-ray [1] (también conocido como Blu-ray Disc o BD) es un formato de disco óptico de nueva generación de 12 cm de diámetro (igual que el CD y el DVD) para vídeo de alta definición y almacenamiento de datos de alta densidad. Su capacidad de almacenamiento actualmente llega a 50 GB a doble capa y a 25 GB a una capa. La Playstation 3 puede leer discos de hasta doble capa.

![Logo de Blu-ray](blu-ray-disc.png)

Fig 1. Logotipo de Blu-ray

Este formato se impuso a su competidor, el HD DVD, en la guerra de formatos iniciada para convertirse en el estándar sucesor del DVD, como en su día ocurrió entre el VHS y el Betamax, o el fonógrafo y el gramófono. Después de la caída de muchos apoyos de HD-DVD, Toshiba decidió abandonar la fabricación de reproductores y las investigaciones para mejorar su formato.

2.2 Funcionamiento

El disco Blu-ray hace uso de un rayo láser de color azul con una longitud de onda corta de 405 nanómetros, a diferencia del DVD, el cual usa un láser de color rojo de 650 nanómetros. Esto, junto con otros avances tecnológicos, permite almacenar sustancialmente más información que el DVD en un disco de las mismas dimensiones y aspecto externo. Blu-ray obtiene su nombre del color azul del rayo láser (blue ray quiere decir "rayo azul"). La letra "e" de la palabra original "blue" fue eliminada debido que, en algunos países, no se puede registrar para un nombre comercial una palabra común.
Blu-ray fue desarrollado en conjunto por un grupo de compañías tecnológicas llamado Blu-Ray Disc Association (BDA), liderado por Sony y Philips y formado por las siguientes empresas:

Estudios en exclusiva:

Sony Pictures Entertainment (Columbia Pictures y TriStar Pictures, entre otros),
Buena Vista (Walt Disney Pictures, Touchstone Pictures, Hollywood Pictures y Miramax, entre otros)
20th Century Fox (incluye el catálogo de Metro-Goldwyn-Mayer y United Artists)
Lions Gate Films
Warner Bros Pictures
New Line Cinema

Estudios colaboradores:

Studio Canal.
Paramount Pictures (sólo para los films dirigidos por Steven Spielberg).
Filmax (sólo en España).
Mar Studio (sólo en España).

El DVD ofreció en su momento una alta calidad ya que era capaz de dar una resolución de 720x480 (NTSC) o 720x576 (PAL) lo que es ampliamente superado por la capacidad de alta definición ofrecida por el Blu-ray que es de 1920x1080 (1080p). Este último es el formato utilizado por los estudios para archivar sus producciones, que anteriormente se convertía al formato que se quisiese exportar. Esto ya no será necesario, con lo que la industria del cine no tendrá que gastar esfuerzo y tiempo en el cambio de resolución de películas a Blu-ray, lo que abaratará sus costes.

2.3 Capacidad de almacenaje y velocidad

Una capa de disco Blu-ray puede contener alrededor de 25 GB o cerca de 6 horas de video de alta definición más audio, y el disco de doble capa puede contener
Programación en red para Blu-ray

aproximadamente 50 GB. La velocidad de transferencia de datos es de 36 Mbit/s (54 Mbps para BD-ROM), pero ya están en desarrollo prototipos a velocidad de transferencia 2x (el doble, 72 Mbit por segundo). Ya está disponible el BD-RE (formato reescribible) estándar, así como los formatos BD-R (grabable) y el BD-ROM, como parte de la versión 2.0 de las especificaciones del Blu-ray.

Fig 2. Disco de Blu-ray

El 19 de mayo de 2005, TDK anunció un prototipo de disco Blu-ray de cuatro capas de 100 GB. El 3 de octubre de 2007, Hitachi anunció que había desarrollado un prototipo de BD-ROM de 100 GB que, a diferencia de la versión de TDK y Panasonic, era compatible con los lectores disponibles en el mercado y solo requerían una actualización de firmware. Hitachi también comentó que está desarrollando una versión de 200GB.

2.4 Tecnología

El tamaño del "punto" mínimo en el cual un láser puede ser enfocado está limitado por la difracción, y depende de la longitud de onda del haz de luz y de la apertura numérica de la lente utilizada para enfocarlo. En el caso del láser azul-violeta utilizado en los discos Blu-ray, la longitud de onda es menor con respecto a tecnologías anteriores, aumentando por tanto apertura numérica (0,85, comparado con 0,6 para DVD). Con ello, y gracias a un sistema de lentes duales y a una cubierta protectora más delgada, el rayo láser puede enfocar de forma mucho más precisa en la superficie del disco. Dicho de otra forma, los puntos de información legibles en el disco son mucho más pequeños
y, por tanto, el mismo espacio puede contener mucha más información. Por último, además de las mejoras en la tecnología óptica, los discos Blu-Ray incorporan un sistema mejorado de codificación de datos que permite empaquetar aún más información.

El DVD tenía dos problemas que se intentaron resolver con la tecnología Blu-Ray, por ello la estructura es distinta. En primer lugar para la lectura en el DVD el láser debe atravesar la capa de policarbonato de 0,6 mm en la cual el láser se puede difractar en dos haces de luz. Si esta difracción es alta, por ejemplo si estuviera rayado, impide la lectura del disco. En el Blu-ray, al tener una capa de sólo 0,1 mm se evita este problema, ya que tiene menos recorrido hasta la capa de datos; además, esta capa es resistente a ralladuras. En segundo lugar si el disco estuviera inclinado, en el caso del DVD, por igual motivo que el anterior problema, la distorsión del rayo láser haría que leyese en una posición equivocada dando lugar a errores. Gracias a la cercanía de la lente y la rápida convergencia del láser la distorsión es inferior pudiéndose evitar el error de lectura.

Otra característica importante de los discos Blu-ray es su resistencia a las ralladuras y la suciedad. La fina separación entre la capa de lectura y la superficie del disco hacía los discos Blu-ray más propensos a ralladuras y suciedad que un DVD normal. Es por ello que se pensó primero en comercializarlos en una especie de carcasa. La idea fue desechada gracias a la elaboración por parte de TDK de un sustrato protector llamado Durabis que no solo compensa la fragilidad del Blu-ray sino que le otorga una protección extra contra las ralladuras. Existen también discos DVD con esta protección, pero no es tan necesaria como lo es en un Blu-ray debido al mayor espesor de la capa que separa los datos de la superficie del disco, 0,6 mm en comparación con los 0,1 mm del Blu-ray.

2.5 Códecs

Un códec es una especificación que utiliza un dispositivo o programa para desempeñar transformaciones bidireccionales sobre datos y señales. De esta manera se pueden interpretar los datos de un archivo y así poder ejecutarlo posteriormente en la aplicación que lo utilice.
El Blu-ray soporta los mismos sistemas de archivos que sus predecesores, como el UDF y el ISO 9660. Gracias a su capacidad de almacenamiento, el vídeo es de alta definición y audio de hasta 8 canales. Soporta los formatos de compresión MPEG-2, MPEG-4 y VC-1. Los formatos MPEG-4 AVC Y SMPTE VC-1 implementan algoritmos de compresión más avanzados que permiten ofrecer gran calidad (a un nivel similar).

El formato MPEG-2 se mantiene por retrocompatibilidad, es casi obsoleto (comparado con los otros) pero aun así se ha utilizado para codificar la mayor parte de las primeras películas que han salido en formato Blu-Ray. Esto es debido a que el MPEG-4 AVC tiene altos requerimientos, consumo de recursos y alto tiempo de codificación. 

El SMPTE VC-1 pertenece a un consorcio de compañías dominado por Microsoft. Esto hace que el grado de implantación del códec sea desigual favoreciendo al HD DVD. A pesar de esto, Microsoft ha hecho un esfuerzo notable realizando seminarios para las compañías del sector lo que ha aumentado la aceptación del VC-1.

Los códecs de compresión de audio soportados por Blu-ray son LPCM (sin compresión), DTS, Dolby Digital, Dolby True HD y DTS-HD Master Audio (hay más pero éstos son los principales). Los tres primeros se mantienen por retrocompatibilidad respecto al DVD al igual que los de vídeo. Los dos últimos representan una gran mejora ya que permiten la compresión sin pérdidas.

2.6 Protecciones anticopia

Los sistemas Blu-ray incorporan siete sistemas anticopia: AACS, BD+ y Rom Mark, SPDG, HGTT, ICT y DCT.

El AACS (Advanced Access Content System) es una mejora respecto al CSS del DVD. Su principal función es el control de la distribución de contenidos. Una de las consecuencias del AACS la de crear una lista negra de grabadores. Este sistema permite dar una clave para cada modelo de grabador. Esto facilita el seguimiento de qué claves son descifradas y qué grabadores permiten la copia de blu-rays, la consecuencia sería
revocar la clave y no incluirla en siguientes blu-rays garantizando la incompatibilidad con el grabador.

En un principio, la Asociación de Discos Blu-Ray decidió incorporar la restrictiva copia gestionada (MC). Inmediatamente, las compañías informáticas involucradas protestaron debido a su alta restricción. Al final decidieron que el control de distribución de contenidos sería copia gestionada obligatoria (MMC), usada en el HD DVD y que permite al menos una copia de un disco para enviarla a otros dispositivos. En esta decisión, influyó el hecho de que HD DVD lo hubiese adoptado ya que el usuario podría decantarse por un sistema menos restrictivo en este aspecto.

Los discos Blu-ray tienen en su estándar un sistema anticopia exclusivo denominado BD+. Este sistema permite cambiar dinámicamente las claves para la protección criptográfica de los BD originales. Si una de estas claves es descubierta, los fabricantes no tienen más que cambiar la clave, de forma que las nuevas unidades del producto no puedan ser pirateadas con dicha clave descubierta. A petición de HP, se añadió la posibilidad de que un usuario pueda comprar dichas claves para realizar un número limitado de copias del disco que ha comprado, quitando derechos de copia a los usuarios que utilizan este formato. El BD+ puede comprobar también si el hardware ha sido modificado e impedir la reproducción.

También se ha acordado que los BD lleven una marca de agua digital. Bajo el nombre de ROM-Mark, esta tecnología estará presente en todos los discos originales y requiere un componente especial de hardware licenciado en grabadores específicos para poder insertar la marca de agua durante la copia. Todos los lectores de Blu-ray deben buscar esa marca. De esta manera, la BDA pretende frenar la copia masiva de Blu-ray.

Otra incorporación es el SPDG (Self-Protecting Digital Content) que es un pequeño programa que se incluirá en los reproductores. Tiene un comportamiento similar al de un sistema operativo y evitará que los grabadores puedan grabar las películas que estén siendo visionadas. Esta también tiene su polémica, no sólo por el extremismo de la política anticopia, sino porque puede suponer una grave vulnerabilidad ya que los sistemas operativos son sensibles ante virus informáticos.
Por último está la inclusión del ICT (Image Constraint Token), que es una señal que evita que los contenidos de alta definición viajen en soportes no cifrados y, por tanto, susceptibles de ser copiados. En la práctica, lo que hace es limitar la salida de vídeo a la resolución de 960x540 si el cable que va del reproductor a la televisión es analógico, aunque la televisión soporte alta definición. El ICT no es obligatorio y cada compañía decide libremente si añadirlo o no a sus títulos. Por ejemplo, Warner está a favor de su uso mientras que Fox está en contra.

2.7 HD Ready, Full-HD y HDMI

Este apartado trata de remarcar la diferencia entre compatibilidad y adaptación y cómo sacar el máximo partido a la alta definición con Blu-ray. Esta información viene referenciada en [3]. Se dice por compatibilidad cualquier televisión que permita reproducir el contenido de discos Blu-ray o HD DVD. Ésta se logra con dispositivos que lleven el logotipo HD (High Definition) Ready o 1080i, lo cual no quiere decir que se le saque el máximo partido a la alta definición puesto que sólo garantiza que la resolución mínima sea de 720 líneas en panorámico (1280x720) con escaneado progresivo (si es en modo escalado garantiza 1920x1028). Si lo que queremos es alta definición con una resolución de 1920x1080 en modo progresivo deberán encontrarse aquellos dispositivos marcados bajo el logotipo de Full-HD o 1080p.

La adaptación significa una conversión que implica una pérdida en la señal de audio y/o vídeo. En esta última entra el HDMI (High-Definition Multi-media Interface), un conector que permite el envío de la señal nativa de alta definición (1920x1080, 24fps y escaneado progresivo) para garantizar la máxima fluidez y calidad de las escenas. Cualquier otra conexión, ya sea analógica o digital tendrá que adaptarse con las lógicas consecuencias. Cualquier conector y medio que se utilice para conectar el reproductor
con el dispositivo de visionado deberá ser compatible con HDCP (HDMI lo es) bajo pena de que convertirá la señal antes de su salida a calidad DVD.

2.8 Diferencias entre el Blu-ray, el HD DVD y el DVD

<table>
<thead>
<tr>
<th></th>
<th>Blu-ray</th>
<th>HD DVD</th>
<th>DVD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacidad</strong></td>
<td>23,3/25/27 GB (capa simple)</td>
<td>15 GB (capa simple)</td>
<td>4,7 GB (Capa Simple)</td>
</tr>
<tr>
<td></td>
<td>46,6/50/54 GB (capa doble)</td>
<td>30 GB (capa doble)</td>
<td>8,5GB (capa doble)</td>
</tr>
<tr>
<td><strong>Longitud de Onda del Rayo Láser</strong></td>
<td>405 nm</td>
<td>405 nm</td>
<td>650 nm</td>
</tr>
<tr>
<td><strong>Tasa de Transferencia datos</strong></td>
<td>36,0 / 54,0 Mbps</td>
<td>36,55 Mbps</td>
<td>11,1 / 10,1 Mbps</td>
</tr>
<tr>
<td><strong>Formatos Soportados</strong></td>
<td>MPEG-2, MPEG-4 AVC, VC-1</td>
<td>MPEG-2, VC-1 (Basado en WMV), H.264/MPEG-4 AVC</td>
<td>DVD, VCD, MPEG-2</td>
</tr>
<tr>
<td><strong>Resistencia a rayas y suciedad</strong></td>
<td>Sí</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Resolución máxima de vídeo soportada</strong></td>
<td>1080p</td>
<td>1080p</td>
<td>480p/576p</td>
</tr>
</tbody>
</table>

Fig 4. Conector HDMI

Fig 5. Tabla de las diferencias entre Blu-ray, HD DVD y DVD [3]
Se puede observar cómo la tecnología del Blu-ray permite una mayor tasa de transferencia para el vídeo. Esta diferencia no ha sido aprovechada en muchos casos debido a varios factores. En primer lugar la tasa es variable y depende de la compañía elegir cuál va usar, no necesariamente la máxima. Segundo, la tasa puede tener otras utilidades aparte de la transmisión del vídeo. Último, influye el códec utilizado, las compañías Blu-ray utilizaron en sus primeros lanzamientos el MPEG-2 lo que hizo que no solo la calidad sea inferior a lo esperado en una reproducción de alta definición, sino que estaba por debajo de su competidor el HD DVD que desde el principio codificó principalmente en VC-1.

La tasa de transferencia para audio también es más alta en Blu-ray que en HD DVD. En la tabla no se puede apreciar, pero para entender las diferencias entre el Blu-ray y HD DVD hay que darse cuenta de que mientras que la primera es una nueva tecnología que busca el aumento de prestaciones, el segundo quiere mejorar el soporte DVD sin aumentar considerablemente los costes. El HD DVD por tanto tiene a su favor que bastaron unas pocas modificaciones en los equipos de producción de DVD mientras que en Blu-ray ha sido necesaria la instalación de nuevos y caros equipos de producción.

En los precios el Blu-ray tiene mucho que perder ya que los costes son más altos, lo que se refleja por ejemplo en los lectores. Los lectores HD DVD cuestan en torno a la mitad que los Blu-ray, aunque ya en el 2007 esta diferencia bajó. Por el contrario las películas tienen un coste similar.

En cuanto a los títulos híbridos (DVD y alta definición) parece mejor el sistema Blu-ray que permite asignar cada formato a una capa sin cambiar de cara. En HD DVD se asigna por caras con lo que se hace necesario cambiarlo. Además el soporte híbrido esta limitado a dos formatos (dos caras) mientras que en Blu-ray se pueden tantos como capas haya (aunque por ahora no tiene sentido añadir otros formatos existe la posibilidad de hacerlo en el futuro).
2.9 Estructura de los discos Blu-ray

El formato de disco Blu-ray dedicado a las películas (Blu-ray Movie Disc o BDMV) ofrece dos modos diferentes de acceder al contenido del disco: HD Movie Mode (HDMV) y BD-Java (BD-J). En HDMV se define una estructura declarativa de navegación, mediante jerarquías de estructuras lógicas relacionadas entre ellas y permitiendo un conjunto básico de comandos. El modo BD-J se accede a través de HDMV y proporciona un nivel más sofisticado de programación en comparación a HDMV, ya que está basado en Java 2 Microedition.

Para ejecutar una aplicación en BD-J hay que crear la base HDMV y acceder mediante un objeto BDJO (BD-J Object) a la aplicación.

A continuación podremos ver la estructura de directorios de un disco BDMV:

Fig 6. Estructura de directorios de un disco BDMV
Index.bdmw → Es la tabla de índices de MovieObjects. Cada elemento de la tabla puede corresponder a un título de un Movie Object o a un objeto BDJO (es decir una aplicación BD-J). Los primeros elementos de la tabla son importantes: el First Playback que define el primer objeto que se reproduce al iniciarse la reproducción y el Top Menu que indica que objeto contiene el menú principal.

MovieObject.bdmw → Es la base de datos de los Movie Objects. Un Movie Object es una secuencia de comandos que puede contener una o más llamadas a una playlist.

Playlist → Se guardan los archivos correspondientes a los Playlist. Estos definen una línea en el tiempo en que se incluyen las diferentes pistas correspondientes a los streams de audio y de video. Cada uno de ellos está organizado secuencialmente sobre la línea del tiempo. Además incluye las marcas que indican donde se encuentran las diferentes escenas.

Clipinf → Se guarda la información sobre los clips y se corresponden con los ficheros de la carpeta Stream.

Stream → Se guardan los archivos que corresponden al contenido de audio y video.

Auxdata → Sonidos y otros archivos auxiliares.

Meta → Ficheros para la librería de discos.

Bdjo → Se guardan los archivos BD-J Object, cada uno de estos representa una aplicación BD-J y contiene información de cómo se ejecuta y con que permisos.

Jar → Fichero correspondiente a las aplicaciones BD-J.

Backup (BDMV) → Contiene copias de los ficheros index.bdmv, MovieObject.bdmv y las carpetas Playlist, Clipinf, Bdjo y Jar.
Certificate ➔ Contiene el identificador de BDMV y el certificado de autentificación.

Backup (Certificate) ➔ Contiene una copia de seguridad del identificador BDMV y del certificado.

2.10 BD-J

La inclusión de la plataforma Java en el estándar de las películas grabadas en Blu-ray fue muy novedosa. Esta plataforma, que ha recibido el nombre de BD-J (Blu-ray Disc Java), permite más libertad a los desarrolladores para incluir contenidos en la película. No es sólo la capacidad de desplegar un menú en cualquier momento de la reproducción, o la posibilidad de incrustar aplicaciones en el disco. También se proyecta la posibilidad de descargar contenidos adicionales desde Internet al disco. Por ejemplo, subtítulos en determinados idiomas, contenidos especiales, escenas cortadas entre otros.

La plataforma BD-J fue construida a partir de las especificaciones del GEM (Globally Executable Multimedia Home Platform). El GEM define las siguientes partes:

- Contenido formatos (audio, vídeo, imágenes).
- Transporte protocolos (emisión, Protocolo de Internet).
- Un modelo de aplicación.
- Un mecanismo para la aplicación de señalización.
- Una plataforma de aplicaciones basadas en Java.

El GEM al crearse cogió referencias del MHP. El MHP define una plataforma común para las aplicaciones interactivas de la televisión digital, independiente tanto del proveedor de servicios interactivos como del receptor de televisión utilizado.

El modelo tradicional de aplicaciones Java es bastante simple: una clase se carga, invoca su método main (), y espera hasta que todos los hilos terminen o se llame a System.exit (). Para muchas aplicaciones, este modelo permite muy poco control sobre el comportamiento de la aplicación. Esta es la razón por la que se define PBP (Personal
Programación en red para Blu-ray

Basis Profile), este permite que el dispositivo pueda iniciar, cargar, detener y pausar aplicaciones en respuesta a acontecimientos externos.

PBP está definido en J2ME (Java 2 Micro Edition). J2ME es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrollo de software para dispositivos con recursos restringidos. Permiten ejecutar programas de seguridad, conectividad y utilidades en tarjetas inteligentes, PDAs, sintonizadores de TV y otros pequeños electrodomésticos.

Como podemos ver en el siguiente diagrama, la arquitectura J2ME [5] se puede dividir en dos grandes bloques de arquitecturas que dependen del tipo de dispositivo y las características de los mismos:

![Fig 7. Arquitectura Java ME](image)

En la actualidad existen dos configuraciones J2ME:
- Connected Limited Device Configuration (CLDC)
- Connected Device Configuration (CDC)
En este caso se usa CDC ya que está diseñada para dispositivos que tienen más memoria, procesadores más rápidos y un ancho de banda mayor, como Set-top boxes, pasarelas residenciales, asistentes personales de gran capacidad, etc. Incluye una máquina virtual Java completa (Java Virtual Machine, JVM) y un subconjunto de APIs de la arquitectura J2SE mucho mayor. Se orienta a dispositivos con CPU de 32 bits y un mínimo de 2 MB de memoria disponible para la plataforma Java y aplicaciones asociadas.

Una aplicación BD-J también llamada Xlet [2] son aplicaciones en Java para entornos de televisión. No siguen el modelo convencional de las aplicaciones Java en el que cada aplicación tiene el control completo de su ciclo de vida y de los recursos de la máquina virtual sobre la que se ejecutan. Además varias aplicaciones pueden estar ejecutándose a la vez y no tienen por qué iniciarse a través de línea de comandos.

El interfaz Xlet permite una fuente externa (gestor de aplicaciones de un receptor de TV Digital) para comenzar y terminar las aplicaciones, al igual que controlarlas de otras maneras. El interfaz Xlet se encuentra en el paquete javax.tv.xlet.package:

```java
public interface Xlet {
    public void initXlet(XletContext ctx);
}
```

Fig 8. Diagrama de recursos de un receptor de TV Digital
throws XletStateChangeException;

public void startXlet()
throws XletStateChangeException;

public void pauseXlet();

public void destroyXlet(boolean unconditional)
throws XletStateChangeException;
}

Como podemos ver dispone de métodos que permiten al gestor de aplicaciones inicializarlos, arrancarlos y pararlos.

El más importante a destacar en la ejecución de un Xlet es que también puede ser pausado y resumido. La razón para ello es muy simple, en un entorno como un receptor de TV Digital, varias aplicaciones pueden estar ejecutándose al mismo tiempo, pero por restricciones de hardware, sólo una de ellas puede estar visible cada vez. En este escenario, las aplicaciones no visibles, deben ser pausadas para dejar libres los recursos para aquella que lo es.

Así, un Xlet tendrá cuatro estados principales – Cargado, pausado, arrancado y destruido. Si se examina el ciclo de vida de un Xlet se puede ver dónde encajan estos estados en el diseño:
El gestor de aplicaciones carga el archivo con la clase principal del Xlet (como se indicó desde el emisor) y crea una instancia del Xlet llamando al constructor por defecto. Esto puede pasar en cualquier momento una vez que la aplicación haya sido indicada. En este momento, el Xlet alcanza el estado Cargado (Loaded).

El Xlet puede ser iniciado de tres formas:

- Por indicación directa del usuario.
- Invocado por otra Xlet.
- Automático por configuración en la AIT (Tabla de información de las aplicaciones).

Entonces, el gestor de aplicaciones en el receptor llama al método initXlet(), pasando un nuevo objeto XletContext para el Xlet.

El Xlet puede usar este XletContext para inicializarse, y para precargar datos de gran tamaño como imágenes que pueden requerir tiempo para cargarse. Cuando la inicialización se ha completado, el Xlet está en el estado Pausa (Paused) y puede comenzar inmediatamente.

Una vez que el método initXlet() ha terminado, el gestor de aplicaciones llama el método startXlet(). Esto pasará el Xlet del estado Pausa al estado Arrancado (Started), y el Xlet podrá interactuar con el usuario.
Durante la ejecución del Xlet, el gestor de aplicaciones puede llamar al método pauseXlet(). Esto hará que la aplicación pase del estado Arrancado al Pausado. La aplicación puede volver a ejecutarse de nuevo llamando al método startXlet(). Esto puede pasar varias veces durante la vida del Xlet.

Al final de su vida, el gestor de aplicaciones llamará al método destroyXlet(), que hará que el Xlet pase al estado Destruido (Destroyed) y libere todos sus recursos. En ese punto, no se puede volver a ejecutar esa instancia del Xlet.

2.11 BD-Live: Acceso a red

Desde la creación del Blu-ray se han ido actualizando sus especificaciones. En este tiempo se han definido tres tipos distintos de perfiles para vídeo. El primero de ellos, llamado BD-Video o perfil 1.0 se limita a definir la capacidad de reproducción de vídeo, sin capacidades adicionales.

Posteriormente, se define el Bonus View o perfil 1.1, que añade nuevas capacidades a la reproducción. Este perfil exige a los reproductores disponer de 256 MB de memoria interna para almacenar datos, además de un decodificador de audio y vídeo adicionales.

Esto permite la reproducción de vídeo con Picture-in-Picture, es decir, poder mostrar dos imágenes al mismo tiempo, por lo que podemos ver, por ejemplo, comentarios del director mientras visualizamos la película.

Finalmente, llegamos al BD-Live [4] o perfil 2.0. En este se ha aumentado la memoria interna que ha de incluir el reproductor hasta 1 GB, además de exigir que este incluya una conexión a Internet, ya sea a través de Ethernet o de Wi-Fi.

La función BD-Live, permite la interactividad con contenidos descargables mediante conexión a internet. De esta forma, en los títulos Blu-ray que dispongan de esta característica, se podrá acceder a chats interactivos, extras exclusivos, material adicional como fondos de pantalla, galerías y incluso compras on-line de productos derivados de la película.
3. Análisis

3.1 Herramientas y lenguajes de programación

Uno de los motivos principales por los que elegí este proyecto es por el lenguaje de programación que tenía que utilizar para su implementación, que en este caso fue Java. Este es el lenguaje utilizado por los discos de Blu-ray ya que permite crear libremente contenido interactivo entre el usuario y el disco. Además al ser Java un lenguaje de programación con mucho futuro y querer enfocar mi carrera profesional en esa dirección, no dude un momento en escoger este proyecto.

- **Java**
  
  Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La filosofía de este lenguaje se base en:

  1. Uso de la metodología de la programación orientada a objetos.
  2. Permite la ejecución de un mismo programa en múltiples sistemas operativos.
  3. Incluye por defecto soporte para trabajo en red.
  4. Puede ejecutar código en sistemas remotos de forma segura.

Para realizar la aplicación no tuve opción de elegir otros lenguajes de programación para su realización. Esto es debido a que Sony decidió que las aplicaciones que se realizaran para sus películas en Blu-ray debían utilizar la especificación BD-J de la plataforma Java.
• **BD-J**

Es una especificación de apoyo a Java. Permite contenido adicional en los títulos de los discos de Blu-ray para ser mucho más sofisticados que los contenidos proporcionados por un DVD estándar. Incluye acceso a red y el acceso a la ampliación de capacidad de almacenamiento local. Todo esto hace permitir a los proveedores entregar un contenido con gran interactividad para los usuarios finales y fácilmente programables para los programadores.

• **Netbeans 6.0**

Es una plataforma para el desarrollo de aplicaciones de escritorio usando Java y es un entorno de desarrollo integrado (IDE). Es también una herramienta para programadores pensada para escribir, compiler, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Además es un producto libre y gratuito sin restricciones de uso. También soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles).

Fig 10. IDE de Netbeans 6.0
He probado también Eclipse (el otro famoso IDE para entorno Java) y no vi ninguna mejora sustancial sobre NetBeans, pero de NetBeans sobre Eclipse si he visto varias:

- Mejores herramientas para diseñar GUI's.
- Interfaz mucho más amigable e intuitiva (mejor usabilidad).
- Apoyo directo de SUN.
- La versión 6 consume igual o incluso menos memoria que Eclipse.

**InterVideo WinDVD 8**

Es un simulador de Blu-ray suministrado por Corel Intervideo de forma gratuita para empresas, para poder realizar las pruebas necesarias de reproducción de aplicaciones en Blu-ray sin la necesidad de disponer de unidades físicas de reproducción de discos de este tipo.

![Simulador de Blu-ray WinDVD 8](image)

**Fig 11. Simulador de Blu-ray WinDVD 8**

**Microsoft Word**

Es un procesador de textos utilizado para escribir la memoria del proyecto.
• **Adobe Photoshop**
  Es una aplicación que utilicé para la edición y retoque fotográfico de imágenes.

• **Microsoft Project**
  Es un software de administración de proyectos diseñado y desarrollado para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo. Esta aplicación la utilicé para planificar las fases del proyecto y poder realizar el diagrama de Gantt.

• **Microsoft Visio**
  Es un software de dibujo vectorial. Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, y más, que permiten iniciar al usuario en los lenguajes de programación. Esta aplicación la necesité para realizar los diagramas conceptuales, de componentes y de clases de la aplicación.

3.2 Fases del proyecto

1. **Estudio de la propuesta del proyecto**
   El día que estuve hablando con el que sería mi futuro tutor de proyecto, me gustó la idea que me propuso ya que me ayudaría a aprender a utilizar una tecnología que está en su auge después de hacerse Blu-ray con todo el mercado. Además al saber que el lenguaje de programación a utilizar era Java y como quiero encarar mi futura carrera profesional a utilizar ese lenguaje no dudé ningún momento en escoger este proyecto. Esta fase duró 7 días.

2. **Familiarización con el lenguaje y el entorno de programación**
   Durante esta etapa me dediqué a leer mucha teoría sobre el BD-J, ya que los conocimientos que tengo yo en Java son generales y esta parte no la había tocado nunca. Además me dediqué a hacer ejemplos prácticos para ir probando los conocimientos teóricos que fui adquiriendo y así también irme...
familiarizando con el entorno de programación que utilizaría que sería Netbeans 6.0. Esta fase duró 40 días.

3. **Análisis y especificación**
   Durante esta etapa se realiza el modelo conceptual. Esta fase duró 15 días.

4. **Diseño y casos de uso**
   En esta etapa se diseñaron las clases que formarían el proyecto y los casos de uso. Esta fase duró 24 días.

5. **Diseño de la interfaz gráfica**
   En esta etapa se dibujó sobre papel un esbozo de la apariencia visual que debería tener la interfaz gráfica en todos los menús. Esta fase duró 3 días.

6. **Implementación**
   a. **Interfaz gráfica**
      En esta etapa se implementó la interfaz gráfica de la aplicación y se prolonga durante toda la implementación. Esta fase tuvo una duración de 105 días.

   b. **Descarga de contenidos**
      En esta etapa se implementó el poder descargar contenidos adicionales de un servidor remoto como imágenes o videos. Esta fase tuvo una duración de 51 días.

   c. **Opciones de video**
      En esta etapa se implementó la reproducción de la película con sus distintas opciones. Esta fase tuvo una duración de 8 días.

   d. **Opciones de audio**
      En esta etapa se implementó el modo de sonido que se quiere al reproducir la película. Esta fase tuvo una duración de 6 días.
e. Extras disponibles

En esta etapa se implementó la galería fotográfica para la visualización de las imágenes, el trailer de una película, la sección para ver los eventos relacionados con la película y también la sección para ver los archivos que hemos descargado del servidor. Esta fase tuvo una duración de 40 días.

7. Pruebas

En esta etapa se comprueba el funcionamiento de la aplicación para corregir posibles bugs que pueda haber durante el transcurso de la ejecución. Esta fase tuvo una duración de 14 días.

8. Redacción de la memoria

Esta etapa dura prácticamente durante todo el transcurso del proyecto, en el cual se redacta toda la memoria del proyecto mediante se va avanzando en la realización de la aplicación. Necesité 160 días para realizarla.
Programación en red para Blu-ray

A continuación se puede ver el diagrama de Gantt con las diferentes etapas del proyecto:

<table>
<thead>
<tr>
<th></th>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Estudio de la propuesta del proyecto</td>
<td>7 días</td>
<td>jue 21/03/09</td>
<td>vie 29/03/09</td>
</tr>
<tr>
<td>2</td>
<td>Familiarización con el lenguaje y el entorno de programación</td>
<td>40 días</td>
<td>lun 01/03/09</td>
<td>vie 24/03/09</td>
</tr>
<tr>
<td>3</td>
<td>Análisis y especificación</td>
<td>15 días</td>
<td>lun 27/03/09</td>
<td>vie 14/04/09</td>
</tr>
<tr>
<td>4</td>
<td>Diseño y casos de uso</td>
<td>24 días</td>
<td>lun 17/03/09</td>
<td>jue 18/03/09</td>
</tr>
<tr>
<td>5</td>
<td>Diseño de la interfaz gráfica</td>
<td>3 días</td>
<td>vie 19/03/09</td>
<td>mar 23/03/09</td>
</tr>
<tr>
<td>6</td>
<td>Implementación de la interfaz gráfica</td>
<td>105 días</td>
<td>vie 19/03/09</td>
<td>jue 14/05/09</td>
</tr>
<tr>
<td>7</td>
<td>Implementación de la descarga de contenido</td>
<td>51 días</td>
<td>vie 19/03/09</td>
<td>vie 27/02/09</td>
</tr>
<tr>
<td>8</td>
<td>Implementación de las opciones de video</td>
<td>8 días</td>
<td>lun 02/03/09</td>
<td>mier 11/03/09</td>
</tr>
<tr>
<td>9</td>
<td>Implementación de las opciones de audio</td>
<td>6 días</td>
<td>jue 12/03/09</td>
<td>jue 19/03/09</td>
</tr>
<tr>
<td>10</td>
<td>Implementación de los extras</td>
<td>40 días</td>
<td>vie 23/03/09</td>
<td>jue 14/03/09</td>
</tr>
<tr>
<td>11</td>
<td>Pruebas</td>
<td>14 días</td>
<td>vie 15/03/09</td>
<td>mier 30/03/09</td>
</tr>
<tr>
<td>12</td>
<td>Redacción de la memoria</td>
<td>160 días</td>
<td>lun 27/10/08</td>
<td>vie 05/06/09</td>
</tr>
</tbody>
</table>

Fig 12. Fases del proyecto
3.3 Modelo conceptual de la aplicación

En este primer modelo se hizo una aproximación de cómo debería ser el diagrama de clases de la aplicación cliente. En este caso se dividió la aplicación en 6 clases.

- **PlayMenuMain**
  Esta es la clase principal, la cual se encarga de permitir que un aparato digital pueda iniciar, pausar y terminar la aplicación.

- **MenusFeature**
  Esta clase será la encargada de capturar todos los eventos que van sucediendo mientras se va ejecutando la aplicación, para saber en que estado de la ejecución se encuentra y ordenar al resto de clases que realicen las peticiones que demanda el usuario.
• **Menu**
  Esta clase es la que sabe cómo se dibuja cada menú y cómo se tienen que tratar todos los eventos de teclado.

• **TratamientoConexión**
  Esta clase se encarga de tratar las peticiones que hace el usuario al servidor.

• **Servidor**
  Esta es la clase principal, la cual se encarga de aceptar las conexiones de los clientes que quieren descargar contenidos.

• **HiloServidor**
  Esta clase es la encargada de recibir y enviar datos al cliente según las peticiones que vaya realizando.

• **PeticionesCliente**
  Esta clase es la que tiene implementado de que manera se tratan las peticiones del cliente.
Programación en red para Blu-ray

A continuación podemos ver el diagrama conceptual de la aplicación:

Fig 14. Diagrama conceptual de la aplicación
3.4 Actores
En esta aplicación interviene solo un actor. El cliente interviene en los casos de uso para encargarse de reproducir el video, pausar o dejar de pausar el video, detener el video, elegir el modo de audio, visualizar la galería fotográfica, ver los archivos descargados, ver los eventos surgidos de la película, conectarse al servidor, visualizar las carpetas y archivos que contiene este, descargar archivos y cancelar descargas.

3.5 Casos de uso y contratos del cliente
En este apartado se comentan los casos de uso y los contratos de la aplicación.

3.5.1 Caso de uso “reproducir video”
Actores: usuario.
Propósito: el usuario visualice el video.
Resumen: el usuario elige la opción del “Play” del menú película o la opción trailer del menú extras y comienza la visualización del video.
Curso típico:
1. El usuario selecciona la opción del “Play” para reproducir la película o la opción trailer del menú extras.
2. Se reproduce el video.
Contrato de los eventos de sistema

Nombre del evento de sistema → verVideo()
Parámetros → Ninguno.
PRE → La opción del menú escogida sea el “Play” del menú de la película o que se haya seleccionado la opción “Trailer” del menú extras.
POST → Comienza la visualización del video.

3.5.2 Caso de uso “pausar video”

Actores: usuario.
Propósito: el usuario pause el video.
Resumen: el usuario elige la opción de pausar el video dentro del menú de opciones de reproducción.
Curso típico:

1. El usuario selecciona la opción del “Pause” para pausar el video.
2. El usuario pausa el video.

Contrato de los eventos de sistema

Nombre del evento de sistema → verVideo()
Parámetros → Ninguno.
PRE → Estar reproduciendo el video y estar en el menú de opciones de reproducción.
POST → Se pausa el video.
3.5.3 Caso de uso “quitar el pause al video”

Actores: usuario.

Propósito: el usuario quita el pause al video.

Resumen: el usuario elige la opción del pause para dejar de pausar el video dentro del menú de opciones de reproducción.

Curso típico:

1. El usuario selecciona la opción del “Pause” para quitar el pause del video.
2. El usuario quita el pause del video.

contrato de los eventos de sistema

Nombre del evento de sistema → verVideo()

Parámetros → Ninguno.

PRE → El video debe estar pausado y se debe estar en el menú de opciones de reproducción.

POST → Se quita el pause al video.

3.5.4 Caso de uso “parar la reproducción del video”

Actores: usuario.

Propósito: el usuario para la reproducción del video.

Resumen: el usuario elige la opción de detener el vídeo dentro del menú de opciones de reproducción.
Curso típico:

1. El usuario selecciona la opción del “Stop” para parar la reproducción del video.
2. El usuario detiene la reproducción del video.

Contrato de los eventos de sistema

Nombre del evento de sistema → verVideo()
Parámetros → Ninguno.
PRE → Estar reproduciendo el video y estar en el menú de opciones de reproducción.
POST → Se detiene la reproducción del video.

3.5.5 Caso de uso “elegir modo de audio”

Actores: usuario.
Propósito: el usuario escoja el modo de audio.
Resumen: el usuario elige el modo de audio que le interesa dentro del menú de opciones de audio.
Curso típico:

1. El usuario selecciona el modo de audio que quiere.
2. Se ha cambiado el modo de audio al escogido por el usuario.
Contrato de los eventos de sistema

Nombre del evento de sistema → elegirAudio(peliActiva)

Parámetros → peliActiva: booleano

PRE → Estar dentro del menú de las opciones de audio.

POST → Cambia el modo de audio.

3.5.6 Caso de uso “visualizar galería fotográfica”

Actores: usuario.

Propósito: el usuario visualice la galería fotográfica.

Resumen: el usuario dentro del menú de extras elige visualizar la galería de imágenes.

Curso típico:

1. El usuario selecciona la opción de visualizar la galería fotográfica.
2. El usuario visualiza la galería de imágenes.
Contrato de los eventos de sistema

Nombre del evento de sistema → `paintVisualizarGaleria(BluGraphics)`
Parámetros → `BluGraphics`: dibujo menú
PRE → Estar dentro del menú de las opciones de extras y que la opción elegida sea “Galería imágenes”.
POST → Se visualiza la galería fotográfica.

3.5.7 Caso de uso “visualizar imagen de la galería fotográfica”

Actores: usuario.
Propósito: el usuario visualice la imagen seleccionada.
Resumen: el usuario selecciona una imagen y este la ve en tamaño original.
Curso típico:

1. El usuario selecciona la imagen que quiere visualizar a tamaño original.
2. El usuario visualiza la imagen.
Contrato de los eventos de sistema
Nombre del evento de sistema → PaintImagenAmpliada(BluGraphics)
Parámetros → BluGraphics: dibujo menú
PRE → Estar visualizando la galería de imágenes.
POST → Se visualiza en tamaño original la imagen seleccionada.

3.5.8 Caso de uso “petición de evento actualizado”

Actores: usuario.

Propósito: el cliente reciba por parte del servidor el archivo con la última actualización de los eventos.

Resumen: el servidor recibe la petición de descarga del archivo de los eventos y este se lo envía al cliente.

Curso típico:
1. El servidor recibe la petición del archivo que quiere descargar el cliente.
2. El servidor le envía el archivo al cliente.

Curso alternativo:
1, 2. Se pierde la conexión, por lo tanto el usuario visualizará la última versión que disponía en su ordenador.
Contrato de los eventos de sistema

Nombre del evento de sistema → peticiónEvento()
Parámetros → Ninguno.
PRE → Estar el cliente conectado al servidor.
POST → El cliente recibe el archivo con los eventos actualizados.

3.5.9 Caso de uso “comprobar si hay una versión de eventos actualizada”

Actores: usuario.

Propósito: el cliente compruebe si hay una versión actualizada de los eventos y que la guarde en memoria.

Resumen: el usuario comprueba si la versión descargada de eventos está más actualizada que la que viene por defecto y la guarda en memoria.

Curso típico:

1. Se obtiene el texto de los archivos.
2. Se comprueba que versión de los archivos está más actualizada.
3. Se guarda la última versión.

Curso alternativo:

1. No existe el archivo descargado, por lo tanto se utiliza la versión que viene por defecto para visualizarla.
Contrato de los eventos de sistema

Nombre del evento de sistema → EventoActualizado()
Parámetros → Ninguno.
PRE → Ninguno.
POST → El cliente guarda la versión más reciente de eventos.

3.5.10 Caso de uso “visualizar la carpeta de los archivos descargados”

Actores: usuario.
Propósito: el usuario visualice la lista de archivos que tiene descargados.
Resumen: el usuario dentro del menú de extras selecciona la opción de visualizar las descargas.
Curso típico:
1. El usuario selecciona la opción descargas del menú de extras.
2. El usuario visualiza la lista de archivos que hay descargados.
Curso alternativo:
1. No existe ningún archivo descargado. Por lo tanto, se nos mostrará un mensaje indicándonoslo.
Contrato de los eventos de sistema

Nombre del evento de sistema → cargarArchivos()
Parámetros → Ninguno.
PRE → Estar en el menú extras y que la opción elegida sea “Descargas”.
POST → Se visualiza la lista de archivos descargados.

3.5.11 Caso de uso “visualizar un archivo de texto descargado”

Actores: usuario.

Propósito: el usuario visualice el contenido del archivo de texto deseado.

Resumen: el usuario dentro de la sección “Descargas” del menú extras, selecciona la opción de visualizar un archivo de texto.

Curso típico:

1. El usuario selecciona un archivo de texto para visualizar su contenido.
2. El usuario visualiza el texto del archivo seleccionado.

Curso alternativo:

1. Se produce un error en la lectura del archivo. Por lo tanto, nos envía al menú de extras y se nos muestra un mensaje de error para indicarlo.
Contrato de los eventos de sistema

Nombre del evento de sistema → verArchivo()
Parámetros → Ninguno.
PRE → Estar en la sección “Descargas” del menú extras.
POST → Se visualiza el archivo de texto.

3.5.12 Caso de uso “visualizar una imagen descargada”

Actores: usuario.
Propósito: el usuario visualice la imagen seleccionada.
Resumen: el usuario dentro de la sección “Descargas” del menú extras selecciona la opción de visualizar una imagen.

Curso típico:
1. El usuario selecciona una imagen para visualizar.
2. El usuario visualiza la imagen seleccionada.

Curso alternativo:
1. Se produce un error en la visualización de la imagen. Por lo tanto, nos envía al menú de extras y se nos muestra un mensaje de error para indicarlo.
Contrato de los eventos de sistema

Nombre del evento de sistema ➔ verArchivo()

Parámetros ➔ Ninguno.

PRE ➔ Estar en la sección “Descargas” del menú extras.

POST ➔ Se visualiza la imagen.

3.5.13 Caso de uso “conexión con el servidor”

Actores: usuario.

Propósito: el usuario se conecte al servidor.

Resumen: el servidor recibe una petición de conexión por parte de un cliente y esta es aceptada.

Curso típico:

1. El servidor recibe la petición de conexión por parte de un usuario.
2. El servidor acepta la nueva conexión.

Curso alternativo:

1. No se puede realizar la conexión con el servidor. Aparece un mensaje de error indicándose que no se ha podido realizar la conexión.
**Contrato de los eventos de sistema**

Nombre del evento de sistema → conexiónInicial()
Parámetros → Ninguno.
PRE → Ninguno.
POST → Se conecta el cliente al servidor.

**3.5.14 Caso de uso “navegar entre los directorios del servidor”**

Actores: usuario.

Propósito: el usuario visualiza un subdirectorio del servidor.

Resumen: el servidor recibe una petición de uno de las carpetas que quiere ver el contenido el cliente y se la envía.

Curso típico:

1. El servidor recibe la petición del directorio que quiere ver el usuario.
2. El servidor le envía el contenido de ese directorio al cliente.
3. El cliente recibe la información de la carpeta que quiere visualizar.

Curso alternativo:

1, 2. Se pierde la conexión con el servidor. Se carga automáticamente el menú inicial de la aplicación del cliente indicándole un mensaje de error.
**Contrato de los eventos de sistema**

Nombre del evento de sistema → enviarPeticion()
Parámetros → Ninguno.
PRE → Estar el cliente conectado al servidor.
POST → El usuario visualiza el contenido del directorio que ha seleccionado del servidor.

### 3.5.15 Caso de uso “descargar archivo del servidor”

Actores: usuario.

Propósito: el usuario descargue un archivo del servidor.

Resumen: el servidor recibe la petición de descarga de un archivo por parte de un cliente y se lo envía.

Curso típico:

1. El servidor recibe la petición del archivo que quiere descargar el cliente.
2. El servidor le envía el archivo al cliente.

Curso alternativo:

1. 2. Se pierde la conexión con el servidor. Se le tiene que indicar al usuario mediante un mensaje de error y le envía automáticamente al menú inicial de la aplicación.
Contrato de los eventos de sistema
Nombre del evento de sistema → enviarPeticion()
Parámetros → Ninguno.
PRE → Estar conectado el cliente al servidor.
POST → El cliente descarga el archivo seleccionado del servidor.

3.5.16 Caso de uso “cancelar la descarga de un archivo”

Actores: usuario.

Propósito: el usuario cancele la descarga del archivo.

Resumen: el usuario selecciona cancelar la descarga de uno de los archivos del servidor, elimina la parte que se haya podido descargar de este del disco duro y le envía al menú anterior donde se encontraba.

Curso típico:

1. El usuario selecciona la opción de cancelar la descarga.
2. El usuario vuelve al menú anterior donde se encontraba.
Contrato de los eventos de sistema
Nombre del evento de sistema → cancelarDescarga()
Parámetros → Ninguno.
PRE → Estar descargando un archivo.
POST → El usuario ha cancelado la descarga del archivo y regresa al menú anterior donde puede visualizar carpetas o seleccionar archivos para descargar.

3.5.17 Caso de uso “desconectar del servidor”
Actores: usuario.
Propósito: el usuario se desconecte del servidor.
Resumen: el usuario decide dejar de visualizar contenido adicional que hay en el servidor y regresa al menú inicial.
Curso típico:
1. El usuario selecciona regresar al menú inicial.
2. El usuario se desconecta del servidor.
Contrato de los eventos de sistema
Nombre del evento de sistema → cerrarConexion()
Parámetros → Ninguno.
PRE → Estar conectado al servidor y visualizando la raíz de la carpeta de contenidos adicionales que dispone.
POST → El usuario se desconecta del servidor.

3.6 Estudio económico

En este apartado damos los datos aproximados de lo que debe ser el coste del proyecto en horas y precio.

- En esta tabla podemos observar el coste del material que utilizamos para realizar la aplicación.

<table>
<thead>
<tr>
<th>Producto</th>
<th>Coste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>0 €</td>
</tr>
<tr>
<td>Netbeans 6.0</td>
<td>0 €</td>
</tr>
<tr>
<td>InterVideo WinDVD 8</td>
<td>0 €</td>
</tr>
</tbody>
</table>

El coste del software utilizado es 0 €.
En la siguiente tabla veremos el coste de las diferentes etapas de la que consta el proyecto.

<table>
<thead>
<tr>
<th>Etapa</th>
<th>Precio/hora</th>
<th>Horas</th>
<th>Coste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiarización con el lenguaje y el entorno de trabajo</td>
<td>30 €</td>
<td>64</td>
<td>1920 €</td>
</tr>
<tr>
<td>Análisis y especificación</td>
<td>40 €</td>
<td>32</td>
<td>1280 €</td>
</tr>
<tr>
<td>Diseño y casos de uso</td>
<td>40 €</td>
<td>50</td>
<td>2000 €</td>
</tr>
<tr>
<td>Diseño de la interfaz gráfica</td>
<td>30 €</td>
<td>10</td>
<td>300 €</td>
</tr>
<tr>
<td>Implementación</td>
<td>30 €</td>
<td>120</td>
<td>3600 €</td>
</tr>
<tr>
<td>Pruebas</td>
<td>30 €</td>
<td>24</td>
<td>720 €</td>
</tr>
</tbody>
</table>

Como podemos ver el coste del proceso del proyecto es de 9820 €.

El resultado final del coste de la aplicación es de 9820 € gracias a que el software utilizado es libre y gratuito.
4. Diseño

4.1 Diagramas de colaboración

Los siguientes diagramas de colaboración describen el funcionamiento más detallado de cada evento de sistema y muestran las interacciones que ocurren entre los objetos que participan en una situación determinada. Utilizando los principios GRASP (General Responsibility Assignment Software Patterns) se definen las responsabilidades de cada componente y sus interacciones con otros componentes.

4.1.1 Caso de uso “reproducir video”

Nombre del evento de sistema → verVideo()
Parámetros → Ninguno.
PRE → La opción del menú escogida sea el “Play” del menú de la película o que se haya seleccionado la opción “Trailer” del menú extras.
POST → Comienza la visualización del video.

Asignación de responsabilidades:
Por el principio de controlador, MenuVideo se encarga de recibir el evento ya que sabe que opción de reproducción ha sido elegida por el usuario y crea un PlaybackManager siguiendo el principio de creador. Después por el principio de alta cohesión y para delegar tareas, la clase PlaybackManager se encarga de reproducir el video. Esto es para que MenuVideo no haga tantas cosas y delegue tareas.
Nota: PlaybackManager es una clase que viene definida en la librería Blubench del módulo Playback. Para más información consultar el apartado del diagrama de clases de la aplicación.

4.1.2 Caso de uso “pausar video”
Programación en red para Blu-ray

Nombre del evento de sistema → verVideo()
Parámetros → Ninguno.
PRE → Estar reproduciendo el video y estar en el menú de opciones de reproducción.
POST → Se pausa el video.

Asignación de responsabilidades:
Por el principio de controlador, MenuVideo se encarga de recibir el evento ya que sabe
que opción de reproducción ha sido elegida por el usuario y crea un PlaybackManager
siguiendo el principio de creador. Después por el principio de alta cohesión y para
delegar tareas, la clase PlaybackManager se encarga de pausar el video. Esto es para que
MenuVideo no haga tantas cosas y delegue tareas.

![Diagrama de colaboración “pausar video”](image)

Nota: PlaybackManager es una clase que viene definida en la librería Blubench del
módulo Playback. Para más información consultar el apartado del diagrama de clases de
la aplicación.
4.1.3 Caso de uso “quitar el pause al video”

Nombre del evento de sistema → verVideo()
Parámetros → Ninguno.
PRE → El video debe estar pausado y se debe estar en el menú de opciones de reproducción.
POST → Se quita el pause al video.

Asignación de responsabilidades:
Por el principio de controlador, MenuVideo se encarga de recibir el evento ya que sabe que opción de reproducción ha sido elegida por el usuario y crea un PlaybackManager siguiendo el principio de creador. Después por el principio de alta cohesión y para delegar tareas, la clase PlaybackManager se encarga de quitar la pausa del video. Esto es para que MenuVideo no haga tantas cosas y delegue tareas.
Fig 17. Diagrama de colaboración “quitar el pause al video”

Nota: PlaybackManager es una clase que viene definida en la librería Blubench del módulo Playback. Para más información consultar el apartado del diagrama de clases de la aplicación.

4.1.4 Caso de uso “parar la reproducción del video”
Nombre del evento de sistema ➔ `verVideo()`
Parámetros ➔ Ninguno.
PRE ➔ Estar reproduciendo el video y estar en el menú de opciones de reproducción.
POST ➔ Se detiene la reproducción del video.

Asignación de responsabilidades:
Por el principio de controlador, `MenuVideo` se encarga de recibir el evento ya que sabe que opción de reproducción ha sido elegida por el usuario y crea un `PlaybackManager` siguiendo el principio de creador. Después por el principio de alta cohesión y para delegar tareas, la clase `PlaybackManager` se encarga de detener el video, que en realidad lo que realiza es empezar a reproducir otro video, el cual siempre visualizamos mientras navegamos por los menús. Esto es para que `MenuVideo` no haga tantas cosas y delegue tareas.

Fig 18. Diagrama de colaboración “parar la reproducción del video”
Nota: PlaybackManager es una clase que viene definida en la librería Blubench del módulo Playback. Para más información consultar el apartado del diagrama de clases de la aplicación.

4.1.5 Caso de uso “elegir modo de audio”

Nombre del evento de sistema → elegirAudio(peliActiva)
Parámetros → peliActiva: booleano
PRE → Estar dentro del menú de las opciones de audio.
POST → Cambia el modo de audio.

Asignación de responsabilidades:
Por el principio de controlador, MenuAudio se encarga de recibir el evento y saber que modo de audio ha sido elegido por el usuario. Debido a eso crea un PlaybackManager siguiendo el principio de creador. Después por el principio de alta cohesión y para delegar tareas, he decidido que la clase PlaybackManager se encargue de ejecutar el sistema de audio que elija el usuario. Esto es para que MenuAudio no haga tantas cosas y delegue tareas.
4.1.6 Caso de uso “visualizar galería fotográfica”

Nombre del evento de sistema → paintVisualizarGaleria(BluGraphics)
Parámetros → BluGraphics: dibujo menú
PRE → Estar dentro del menú de las opciones de extras y que la opción elegida sea “Galeria imágenes”.
POST → Se visualiza la galería fotográfica.
Asignación de responsabilidades:
Por el principio de controlador, MenuFotos se encarga de recibir el evento y saber que imágenes se tienen que visualizar por pantalla de todas las que hay disponibles en la galería fotográfica. Después por el principio de alta cohesión y para delegar tareas, la clase BluGraphics es la que sabe como se dibujan estas imágenes. Como podemos ver se realiza una iteración para dibujar las N fotos que MenuFotos decide que se tienen que visualizar.

Nota: BluGraphics es una clase que viene definida en la librería Blubench del módulo Graphics. Para más información consultar el apartado del diagrama de clases de la aplicación.
4.1.7 Caso de uso “visualizar imagen de la galería fotográfica”

Nombre del evento de sistema → PaintImagenAmpliada(BluGraphics)
Parámetros → BluGraphics: dibujo menú
PRE → Estar visualizando la galería de imágenes.
POST → Se visualiza en tamaño original la imagen seleccionada.

Asignación de responsabilidades:
Por el principio de controlador, MenusFotos se encarga de recibir el evento y saber que imagen ha seleccionado el usuario para visualizar a tamaño original. Después por el principio de alta cohesión y para delegar tareas, la clase BluGraphics es la que sabe como se dibuja la imagen por pantalla.

Fig 21. Diagrama de colaboración “visualizar imagen de la galería fotográfica”
Nota: BluGraphics es una clase que viene definida en la librería Blubench del módulo Graphics. Para más información consultar el apartado del diagrama de clases de la aplicación.

4.1.8 Caso de uso “petición de evento actualizado”

Nombre del evento de sistema ⇒ peticiónEvento()
Parámetros ⇒ Ninguno.
PRE ⇒ Estar el cliente conectado al servidor.
POST ⇒ El cliente recibe el archivo con los eventos actualizados.

Asignación de responsabilidades:
Por el principio de controlador, HiloServidor se encarga de recibir el evento del cliente para enviarle los datos necesarios para que reciba el archivo con los eventos actualizados. Por el principio de alta cohesión y para delegar tareas, he decidido que la clase PeticionesCliente se encargue de recibir la petición y de tramitarla. Después se le envía a Eventos el archivo con los eventos actualizados. Entonces utilizando el principio de experto, ya que Eventos es quien tiene mayor conocimiento de estos datos recibidos, guarda el archivo.
4.1.9 Caso de uso “comprobar si hay una versión de eventos actualizada”
Nombre del evento de sistema → EventoActualizado()
Parámetros → Ninguno.
PRE → Ninguno.
POST → El cliente guarda la versión más reciente de eventos.

Asignación de responsabilidades:
Por el principio de controlador, Eventos se encarga de recibir el evento para comprobar si el último archivo de eventos recibido está más actualizado que la versión que teníamos. Después por el principio de experto ya que Eventos es quien tiene la información necesaria para mirarlo, se obtiene el texto de los dos archivos a comparar, se comprueba que versión está más actualizada y se guarda la última versión de texto de eventos para que posteriormente se nos pueda mostrar por pantalla para visualizarlo.

Fig 23. Diagrama de colaboración “comprobar evento actualizado”
4.1.10 Caso de uso “visualizar la carpeta de los archivos descargados”

Nombre del evento de sistema ➔ cargarArchivos()
Parámetros ➔ Ninguno.
PRE ➔ Estar en el menú extras y que la opción elegida sea “Descargas”.
POST ➔ Se visualiza la lista de archivos descargados.

Asignación de responsabilidades:
Por el principio de controlador, ArchivosDescargados se encarga de recibir el evento para guardar el nombre de todos los archivos que hay descargados. Después por el principio de experto, ArchivosDescargados es quien tiene la información necesaria para comprobar si hay archivos descargados.
Fig 24. Diagrama de colaboración “visualizar la carpeta de archivos descargados”

4.1.11 Caso de uso “visualizar un archivo de texto descargado”

Nombre del evento de sistema ➔ verArchivo()
Parámetros ➔ Ninguno.
PRE ➔ Estar en la sección “Descargas” del menú extras.
POST ➔ Se visualiza el archivo de texto.
Asignación de responsabilidades:
Por el principio de controlador, ArchivosDescargados se encarga de recibir el evento para visualizar el archivo de texto deseado. Después por el principio de alta cohesión y para delegar tareas, Eventos es quien tiene la información necesaria para leer el contenido del archivo de texto, para posteriormente se pueda visualizar el contenido por pantalla.

Fig 25. Diagrama de colaboración “visualizar un archivo de texto descargado”

4.1.12 Caso de uso “visualizar una imagen descargada”

Nombre del evento de sistema ➔ verArchivo()
Parámetros ➔ Ninguno.
PRE ➔ Estar en la sección “Descargas” del menú extras.
POST ➔ Se visualiza la imagen.
Asignación de responsabilidades:
Por el principio de controlador, ArchivosDescargados se encarga de recibir el evento para visualizar la imagen seleccionada. Después utilizando el principio experto, ya que ArchivosDescargados es quien posee mayor conocimiento tiene de los datos, carga la imagen seleccionada.

Fig 26. Diagrama de colaboración “visualizar una imagen descargada”

4.1.13 Caso de uso “conexión con el servidor”
Nombre del evento de sistema → conexiónInicial()
Parámetros → Ninguno.
PRE → Ninguno.
POST → Se conecta el cliente al servidor.

Asignación de responsabilidades:
Por el principio de controlador, Servidor se encarga de recibir el evento para aceptar la petición de conexión por parte de un cliente. Por el principio de creador, Servidor se encarga de crear un HiloServidor para la conexión de cada cliente y este a la vez crea un PeticionesCliente para tratar todas las peticiones que vaya realizando el cliente durante la conexión. Finalmente por el principio de alta cohesión el HiloServidor le envía los datos iniciales al cliente mediante MenuConexionCliente.

Fig 27. Diagrama de colaboración “conexión con el servidor”
4.1.14 Caso de uso “navegar entre los directorios del servidor”

Nombre del evento de sistema → enviarPeticion()
Parámetros → Ninguno.
PRE → Estar conectado al servidor.
POST → El usuario visualiza el contenido del directorio que ha seleccionado del servidor.

Asignación de responsabilidades:
Por el principio de controlador, HiloServidor se encarga de recibir el evento del cliente para enviarle los datos necesarios para que el usuario visualice el nuevo contenido. Por el principio de alta cohesión y para delegar tareas, he decidido que la clase PeticionesCliente se encargue de recibir la petición y de tramitarla. Después también utilizando el principio de alta cohesión se le envía una respuesta a TratamientoConexion. Entonces utilizando el principio de experto, ya que TratamientoConexion es quien mayor conocimiento de estos datos recibidos, realiza los trámites necesarios para prepararse para recibir por parte del servidor los datos para que el usuario visualice el nuevo directorio. Una vez hecho esto PeticionesCliente le envía el directorio. Finalmente por el principio de Experto TratamientoConexion guarda el directorio que acaba de recibir.
Programación en red para Blu-ray

Fig 28. Diagrama de colaboración “navegar entre los directorios del servidor”

4.1.15 Caso de uso “descargar archivo del servidor”
Programación en red para Blu-ray

Nombre del evento de sistema → enviarPeticion()
Parámetros → Ninguno.
PRE → Estar conectado al servidor.
POST → El usuario descarga el archivo seleccionado del servidor.

Asignación de responsabilidades:
Por el principio de controlador, HiloServidor se encarga de recibir el evento del cliente para enviarle los datos necesarios para que reciba el archivo que desea. Por el principio de alta cohesión y para delegar tareas, he decidido que la clase PeticionesCliente se encargue de recibir la petición y de tramitarla. Después también utilizando el principio de alta cohesión se le envía una respuesta a TratamientoConexión. Entonces utilizando el principio de experto, ya que TratamientoConexión es quien mayor conocimiento de estos datos recibidos, creó un DescargaFichero siguiendo el principio creador. Una vez hecho esto el servidor le envía el archivo al cliente.

Fig 29. Diagrama de colaboración “descargar archivo del servidor”
4.1.16 Caso de uso “cancelar la descarga de un archivo”

Nombre del evento de sistema → cancelarDescarga()
Parámetros → Ninguno.
PRE → Estar descargando un archivo.
POST → El usuario regresa al menú anterior donde puede visualizar carpetas o seleccionar archivos para descargar.

Asignación de responsabilidades:
Por el principio de controlador, MenuConexionCliente se encarga de recibir el evento para cancelar la descarga del fichero. Después utilizando el principio experto, ya que TratamientoConexion es quien mayor conocimiento tiene de los datos, primeramente se cierra la conexión de la descarga.
4.1.17 Caso de uso “desconectar del servidor”

Nombre del evento de sistema → cerrarConexion()
Parámetros → Ninguno.
PRE → Estar conectado al servidor y visualizando la raíz de la carpeta de contenidos adicionales que dispone.
POST → El usuario se desconecta del servidor.
Asignación de responsabilidades:
Por el principio de controlador, MenuConexionCliente se encarga de recibir el evento para desconectarse del servidor. Después utilizando el principio experto, ya que MenuConexionCliente es quien mayor conocimiento tiene de los datos, cierra la conexión con el servidor.

Fig 31. Diagrama de colaboración “desconectar del servidor”
4.2 Diseño gráfico

En este apartado visualizaremos el diseño de la interfaz gráfica de la aplicación.

4.2.1 Pantalla del menú inicial

Esta pantalla es la primera que visualiza el usuario al inicializar la aplicación. En ella se podrá escoger entre ir al menú de reproducción de la película, el menú con los modos de audio, el menú con los extras que dispone la película y el menú de descarga de contenidos adicionales.

Fig 32. Diseño del menú inicial
4.2.2 Pantalla del menú reproducir película

Esta pantalla es la que visualiza el usuario después de haber seleccionado la opción del menú inicial “Reproducir película”. En ella se podrá escoger entre inicializar la reproducción de la película, pausarla en un momento determinado y detener su reproducción.

![Diagrama de la pantalla del menú reproducir película]

Botón para ver la película.
Botón para pausar la película.
Botón para detener la película.

Fig 33. Diseño del menú de opciones de reproducción
4.2.3 Pantalla del menú modo de audio

Esta pantalla es la que visualiza el usuario después de haber seleccionado la opción del menú inicial “Audio”. En ella se podrá seleccionar entre las opciones del modo de audio “Estereo PCM” y el “5.1 Dolby Digital”.

Fig 34. Diseño del menú de los modos de audio
4.2.4 Pantalla del menú extras

Esta pantalla es la que visualiza el usuario después de haber seleccionado la opción del menú inicial “Extras”. En ella se podrá seleccionar entre las opciones de visualizar la galería fotográfica, ver el trailer que lleva o mirar la sección de eventos sobre la película.

Fig 35. Diseño del menú de extras
4.2.5 Pantalla de la galería fotográfica

Esta pantalla es la que visualiza el usuario después de haber seleccionado la opción del menú extras “Galería fotográfica”. En ella se puede visualizar una previsualización de hasta cinco imágenes por pantalla simultáneamente.

Fig 36. Diseño de la galería fotográfica
4.2.6 Pantalla de una imagen ampliada de la galería de imágenes

Esta pantalla es la que visualiza el usuario después de haber seleccionado una de las imágenes que se estaban previsualizando por pantalla para verla ampliada.

Fig 37. Diseño de la imagen seleccionada ampliada
4.2.7 Pantalla de la sección de eventos

Esta pantalla es la que visualiza el usuario después de haber seleccionado la opción del menú extras “Eventos”. En ella se puede visualizar los eventos actualizados que van surgiendo sobre la película de que viene en el disco Blu-ray.

Fig 38. Diseño de la sección de eventos
4.2.8 Pantalla de la sección de archivos descargados

Esta pantalla es la que se visualiza después de haber seleccionado la opción “Descargas” del menú de extras. En ella podemos visualizar una lista de los archivos que hemos descargado con la aplicación.

Fig 39. Diseño de la sección de descargas
4.2.9 Pantalla para visualizar un archivo de texto descargado.

Esta pantalla es la que visualiza el usuario después de haber seleccionado visualizar un archivo de texto descargado.

![Diagrama de pantalla para visualizar un archivo de texto descargado]

Campo de texto donde podemos leer el contenido de un archivo descargado.

Fig 40. Diseño para visualizar un archivo de texto
4.2.10 Pantalla para visualizar una imagen descargada.

Esta pantalla es la que visualiza el usuario después de haber seleccionado ver una de las imágenes de la lista de archivos descargados de la sección “Descargas” que pertenece al menú extras.

Fig 41. Diseño para visualizar una imagen
4.2.11 Pantalla del menú descarga de contenidos

Esta pantalla es la que visualiza el usuario después de haber seleccionado la opción del menú inicial “Descarga de contenidos”. En ella se podrá seleccionar cualquier carpeta para ver el subdirectorio de esta o seleccionar algún archivo para inicializar la descarga.

Fig 42. Diseño del menú descarga de contenidos
4.2.12 Pantalla de descarga de un archivo

Esta pantalla es la que visualiza el usuario después de haber seleccionado el archivo a descargar del menú “Descarga de contenidos”. En ella se podrá ver el estado de la descarga actual en cada momento.

Fig 43. Diseño de la pantalla de descarga de un archivo
4.3 Diagrama de clases

Como ya comenté brevemente en la introducción, a la hora de realizar mi aplicación disponía de una aplicación base que dispone de una librería de usuario con la cual poder a partir de ahí realizar mi autoría de Blu-ray.

Para que se vea el estado inicial de la aplicación base que he utilizado, he realizado el siguiente diagrama de clases. Este nos permite observar el estado inicial de la aplicación que utilicé.

![Diagrama de clases](image)

*Fig 44. Diagrama de clases de la aplicación base*

En mi aplicación como se puede ver viene implementado el “main” de la aplicación (en este caso “PlayMenuMain”), con el cual nos permite que una fuente externa (en este caso un reproductor de Blu-ray) pueda entre otras cosas inicializar y finalizar la aplicación en cualquier momento.

También disponemos de “MenusFeature”, que es una clase en la cual vienen declarados unos métodos sin definir que podemos ver en la imagen anterior. A partir de ahí yo me encargo de implementarlos y así crear mi autoría de Blu-ray. En este caso tenemos un método encargado de inicializar los datos, también disponemos de un método que lo
llama el hilo principal de la aplicación (en “PlayMenuMain”) cada “x” tiempo para refrescar la pantalla (este es el único método implementado de esta clase), otro encargado para dibujar la interfaz gráfica de la aplicación y el último que se dedica a capturar los eventos de teclado para realizar su función correspondiente dependiendo de lo que demande el usuario.

Para el funcionamiento de la aplicación disponemos de una librería de usuario llamada “Blubench”. Para más información consultar el proyecto [6].

Esta es una librería de usuario que utilizaré para el desarrollo de la autoría de Blu-ray. Está estructurada en diversos módulos que nos permiten agilizar el desarrollo de aplicaciones en BD-J y promueve una interfaz que facilita la compatibilidad entre los diversos reproductores.

A continuación explico un poco más la función de cada uno de los módulos de los que dispone:

- **Módulo Graphics**
  Se encarga de encapsular toda la funcionalidad gráfica, incluyendo la gestión de imágenes y fuentes.

- **Módulo GUI**
  Proporciona un bucle principal para la gestión de los elementos de la interfaz gráfica.

- **Módulo Playback**
  Aquí se reúnen todas las operaciones relacionadas con la reproducción de video.

- **Módulo Debug**
  Accede desde cualquier lugar de la aplicación a un servicio que permite mostrar mensajes de error y excepciones.
- **Módulo Sound**
  Aquí se reúnen todas las operaciones relacionadas con la reproducción de sonidos.

- **Módulo Input**
  Gestiona de forma centralizada la entrada del usuario a través del control remoto.

A partir de la aplicación base que disponemos y del diseño realizado surge un nuevo diagrama. Como podemos ver a partir de la clase Menu y aplicando la alta cohesión, he creado varios tipos de Menu para cada uno de los diferentes menús que disponemos en la aplicación, para así tener el contenido mucho mejor estructurado. En este caso disponemos de 7 tipos de menú:

- **MenuInicial** el cual será el primero que veamos nada más iniciar la aplicación para acceder a los submenús que dispone.
- Después disponemos de **MenuVideo** para cuando queremos reproducir la película o el trailer.
- También he creado un **MenuAudio** para escoger el tipo de audio con el cual queremos escuchar el video.
- También disponemos de un nuevo **Menu Extras** para elegir entre visualizar la galería fotográfica, el trailer, visualizar los eventos o mirar los archivos que hemos descargado.
- Además he hecho una clase **Eventos** la cual nos permite visualizar por pantalla un texto con eventos actualizados que han ido surgiendo desde la salida de la película al público.
- Para ello se conecta al servidor para descargar la última versión y comprueba si está actualizada para más tarde mostrarnos por pantalla los eventos.
- Luego he creado un menú de fotos para poder visualizar la galería de imágenes y también ver la foto seleccionada a tamaño original.
- También se dispone de un menú de archivos descargados, por el cual podemos ver la lista de archivos que hemos descargado a través de la aplicación y además permite visualizar al cliente el archivo que haya seleccionado para ver su contenido.
- Finalmente el último menú que he creado es **MenuConexiónCliente**. He hecho que este menú se encargue de realizar la conexión inicial con el servidor y que delegue las
peticiones que vaya realizando el cliente a TratamientoConexion, ya sea para ver un subdirectorio o descargar un archivo. Como podemos ver TratamientoConexion ha sido rediseñada y realiza todas las tareas que pide el usuario al servidor a través de MenuConexionCliente.

La última clase que he creado ha sido un DescargaFichero. Es un hilo el cual se encarga de recibir el archivo que le envía el servidor. Este hilo trabaja paralelo al refresco de la pantalla para poder ver el % de descarga del archivo en todo momento.

Asimismo, se añaden atributos y métodos que se han visto necesarios a lo largo de la fase de diseño como podemos ver en el siguiente diagrama de clases.

En la siguiente página podemos ver el diagrama clases de la aplicación final:
Programación en red para Blu-ray

Fig 45. Diagrama de clases de la aplicación
5. Implementación

En este apartado explicaré los detalles de programación de la aplicación realizada.

5.1 Menú inicial

- Funcionamiento

Cuando iniciamos la ejecución de la aplicación, disponemos de un atributo (entero) que nos comprueba en qué estado de la aplicación nos encontramos. Dependiendo de eso inicialmente lo que hará será dibujarnos la pantalla del menú inicial.

Las opciones disponibles que tenemos en este menú vienen contenidas en un vector. Además disponemos de otro atributo (entero) el cual nos indica en qué opción estamos situados en ese momento. Gracias a ese atributo cuando se dibuja la pantalla nos muestra sobre qué opción estamos ya que nos aparece un sombreado debajo de ella.

En el momento que seleccionamos una de las opciones disponibles del menú inicial, el atributo que nos indicaba el estado de la aplicación toma otro valor, con lo cual hace que se dibuje por pantalla uno de los submenús que hayamos elegido.
Este es el método encargado de ello:

```java
/* Método que actualiza el menú donde nos encontramos después de haber seleccionado una de las opciones del menú inicial */
public void verSubMenu(){

    if(menuInicial[opcionMenuActual].equals("REPRODUCIR PELÍCULA")){
        MenusFeature.instance.menu = 3;
    }

    else if(menuInicial[opcionMenuActual].equals("DESCARGA DE CONTENIDOS")){
        MenusFeature.instance.menu = 1;
        opcionMenuActual=0;
    }

    else if(menuInicial[opcionMenuActual].equals("AUDIOS")){
        MenusFeature.instance.menu = 2;
        opcionMenuActual=0;
    }

    else if(menuInicial[opcionMenuActual].equals("EXTRAS")){
        MenusFeature.instance.menu = 3;
        opcionMenuActual=0;
    }
}
```
5.2 Opciones de reproducción de video

-Funcionamiento

Las opciones disponibles que tenemos en el menú de videos vienen contenidas en un vector.

Cuando seleccionamos una de las opciones disponibles, disponemos de un atributo (entero) que guarda la posición del vector que ha sido elegida y dependiendo de eso te ejecuta el play, pause o el stop.

A continuación podemos ver el método que se encarga de reproducir cualquier opción de video:

```java
//Método que sirve para reproducir, pausar o detener un video.
//(PRE) Ninguna.
public void verVideo(){
    PlaybackManager pm=new PlaybackManager();

    if(menuPelícula[opciónMenuActual].equals("PLAY")){
        peliActiva=true;
        MenuFeature.instance.menuVisible=false;

        try{
            if(trailer){
                video=2;
            }
            else{
                video=0;
            }
            pm.initialise(video, new PlaybackListener() {
                public void onMediaStart() {
                }

                public void onMediaEnd() {
                    peliActiva=false;
                }
            });
        }catch{...}
    }
    else{
        System.out.println("Error en el Menú");
    }
}
```

Página | 98
Programación en red para Blu-ray

MenusFeature.instance.menuVisible=true;
opcionMenuActual=0;
if(trailer==true){
    trailer=false;
    error=false;
    MenusFeature.instance.menu=3;
    MenusFeature.instance.submenu=0;
}
RepetirEscena();
}
}

} catch (Exception ex) {
    MenusFeature.instance.menuVisible=true;
peliactiva=false;
error=true;
}

}
else if(menuPelicula[opcionMenuActual].equals("PAUSE")){

    if(peliActiva==true){
        if(pause){
            pm.pauseOn();
        }
        else{
            MenusFeature.instance.menuVisible=false;
            pm.pauseOff();
        }
        pause=!pause;
    }
}
else if(menuPelicula[opcionMenuActual].equals("STOP")){

    if(peliActiva==true){
        try{
            pm.initializeApp(1, new PlaybackListener() {

                public void onMediaStart() {
                }

                public void onMediaEnd() {
                }
            });
        }
        catch (Exception ex) {
        }
    }
    peliActiva=false;
pause=true;
if(trailer==true){
    trailer=false;
A la hora de reproducir el video, disponemos de un atributo (booleano) que nos indica si hay un error en el montaje del Blu-ray y no se puede reproducir el video que contiene para mostrar un mensaje de error al usuario.

Disponemos también de otro atributo (booleano) que le sirve a la aplicación para saber si tiene que mostrar el menú de opciones de reproducción o no, ya que cuando se reproduce la película no se quiere que se esté visualizando el menú.

También disponemos un atributo (booleano) el cual nos indica si el video que hay que reproducir es el video de la película o el del trailer que viene incorporado. En ese momento dependiendo del resultado tenemos un atributo (entero) el cual recibe un valor para que al ejecutar el video se visualice el seleccionado.

Finalmente disponemos de un último atributo (booleano) que sirve para que sepa la aplicación que hay un video activo reproduciéndose. De esta manera le permite a la aplicación poder ejecutar las opciones de pause y stop que dispone el menú de video o cambiar el audio con el que se está reproduciéndose la película.

Para pausar o dejar de pausar un video primero se comprueba el atributo que nos decía si había un video reproduciéndose. Una vez hecho esto tenemos un atributo (booleano)
Programación en red para Blu-ray

que nos indica si el video está en pause o no en ese momento para ejecutar la acción adecuada.

Cuando se finaliza la reproducción del video o simplemente queremos detener la reproducción de este se ejecuta este método:

```java
/*Método que sirve para visualizar el video de los menús después de visualizar un video.*
public void RepetirEscena(){
    try{
        PlaybackManager.initialise(i, new PlaybackListener() {

            public void onMediaStart() {
            }

            public void onMediaEnd() {
                RepetirEscena();
            }
        });

    } catch (Exception ex) {
    }
}
```

Con este método lo que se hace es que una vez finalizado la reproducción del video de la película o simplemente que lo queremos detener, hace ejecutar que comience la reproducción del video que se visualiza de fondo en los menús. Como el video de fondo de los menús tiene un tiempo de reproducción una vez que finaliza su reproducción se vuelve a ejecutar este método haciendo como un bucle infinito hasta que se decida reproducir la película o el trailer que disponemos.
5.3 Escoger el modo de audio

-Funcionamiento
Las opciones disponibles que tenemos en el menú de videos vienen contenidas en un vector.

Cuando seleccionamos una de las opciones disponibles, disponemos de un atributo (entero) que guarda la posición del vector que ha sido elegida y dependiendo de eso te ejecuta el modo de audio “Estereo PCM” o el “5.1 Dolby Digital” (por defecto el audio que se ejecuta al reproducir video es el “Estereo PCM”).

A continuación podemos ver el método que se encarga de cambiar el modo de audio del video que se está reproduciendo:

```java
/*Método que sirve para cambiar el modo de audio con el cual escuchamos el video que estamos reproduciendo.*/
//{PRE} Se tiene que estar reproduciendo la película o el trailer.
public void elegirAudio (boolean peliActiva){
    
    if(peliActiva){
        PlaybackManager pm=new PlaybackManager();

        if(menuAudio[opcionMenuActual].equals("ESTEREO PCM")){
            pm.setAudio (1);
        }
        else if(menuAudio[opcionMenuActual].equals("5.1 DOLEY DIGITAL")){
            pm.setAudio (2);
        }

        MenuFeature.instance.menuVisible=false;
        MenuFeature.instance.menu=5;
        opcionMenuActual=0;
    }
}
//{POST} Se cambia el modo de audio al escogido.
```
Como se puede ver recibe el método un atributo (booleano) que nos indica si se está reproduciendo un video. Este atributo sirve para que cuando queramos cambiar el audio cuando no se esta ejecutando ningún video no realice nada.
Finalmente cuando cambiamos el modo de audio disponemos de un atributo (booleano) que sirve para que el menú deje de estar visible y se siga viendo la película.
5.4 Menú extras

-Funcionamiento
Las opciones disponibles que tenemos en el menú de videos vienen contenidas en un vector.

Cuando seleccionamos una de las opciones disponibles, disponemos de un atributo (entero) que guarda la posición del vector que ha sido elegida y dependiendo de eso te ejecuta entre visualizar la galería de imágenes, ver el trailer, mirar los eventos de la película o ver la lista con los archivos que hemos descargado.

En el momento que seleccionamos una de las opciones disponibles del menú de extras, el atributo que nos indicaba el estado de la aplicación toma otro valor, con lo cual hace que se dibuje por pantalla uno de los submenús que hayamos elegido.

Este es el método encargado de ello:

```java
//Método que sirve para visualizar unas de las opciones del menú de extras.
public void verExtra(){
    MenusFeature.instance.gfoto=true;
    error=false;
    ad.existenArchivos=false;

    if(opcionMenuActual==0){
        MenusFeature.instance.submenu=1;
    }

    if(opcionMenuActual==1){
        MenusFeature.instance.menuVisible=false;
        MenusFeature.instance.menu=5;
        MenusFeature.instance.mp.trailer=true;
        MenusFeature.instance.mp.verVideo();
    }

    if(opcionMenuActual==2){
        MenusFeature.instance.submenu=2;
        e.verEventos(MenusFeature.instance.cl);
    }
}```
if(opcionMenuActual==3)
{
    ed.cargarArchivos();
}
opcionesMenuActual=0;
}
5.5 Galería de imágenes

-Funcionamiento

Las imágenes disponibles que disponemos en la galería fotográfica vienen contenidas en un vector que se cargan en memoria nada más inicializar la aplicación.

Para visualizar la galería de imágenes se hace una preselección de cinco imágenes del total que hay en ese vector (que serán las que se puedan visualizar) y se visualizan por pantalla.

Si en ese vector disponemos de 10 imágenes, el método que se encarga de dibujar por pantalla, cogerá 5 fotografías. Si el vector tiene de 0…9 posiciones, solo dibujará al principio de la 0…4.

Para ello disponemos de dos atributos (enteros) que nos marcan el inicio y el final de un subvector imaginario dentro del vector que contiene las imágenes que nos sirve de referencia para saber cuales imágenes se deben de mostrar por pantalla en cada momento.

Si tenemos dos atributos llamados “inicio” y “fin” uno tomará por valor cero y el otro cuatro para que apunte al principio y al final de ese subvector imaginario como podemos ver a continuación.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>inicio</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>fin</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Si el usuario tiene marcada la última fotografía de ese subvector y quiere ver la siguiente imagen, estos dos atributos aumentaran su valor en uno con lo que se situaran en esta posición como se ve en la siguiente imagen:
De esta manera siempre se ven previsualizadas cinco imágenes a la vez por pantalla.

Para saber si hay más imágenes para visualizar, disponemos a los laterales de dos flechas para que el usuario sepa que existen más fotografías para mirar. En el momento que no haya esa flecha, desaparece con lo cual el usuario se percata que no hay más imágenes. Esto se consigue teniendo un atributo (entero) el cual sepa el tamaño del vector donde están todas las imágenes y comparándolo con el atributo que marca la posición final del subvector. Para conseguir el mismo efecto para cuando estemos en la primera imagen del vector se consigue comparando el atributo que te marca el inicio del subvector con cero para que así desaparezca la flecha que te marca que aún no estas en el inicio de la galería de imágenes.

En la siguiente imagen podemos ver el método que dibuja la galería fotográfica:

```java
//Método que dibuja por pantalla la galería fotográfica.
//(P) Ninguna.
public void PaintVisualizarGaleria(BluGraphics dc){
    dc.SetColor(new Color(80, 80, 80, 80));
    if(flechaSel1==true & & tamano>5 & & finSubVector!=tamano){
        FontManager.setFontSmallSel1.drawString(dc, ">", 1740, 640);
        flechaSel1=false;
    }
    if(opcionMenuActual==0 & & tamano>5 & & opcionMenuActual<tamano-1 & &
        finSubVector!=tamano){
        FontManager.setFontSmallSel1.drawString(dc, ">", 1740, 640);
    }
    if(flechaSel==true & & inicioSubVector==0){
        FontManager.setFontSmallSel1.drawString(dc, "<", 100, 640);
        flechaSel=false;
    }
    if(inicioSubVector==0){
        FontManager.setFontSmallSel1.drawString(dc, "<", 100, 640);
    }
    int i=inicioSubVector;
    int x = 0;
    dc.FillRect(210+pxFotocurrent, 490, 300, 300);
}
```
Programación en red para Blu-ray

```java
while (i < finSubVector && i<tamano) {
    dc.drawImage(imagen[i], x+260, 540);
    x=x+300;
    i++;
}

// (POST) Se visualiza la galería de imágenes.
```

Después para ir cambiando entre la galería fotográfica y la visualización de la imagen elegida a tamaño original utilizamos un atributo (entero) y otro atributo (booleano) que le permite a la aplicación saber que debe mostrar por pantalla.

Para visualizar la imagen a tamaño original disponemos de otro vector con todas las imágenes de la galería fotográfica pero a tamaño original. Disponemos de un atributo (entero) que apunta la posición actual sobre la que nos encontramos de la galería fotográfica, por lo que al momento de querer visualizar la imagen seleccionada a tamaño original se mira el valor de ese atributo con el vector y se dibuja por pantalla.

En la siguiente imagen podemos ver el método que dibuja la imagen seleccionada a tamaño original:

```java
// Método que dibuja por pantalla la imagen seleccionada a tamaño original.
// (PRE) Ninguna.
public void PaintImageAmpliada(BluGraphic dc) {
    int ancho;
    int alto;
    anchor.imagengrande[opc1侯].getWid
    ancho-imagengrande[opc1侯].getWid
    ancho-1920/2-ancho/2;
    alto-1080/2-alto/2;
    dc.setColor(new Color(80, 80, 80, 80));
    dc.fillRect(ancho-100, alto-100, imagengrande[opc1侯].getWid
    imagengrande[opc1侯].getWid
    dc.drawImage(imagengrande[opc1侯], ancho, alto);
}

// (POST) Se visualiza la imagen seleccionada a tamaño original.
```
5.6 Eventos

-Funcionamiento
Para visualizar la última versión del archivo de texto de eventos el cliente se tiene que conectar al servidor. Este paso lo comentaré más adelante cuando explique la descarga de contenidos ya que el procedimiento es el mismo.

Una vez que la conexión con el servidor está realizada, el cliente envía un atributo (entero). Le envía un valor que siempre es el mismo cuando lo que quiere el usuario es descargar el archivo actualizado de eventos. Esto es lo que realiza el método en cuestión:

```java
//Método que sirve para pedir el .txt de eventos del servidor.
public void peticionEvento(MensajeConexionCliente c){
    try{
        c.out.writeInt(999);
        MenuFeature.instance.cl.v.clear();
    } catch (Exception evt){
    }
}
```

Una vez hecho esto el servidor recibe la petición y la guardará. Este tratamiento vendrá explicado más adelante cuando explique la descarga de contenidos.

Después se tratará la petición. Para ello se creará un flujo de entrada para leer los datos del archivo de eventos y así poder enviar estos datos por el flujo de salida hacia el cliente. Una vez hecho esto se cierra el flujo de datos por parte del servidor.
En la siguiente imagen podemos ver el método que se encarga de ello:

```java
// Método que envia el .txt más actualizado de eventos que dispone el servidor al cliente. 
//(PRE) El cliente está conectado al servidor. 
public void enviarEventos(DataOutputStream out) throws IOException {
    int len;
    File f = new File("Eventos/Eventos.txt");
    FileInputStream in = new FileInputStream(f);
    byte[] buf = new byte[1024];
    out.writeUTF("Eventos.txt");
    long l = f.length();
    out.writeLong(l);
    while((len = in.read(buf)) > 0) {
        out.write(buf, 0, len);
    }
    in.close();
}
//(POST) El archivo se envía correctamente al cliente.
```

A la vez, la aplicación cliente crea también otros dos flujos. Uno de entrada para ir recibiendo los datos que le está enviando el servidor del archivo de eventos y otro flujo de salida para guardar los datos en el disco duro. Una vez hecho esto se cierra el flujo.
Aquí podemos ver como el archivo que va descargando el cliente se va guardando en el disco duro:

```java
//Método que descarga y guarda el .txt de eventos del servidor.
public void guardarEvento(MenuConexcionCliente c){

    try{
        String nomfichero = c.in.readUTF();
        File f = new File("C:/Descargas/" + nomfichero);
        out = new FileOutputStream(f);
        long tamanofichero = c.in.readLong();
        byte[] b = new byte[1024];
        int len;
        long bytesleidos = 0;

        while (bytesleidos < tamanofichero) {
            len = c.in.read(b);
            out.write(b, 0, len);
            bytesleidos = bytesleidos + len;
        }
        out.close();
    }

    Una vez finalizada la descarga se cierra la conexión con el servidor. Para ello cerramos el socket como podemos ver a continuación:

    //Método que cierra la conexión con el servidor.
    //El usuario se desconecta del servidor.
    public void cerrarConexcion(){

        try{
            socket.close();
        }
        catch (Exception e){
        }
    }
```
Una vez hecho esto se comprueba si la versión descargada está más actualizada que la que disponemos:

```java
/*Método que comprueba si el .txt descargado está más actualizado que el .txt que dispone el cliente*/
//{PRE} Ninguna.
public void EventoActualizado(){
    FileInputStream entrada1;
    FileInputStream entrada2;
    String linea1="A";
    String linea2="B";
    boolean linealiguales=true;
    Vector aux=new Vector();
    Vector aux2=new Vector();
    try{
        if(!f3.exists()){
            if(f2.exists() && f.exists()){
                entrada1 = new FileInputStream(f);
                entrada2 = new FileInputStream(f2);

                aux=obtenerTxt(entrada1);
                aux2=obtenerTxt(entrada2);

                int i=0;
            }
        }
    }
```

Para ello disponemos de dos buffer que van leyendo línea a línea los dos archivos de texto y se guarda cada uno de ellos en un vector. En cada posición de este se guarda una línea de texto:
Después comienza la comprobación para verificar si la versión descargada tiene alguna modificación con la que nosotros disponemos. Dependiendo de esto se visualizará la versión descargada o la que disponemos en la aplicación. Para ello se va comparando cada posición de los dos vectores, en el momento que no son iguales descubrimos que versión de archivos de eventos visualizar.
Programación en red para Blu-ray

En la siguiente imagen podemos ver el método que se encarga de ello:

```java
//Método que comprueba que versión de los .txt está más actualizado.
public boolean comprobarActualizacion(Vector aux, Vector aux2){
    boolean lineasIgual=false;
    String linea="\n";
    String linea2="\n";
    int i=0;

    while(i<aux.size() && !linea.equals("\n") && !linea2.equals("\n")
        && lineasIguals){
        linea=(String)aux.get(i);
        linea2=(String)aux2.get(i);
        if(!linea.equals(linea2))
            lineasIgual=false;
        i++;
    }

    return lineasIgual;
}
```

A continuación seguimos con el método principal encargado de hacer la comprobación. Si son iguales los dos archivos te elimina la versión que se ha descargado. En el caso que no sean iguales. Se borra el archivo .txt antiguo y se renombra el archivo descargado con el nombre del archivo .txt original. También tenemos en cuenta el caso que la aplicación por si sola no disponga del .txt de eventos. Entonces hasta que no se descargue una primera versión no se podrá visualizar.
lineaIgual=comprobarActualizacion(aux, aux2);

if(lineaIgual){
    entrada2.close();
    f2.delete();
} else{
    actualizado=true;
    f2.renameTo(f3);
    entrada2.close();
    f2.delete();
}

else if(!f.exists() && f2.exists()){
    actualizado=true;
    f2.renameTo(f3);
    f2.delete();
}
else{
    if(f2.exists()){

        entrada2 = new FileInputStream(f3);
        entrada2 = new FileInputStream(f2);

        aux=obtenerTxt(entrada2);
        aux2=obtenerTxt(entrada2);

        lineaIgual=comprobarActualizacion(aux, aux2);

        if(lineaIgual){
            entrada2.close();
            f2.delete();
        } else{
            actualizado=true;
            f3.delete();
            f2.renameTo(f3);
            entrada2.close();
            f2.delete();
        }
    }
}

}

} catch (Exception evt){
    MenuFeature.instance.submenu=0;
    MenuFeature.instance.me.error=true;
    f2.delete();
}
Programación en red para Blu-ray

guardarTextoElegido();
}

// (POST) Se guarda la versión del archivo de eventos más actualizada.

Finally se coge y se guarda en memoria el archivo de texto a visualizar como podemos ver a continuación:

/*Método que guarda cada línea de texto del archivo elegido a visualizar en una posición del vector.*/
public void guardarTextoElegido()
{
    FileInputStream fis;
    try{
        if(re.exists()){  
            fis = new FileInputStream(re); 
        }
        else{
            fis = new FileInputStream(f); 
        }
        txt = obtenerTxt(fis);
    }
    catch (Exception evt){
        MenuFeature.instance.me.error=true;
        MenuFeature.instance.submenu=0;
    }
}
5.7 Archivos descargados

-Funcionamiento

Para visualizar la lista de archivos que hemos descargado con la aplicación, disponemos de una array (String), en la cual guardamos el nombre de los archivos cogiendo la ruta donde están guardados los archivos descargados.

A continuación podemos ver el método que se encarga de ello:

```java
/*Método que sirve para guardar el listado de archivo que hay descargados en
la carpeta Descargas*/
//(PRE) Ninguna.
    public void cargarArchivos(){
        File f=new File("C:/Descargas");
        listaArchivos = f.list();
        int i = listaArchivos.length;
        if(i==0){
            existenArchivos=false;
        }else{
            existenArchivos=true;
        }
        MenuFeature.instance.submenu=3;
    }

    // (POST) Se guarda el nombre de los archivos descargados en el array.
```

Como podemos observar, hay que comprobar si en la carpeta de las descargas hay archivos, para que en el caso que no haya se nos muestre un mensaje con ello.

Después cuando seleccionamos un archivo para visualizarlo, primeramente en un atributo (String) guardamos la extensión del archivo del array donde se guarda el nombre de los archivos de la carpeta “Descargas” para saber que tipo de fichero es y así que la aplicación pueda proceder a cargarlo.
A continuación podemos ver el método encargado de ello:

```java
//Método que carga el archivo seleccionado para su posterior visualización. */
//{(PRX) Ninguna.
public void verArchivo(){
    int i = listaArchivos[opcionMenuActual].length();
    String aux = listaArchivos[opcionMenuActual].substring(i-3);
    if(aux.equals(".txt")){
        FileInputStream fi;
        try{
            fi = new FileInputStream(new File("C:/Descargas/" + listaArchivos[opcionMenuActual]));
            txt=MenusFeature.instance.me.e.obtenerTxt(fi);
            MenusFeature.instance.submenu=4;
        }
        catch(Exception e){
            errorArchivo=true;
            MenusFeature.instance.submenu=0;
        }
    } else if(aux.equals(".jpg") || aux.equals(".png")){
        MenusFeature.instance.submenu=5;
        cargarImagen();
    }
}

//{POST} Se carga el archivo.
```

Si queremos visualizar un archivo de texto, utilizaremos para cargar el fichero en memoria un método de la clase Eventos que realiza esa función que ya viene explicado anteriormente cuando queremos visualizar el archivo de eventos. Si en cambio queremos ver una imagen, antes de poder visualizarse hay que cargarla en memoria.

Para ello disponemos de este método:

```java
//Método que carga la imagen que queremos visualizar.
//{(PRX) Ninguna.
public void cargarImagen(){
    File f=new File("C:/Descargas/" + listaArchivos[opcionMenuActual]);
    if(f.exists()){image=ImageManager.loadImage("C:/Descargas/" + listaArchivos[opcionMenuActual]);
    }
    else{
        errorArchivo=true;
        MenusFeature.instance.submenu=0;
    }

    //{(POST} Se carga la imagen en memoria.
```
5.8 Descarga de contenidos

-Funcionamiento

Para visualizar el contenido del servidor, primero hay que conectarse al servidor. Para ello se crea un socket con la dirección ip del servidor (esta se lee de un archivo) y el puerto por el cual el cliente se conectara a él. Una vez hecho esto se crean dos flujos. Uno para enviar datos y otro para recibirllos.

A continuación podemos ver el método que se encarga de ello:

```java
//Método que realiza la conexión inicial con el servidor.
//(PRE) Ninguna.
public void conexionInicial(){
    String ip="";
    try{
        FileInputStream f = new FileInputStream("resources/ip.txt");
        byte b[] = new byte[32];
        while(f.read(b)!=-1){
            ip=new String(b);
        }
    }catch (Exception e){
        MenuFeature.instance.menu=0;
        MenuFeature.instance.error=true;
    }
    try{
        socket = new Socket(ip, 9351);
        in = new DataInputStream (socket.getInputStream());
        out = new DataOutputStream (socket.getOutputStream());
        if(v.size()>0){
            MenuFeature.instance.cl.to.primerConexion=false;
        }
    }
    catch (Exception e){
        if(MenuFeature.instance.menu==3){
            MenuFeature.instance.menu=0;
            MenuFeature.instance.error=true;
            conectando=true;
        }
    }
    //{POST} La conexión se ha realizado correctamente.
```
Desde el lado servidor lo que realiza nada más ejecutarse es crear otro socket abriendo el puerto por el cual se van a conectar los clientes y del cual recibirá las peticiones de estos.

Una vez reciba la petición de conexión de un cliente, se acepta y se crea un hilo el cual tratará las peticiones que realice ese cliente. Para cada cliente habrá creado un hilo de manera que se permitan tratar múltiples peticiones de distintos clientes a la vez.

Además por pantalla muestra en el servidor quien se ha conectado, ya que nos muestra la dirección ip del cliente.

A continuación podemos ver como el servidor recibe las conexiones:

```java
public class Servidor{

    ServerSocket serv = new ServerSocket(9351);

    // *Se encarga de aceptar las peticiones de conexión de los clientes y de crear un hilo independiente para cada uno de ellos y así tratar las peticiones que vayan realizando*/
    public Servidor () throws IOException{
        try{
            while(true){
                Socket clients = serv.accept();
                System.out.println("Cliente:"+clients.get InetAddress()+" se ha conectado");
                HiloServidor hs = new Hilo Servidor(clients);
                hs.start();
            }
        } catch (Exception e){
            serv.close();
            System.out.println("Servidor finalizado");
        }
    }  
}
```

Después de crear el hilo para esa conexión, se crean los flujos de entrada y salida de datos.

Después se crea un vector (String) para almacenar el nombre de los directorios o archivos que hay en la primera capa de la carpeta donde el servidor tiene el contenido.
Para que el usuario sepa que ya no tiene que seguir a la espera de recibir más datos, el servidor le envía una palabra concreta y de esta manera el cliente pueda seguir con la ejecución de la aplicación.

Una vez hecho esto se comienza a enviar estos datos al cliente como podemos ver a continuación:

```java
// Metodo que sirve para recibir y transmitir las peticiones del cliente.
public void run() {
    try {
        PeticionesCliente lc = new PeticionesCliente();
        String s[] = lc.lectura();
        int len = s.length;
        for (int i = 0; i < len; i++) {
            out.writeUTF(s[i]);
        }
        out.writeUTF("hola");
        while (true) {
            lc.recibirPeticion(in);
            lc.transmitirPeticion(out);
        }
    } catch (Exception e) {
        System.out.println("El cliente=\"+socket.getInetAddress()+\" se ha desconectado");
    }
}
```

Una vez hecho esto se queda a la espera de recibir una petición por parte de ese cliente.

Por otra parte desde la aplicación cliente se recibe el nombre de los archivos y los directorios del servidor y los guarda en un vector (String) para mostrárnoslos por pantalla posteriormente.
Aquí podemos ver el método encargado de ello:

```java
/**Método que recibe los datos iniciales para visualizar el contenido del menú descarga de contenidos.*/
//(PRE) Estar conectado al servidor.
public void recibirDatosIniciales()
{
    String s="0";
    try{
        while(s!=null){
            s = in.readUTF();
            if(s.equals("host") ||
                s=null;
            } else {
                v.addElement(s);
            }
        }
    }
    catch (Exception e){
        if(MenusFeature.instance.menu!=3){
            MenusFeature.instance.menu=0;
            MenusFeature.instance.error=true;
            conectando=true;
            v.clear();
            ve.clear();
            cerrarConexion();
        }
    }
    //*(POST) Se ha recibido correctamente el nombre de los archivos y carpetas del servidor.*/
```

Una vez hecho esto el método encargado de dibujar por pantalla el menú de descarga de contenidos leerá ese vector e irá mostrando por pantalla las opciones disponibles.
Programación en red para Blu-ray

Cuando el usuario quiere visualizar el contenido de un directorio del servidor, primeramente lo que se realiza es guardar la opción del menú que ha marcado en un vector (enteros). Por lo tanto en cada posición de este vector se guarda la opción que ha marcado el usuario en cada subnivel de los directorios a partir de la raíz de donde se visualiza el contenido del servidor.

Ejemplo:

En la raíz del servidor disponemos de tres carpetas llamadas X, Y, Z. El usuario pulsa Y. Por lo tanto en el vector de enteros se guarda un 1.

```
1   
```

En el subdirectorio de Y supongamos que tenemos otras tres carpetas llamadas A, B, C. Si queremos visualizar el contenido de A se guardará en ese vector a continuación un 0.

```
1   0
```

Finalmente si en el subdirectorio de A supongamos que tenemos otras tres carpetas llamadas D, E, F. Si queremos visualizar el contenido de F se guardará en ese vector a continuación un 2.

```
1   0   2
```

Además si queremos volver al subdirectorio anterior, simplemente se borra el valor de la última posición del vector y se le envía ese vector al servidor.
El método de encargar de guardar la opción seleccionada por el cliente es el que viene a continuación:

```java
/*Método que guarda la opción sobre la cual el cliente quiere recibir información del servidor.*/
public void AnadirOpcion(MenuConexionCliente c, Integer posCarpetaElegida) throws IOException {
    c.ve.addElement(posCarpetaElegida);
}
```

Una vez hecho esto se coge y se envían los elementos de este vector (enteros) al servidor. Como la aplicación cliente no sabe determinar si lo que ha seleccionado el usuario ha sido visualizar un nuevo directorio o descargar un archivo (aunque el usuario sí lo sabe distinguir), primeramente le envía estos datos para que el servidor le de una respuesta y así la aplicación cliente haga los requerimientos necesarios.

Aquí podemos ver el método que se encarga de enviar los datos al servidor:

```java
/*Método que envía la petición para visualizar el directorio o descargar el archivo que haya seleccionado el usuario.*/
public void EnviarPeticion(MenuConexionCliente c) throws IOException {
    try {
        int n = c.ve.size();
        int aux;
        Integer inte;
        for (int j = 0; j < n; j++) {
            inte = (Integer)c.ve.get(j);
            aux = inte.intValue();
            c.out.writeInt(aux);
        }
        c.out.writeInt(111);
        c.ve.remove(n-1);
    }
    catch (Exception evt) {
        MenusFeature.instance.menu = 0;
        MenusFeature.instance.error = true;
        c.conectando = true;
        c.v.clear();
        c.ve.clear();
    }
}
```
Para que el servidor sepa que no tiene que estar a la espera de recibir más datos se le envía finalmente un entero “x” con el cual finalizará de recibir datos.

A continuación podemos ver el método que recibe las peticiones del cliente:

```java
//Método que guarda la petición recibida por parte de un cliente.
//{PRE} Tener conexión con el cliente.
public void RecibirPeticion(DataInputStream buff) throws IOException{
    int i=0;
    try{
        while(i!=-5){
            i = buff.readInt();
            if(i!=111 && i!=999){
                directorioCliente.addElement(new Integer(i));
            } else{
                directorioCliente.addElement(new Integer(i));
                i=-5;
            }
        }
    }
    catch (Exception e){
    }
    //{POST} Se guardan los datos recibidos del cliente correctamente.
}
```

Como podemos ver cada dato recibido se guarda en un vector.

Después el servidor comienza a tramitar la petición. Para ello empieza a leer el contenido de ese vector donde se han guardado los datos del cliente. Con ello el servidor llega al directorio elegido por el cliente para así después ejecutar el envío de la carpeta seleccionada:
Programación en red para Blu-ray

//Método que tramita la petición recibida por el cliente.
public void TramitarPeticion(DataOutputStream out) throws IOException {
    int i = directorioCliente.size() - 1;
    int n = 0;
    File f[] = fi;
    int j = 0;

    Integer inte = (Integer) directorioCliente.lastElement();
    int m = inte.intValue();

    if (m == 111) {
        while (j < i) {
            Integer inte2 = (Integer) directorioCliente.get(j);
            n = inte2.intValue();
            if (j + 1 == i) {
                f = f[n].listFiles();
            }
        }
        j++;
    }
    directorioCliente.clear();

    String s[];
    int len;

    if (i == 0) {
        s = lectura();
        len = s.length;
        i = 0;

        while (i < len) {
            out.writeUTF(s[i]);
            i++;
        }
        out.writeUTF("hola");
    }
    else {
        if (f[n].isDirectory()) {
            enviarDirectorio(f, n, out);
        }
    }
}
Programación en red para Blu-ray

A continuación podemos ver el método encargado de enviar el contenido del directorio al cliente:

```java
//Método que envía el directorio que ha demandado el cliente.
//{PRE} Exista el directorio y haya conexión entre el servidor y el cliente.
public void enviarDirectorio(File f, int n, DataOutputStream out) throws IOException{
    String s[];
    int len;
    s = f[n].list();
    len = s.length;
    int i=0;
    out.writeUTF("Directorio");
    while(i<len){
        out.writeUTF(s[i]);
        i++;
    }
    out.writeUTF("hola");
}
//{POST} Se envía correctamente al cliente el contenido del directorio elegido.
```

Como podemos ver primeramente le manda un String a la aplicación cliente para que esta sepa que la opción seleccionada por el usuario era un directorio y no un archivo. De esta manera la aplicación cliente ejecuta el método para recibir el contenido del directorio.

Después la aplicación servidor comienza a enviarle el contenido del directorio al cliente.
Programación en red para Blu-ray

A continuación podemos ver el método que guarda los datos enviados por el servidor:

```java
//Método que guarda el nuevo directorio recibido que ha elegido el cliente
//Primero se elimina todo el contenido que había guardado en el vector (String) y se comienzan a añadir el nombre de los elementos que contiene la carpeta que había seleccionado el usuario.
//Además se añade en el vector (enteros) el nuevo directorio después de que el usuario hiciese la petición para saber si la opción elegida por el usuario era un archivo o una carpeta.
//Una vez hecho esto el método encargado de dibujar la pantalla del menú de descarga de contenidos leerá el vector (String) y mostrará al usuario el contenido del directorio que había seleccionado desde un principio.
```
Si por el contrario la opción elegida hubiera sido un archivo, todo el procedimiento entre el cliente y el servidor hubiera sido el mismo. Solo cambiaría al final que en vez de ejecutarse el método para enviar el directorio, se hubiera ejecutado el método para enviar un archivo desde la aplicación servidor. Además de que al cliente le hubiera llegado otro dato diferente con el cual en vez de ejecutar el método para añadir un directorio, haría el procedimiento para la descarga del archivo.

A continuación podemos ver el método encargado de enviar el archivo al cliente por parte del servidor:

```java
//Método que envía el archivo demandado por el cliente.
// (PRE) Tiene que haber conexión entre el servidor y el cliente.
public void enviarArchivo(File file[], int n, DataOutputStream out) throws IOException{
    try{
        byte[] buf = new byte[1024];
        out.writeUTF("Archivo");
        String nombreArchivo = file[n].getName();
        out.writeUTF(nombreArchivo);

        long l = file[n].length();
        out.writeLong(l);
        while((l = in.read(buf)) > 0){
            out.write(buf, 0, l);
        }
        in.close();
    }
    catch (Exception e){
    }
}
//(POST) Se envía correctamente el archivo al cliente.
```

Como podemos ver se crea un flujo de entrada para leer el archivo a enviar al cliente. Después se envía al cliente el nombre del archivo a descargar y seguidamente se comienzan a enviar los datos del archivo. Una vez hecho esto se cierra el flujo que habíamos abierto para leer los datos del archivo que estábamos enviando.
Por otra parte la aplicación cliente descarga el archivo a través de este método:

```java
// Método que sirve para descargar el archivo seleccionado del menú descarga de contenidos. */
//(PRE) Existe conexión entre el servidor y el cliente.
public void run() {

    String nomfichero="000099";

    try{

        nomfichero = c.in.readUTF();
        f = new File("C:/Descargas/" + nomfichero);
        out = new FileOutputStream(f);
        long tamanofichero = c.in.readLong();
        byte[] b = new byte[1024];
        int len;
        long bytesleidos = 0;

        while (bytesleidos < tamanofichero) {
            len = c.in.read(b);
            out.write(b, 0, len);
            bytesleidos = bytesleidos + len;
            long p = bytesleidos*100/tamanofichero;
            MenusFeature.estadoDescarga=String.valueOf(p);

        }
        out.close();

    }

    catch (Exception evt){
        try{
            out.close();
        }
        catch (Exception e){
        }

        if(errorDescarga){
            if(!nomfichero.equals("000099")){
                if(f.exists()){f.delete();}
            }
            MenusFeature.instance.wcru=0;
            MenusFeature.instance.error=true;
        }
        else{
            f.delete();
        }
    }

    } // (POST) Se descarga el archivo correctamente.
Programación en red para Blu-ray

Primeramente a través del flujo de entrada de datos de la conexión entre el cliente y servidor recibe el nombre del archivo descargado. Después se crea un flujo de salida donde se guardará el archivo descargado con la ruta indicada y con el nombre del archivo que nos han dado. Una vez hecho esto comenzará la descarga de datos por parte de servidor hasta finalizar. Finalmente una vez que esté completo el fichero, se cierra el flujo de datos de salida por el cual se ha escrito el fichero en el disco duro.

Para realizar esto utilizó un hilo ya que por pantalla se va actualizando constantemente el porcentaje de descarga que lleva el archivo. De esta manera se está ejecutando este método y a la vez el de dibujar por pantalla que se va actualizando en todo momento. Sino lo hubiera implementado así hasta que no se acabara la descarga del archivo no se ejecutaría el método de dibujar por pantalla y entonces directamente se me vería la barra de descarga al 100% y no se vería el progreso de esta.
6. Juego de pruebas

Para realizar el juego de pruebas he decidido crear un módulo para cada clase que he creado quitando algún caso puntual en concreto que comentaré más adelante.

-Del menú inicial debemos comprobar dos cosas:

La primera es que el desplazamiento por las diferentes opciones que dispone el menú inicial a través de las flechas de dirección que dispone el teclado sea el correcto. Para ello hemos creado unos métodos que comprueban que si al producirse este tipo de evento hay unos ciertos atributos que deben tomar unos valores concretos.

Como tenemos un atributo que apunta a la posición del vector que hay seleccionada en ese momento si queremos desplazarnos hacia la izquierda pero estamos en la posición inicial del vector que es cero, ese atributo no cambia de valor y por lo tanto devolverá por pantalla cierto.

Aquí podemos ver el método que lo comprueba:

```java
public boolean left(){
    mf.init();
    mf.mi.left();
    if (mf.mi.opcionMenuActual==0){
        return true;
    }
    else{
        return false;
    }
}
```

A continuación podemos ver el método que se encarga de comprobar si el atributo actualiza la posición si no se encuentra en la posición inicial como en el caso anterior.
Como cambiará de valor devolverá cierto:

```java
public boolean left1(){
    mf.mi.opcionMenuActual=2;
    mf.mi.left();
    if(mf.mi.opcionMenuActual==1){
        return true;
    }
    else{
        return false;
    }
}
```

Para comprobar lo mismo si lo que decidimos es desplazarnos hacia la derecha en los menús disponemos de los siguientes métodos.

Este método hará que el atributo se actualice ya que no se encuentra en la última posición del vector:

```java
public boolean right(){
    mf.mi.opcionMenuActual=1;
    mf.mi.right();
    if(mf.mi.opcionMenuActual==2){
        return true;
    }
    else{
        return false;
    }
}
```

Si el atributo apunta al final del vector no cambiará de valor por lo que tendrá que devolver cierto como podemos ver a continuación:
La segunda comprobación que hay que hacer es si al querer acceder a uno de los menús disponibles en la pantalla inicial se cambia un atributo de valor el cual es el que hace saber al método paint() que es lo que toca dibujar en cada momento.

A continuación podemos ver los métodos que comprueban cada opción del menú. En este caso si todo es correcto devolverá cierto los resultados:

```java
public boolean right1(){
    mf.mi.opcion_Menu_Actual=3;
    mf.mi.right();
    if(mf.mi.opcion_Menu_Actual==3){
        return true;
    }
    else{
        return false;
    }
}
```

```java
public boolean submenu(){
    mf.mi.opcion_Menu_Actual=0;
    mf.mi.ver_Sub_Menu();
    if(mf.menu==5){
        return true;
    }
    else{
        return false;
    }
}
```

```java
public boolean submenu2(){
    mf.mi.opcion_Menu_Actual=1;
    mf.mi.ver_Sub_Menu();
    if(mf.menu==2){
        return true;
    }
    else{
        return false;
    }
}
```

```java
public boolean submenu1(){
    mf.mi.opcion_Menu_Actual=3;
    mf.mi.ver_Sub_Menu();
    if(mf.menu==1){
        return true;
    }
    else{
        return false;
    }
}
```

```java
public boolean submenu3(){
    mf.mi.opcion_Menu_Actual=2;
    mf.mi.ver_Sub_Menu();
    if(mf.menu==3){
        return true;
    }
    else{
        return false;
    }
}
```
-Para comprobar el menú de la película hay que comprobar tres cosas:

La primera es que el desplazamiento por las diferentes opciones que dispone el menú de video a través de las flechas de dirección que dispone el teclado sea el correcto. En este caso obviaremos describir el funcionamiento porque trabaja de la misma manera que con el menú inicial y los métodos para probarlo son iguales.

Lo segundo que hay que mirar es si cuando se está viendo la película y decides regresar hacia atrás para que te aparezca el menú de video y si ya estás en el poder regresar al menú inicial. En el caso que estemos viendo el menú de video y queremos volver atrás, lo que debemos hacer es cambiar el atributo que nos indica en el menú que estamos por el valor del que tiene el menú inicial. Además también comprueba si el atributo que te apunta en la posición del menú donde nos encontramos se reinicia y apuntar a la primera posición del menú para cuando queramos volver acceder a él.

Aquí podemos observar el método que lo comprueba:

```java
public boolean esc(){
    m.menu=5;
    m.mp.peliActiva=false;
    m.mp.opcionMenuActual=2;
    m.mp.esc();
    if (m.menu==0 && m.mp.opcionMenuActual==0){
        return true;
    }
    else{
        return false;
    }
}
```

En el caso que ya estamos reproduciendo un video, tenemos que comprobar que el atributo que se encarga de decir si se ha de mostrar el menú es cierto y que el atributo que apunta a la primera opción disponible en el menú valga uno. Esto es debido a que la opción del “PLAY” desaparece mientras hay un video activo para que no podamos utilizarla.
A continuación podemos ver el método que lo comprueba:

```java
public boolean esc1()
{
    m.menu=5;
    m.mp.peliActiva=true;
    m.menuVisible=false;
    m.mp.opcionMenuActual=2;
    m.mp.esc();
    if(m.menuVisible==true && m.mp.opcionMenuActual==1){
        return true;
    }
    else {
        return false;
    }
}
```

La tercera comprobación que hay que hacer es mirar que las opciones de reproducción de video funcionan perfectamente. Para ello no he podido realizar un juego de pruebas debido a que el lenguaje Java por sí solo no puede reproducir un video ya que para ello hay que utilizar aplicaciones que se dediquen a ello. Entonces para probar que estas opciones funcionaban correctamente tenía que ejecutar mi aplicación con la imagen de Blu-ray que dispongo a través del reproductor que utilizo para visualizar mi aplicación en funcionamiento y poder probar que se reproduce el video, que se puede pausar, quitar la pausa y detener la reproducción de este.

-Para comprobar el menú de audio hay que comprobar tres cosas:

Aquí también debemos comprobar que el desplazamiento por las diferentes opciones que dispone este menú sea el correcto. Como el juego de pruebas funciona también igual en estos métodos que los anteriores, obviaremos el funcionamiento de estos.

También debemos mirar si al intentar regresar al menú inicial desde el menú de audio se modifican los atributos correctamente como en el menú de video.
Como podemos ver la comprobación es prácticamente igual que las anteriores:

```java
public boolean esc() {
    mf.ma.esc();
    mf.ma.opcionMenuActual=1;
    if (mf.menu==0 &amp; mf.ma.opcionMenuActual==0) {
        return true;
    } else {
        return false;
    }
}
```

Para comprobar si hay que cambiar el modo de audio, disponemos de un atributo que nos dice si hay una película reproduciéndose para así acceder a esta función. Por lo tanto si se modifica el atributo que nos indica que estamos en el menú de audio por el del menú de la película y el atributo que nos indica si el menú visible se pone falso significaría se vuelve invisible que es lo que pasa cuando se reproduce una película. Si los atributos mantienen el valor que tenían significa que todo ha ido correcto.

A continuación podemos ver el método que lo mira:

```java
public boolean ponerAudio(){
    mf.ma.elegirAudio(false);
    if (mf.menuVisible==true &amp; mf.menu==2){
        return true;
    } else{
        return false;
    }
}
```

Después si al reproducir un video, decidimos cambiar el modo de audio y comprobar si este se ha modificado correctamente, debemos hacerlo como en el caso que queremos probar que funcionan las opciones de reproducción de video. Se ha de ejecutar la aplicación en el WinDVD con la imagen de Blu-ray que disponemos.
- Para comprobar que funciona el menú de extras miraremos los siguientes detalles:

Lo primero es comprobar que el desplazamiento entre las opciones que dispone el menú extras sea el adecuado. Al tener el mismo funcionamiento que el menú inicial y este ya está comentado obviaré la explicación.

Para comprobar que puede regresar desde el menú de extras hasta el menú inicial, tiene el mismo funcionamiento que el menú de audio. Por lo tanto el juego de pruebas es el mismo que en ese caso.

Para acceder a una de las funcionalidades del menú de extras, disponemos de otro atributo (a parte del que nos indica en el menú que estamos) para saber que opción de extras se está o se tiene que ejecutar en ese momento. Por lo cual dependiendo del valor que le demos ejecutará un extra u otro.

A continuación podemos ver los métodos que comprueban que el funcionamiento de la aplicación en esta parte es el correcto (debe de dar cierto de resultado en todas las comprobaciones):

```java
public boolean extra2() {
    mf.me.opcionMenuActual=2;
    mf.me.varExtra();
    if (mf_submenu==2) {
        return true;
    }
    else {
        return false;
    }
}
```
-Para comprobar que la galería fotográfica funciona correctamente he comprobado varias cosas:

Para ver si se desplaza correctamente entre las diferentes imágenes que dispone la galería he realizado varias pruebas.

Disponemos de un atributo que marca a partir de donde hay que dibujar el sombreado de la opción que tenemos seleccionada. Además tenemos otros dos atributos que nos marcan dentro del vector que contiene las imágenes el inicio y el final de un subvector de este que nos indica que imágenes del total que dispone la galería fotográfica hay que mostrar.
El primer caso es estar en la primera foto de la galería e intentar desplazarme hacia la izquierda para ver que el atributo que apunta la posición donde nos encontramos es el mismo:

```java
public boolean left(){
    mf.init();
    mf.left();
    if(mf.mo.opcionMenuActual==0 && mf.mo.pxFotoActual==0 && mf.mo.inicioSubVector==0){
        return true;
    }
    else{
        return false;
    }
}
```

Después pruebo el caso que no estemos en la primera posición de las imágenes que hay e intentemos seleccionar una imagen que se encuentre a su izquierda como vemos a continuación:

```java
public boolean left1(){
    mf.mo.pxFotoActual=300;
    mf.mo.opcionMenuActual=1;
    mf.left();
    if(mf.mo.opcionMenuActual==0 && mf.mo.pxFotoActual==0){
        return true;
    }
    else{
        return false;
    }
}
```

Ahora pruebo estar en el caso en el que la galería de fotos no se ve la primera imagen e intento acceder a ella para que pueda verse de nuevo en ese subconjunto de fotos que se pueden ver:
Ahora probaré lo mismo desplazándome hacia la derecha en la galería de fotos. Si el atributo que me indica sobre que imagen estoy se ha modificado y también el de la imagen que hay seleccionada en ese momento significará que se ha realizado correctamente como podemos ver a continuación:

```java
public boolean left2(){
    mf.fo.pxFotoActual=0;
    mf.fo.inicioSubVector=1;
    mf.fo.left();
    if(mf.fo.inicioSubVector==0){
        return true;
    }
    else{
        return false;
    }
}
```

A continuación probamos el caso en que tenemos seleccionada la última imagen de la galería de fotos. Con lo cual el atributo que indica la posición en que nos encontramos no variará al igual que el que marca la opción seleccionada de las imágenes previsualizadas de la galería.
A continuación vemos el método encargado de comprobarlo:

```java
public boolean right1(){
    mf.fo.pxFotoActual=1200;
    mf.fo.opcionMenuActual=mf.fo.tamaño-1;
    mf.fo.right();
    if(mf.fo.opcionMenuActual==mf.fo.tamaño && mf.fo.pxFotoActual==1200){
        return true;
    }
    else{
        return false;
    }
}
```

Después el último caso que compruebo es cuando queremos visualizar la siguiente imagen que no se puede visualizar del subvector de imágenes que se muestran por pantalla en ese momento. Por lo tanto los dos atributos que te marcan el inicio y el fin de este subvector incrementarán su valor en uno.

El método que comprueba esto viene a continuación:

```java
public boolean right2(){
    mf.fo.opcionMenuActual=4;
    mf.fo-finSubVector=5;
    mf.fo.right();
    if(mf.fo.inicioSubVector==1 && mf.fo-finSubVector==6){
        return true;
    }
    else{
        return false;
    }
}
```

Para comprobar si estamos visualizando la galería de imágenes o la imagen que hemos seleccionada a tamaño original, tenemos un atributo el cual dependiendo del valor se hace mostrar por pantalla una de las dos cosas.
A continuación mostraremos el caso en que estemos visualizando la imagen a tamaño original:

```java
public boolean esc(){
    mf.gfot=false;
    mf.to.esc();
    if(mf.gfot==true){
        return true;
    }else{
        return false;
    }
}
```

Ahora miraremos el caso para pasar de visualizar la galería de imágenes, a visualizar el menú de extras. Para ello tenemos un atributo que dependiendo del valor muestra por pantalla en este caso el menú de extras como podemos ver a continuación:

```java
public boolean esc1(){
    mf.gfot=true;
    mf.submenu=1;
    mf.to.esc();
    if(mf.submenu==0){
        return true;
    }else{
        return false;
    }
}
```

-Para probar el correcto funcionamiento del módulo de eventos hice varias pruebas:

Para pasar de visualizar la sección de eventos al menú de extras, el juego de pruebas funciona igual que en este último caso explicado en la galería de imágenes por lo que no hace falta explicar el código de pruebas.
Como en este juego de pruebas lo hacemos sin conexión al servidor cuando probemos descargar el archivo de texto de eventos devolverá falso ya que no existirá el archivo descargado:

```java
public boolean descargaEvento(){
    mf.me.e.peticionEvento(mf.c1);
    mf.me.e.guardarEvento(mf.c1);
    if(mf.me.e.f2.exists()){ 
        return true;
    }
    else{ 
        return false;
    }
}
```

Para probar que el método que guarda el texto de un archivo que hay en disco a un vector se hace correctamente, se comprueba con el método del juego de pruebas que tenemos a continuación. Creamos un flujo del cual extraemos una línea de texto del archivo y lo comparamos con la primera posición del vector creado, el cual tiene una línea de texto guardada. Esto se hace así sucesivamente volviendo a leer la siguiente línea del archivo del flujo creado con la siguiente posición del vector hasta el final. Si finalmente devuelve cierto significa que se ha guardado el contenido correctamente del archivo a memoria (en el vector).
Si disponemos de un archivo de eventos de gran tamaño, no podremos visualizar todo el texto que lleva el fichero por pantalla a la vez. Para ello decidí que solo podía mostrar por pantalla un máximo de 22 líneas de texto. Para que se pueda visualizar todo el contenido dispongo de un atributo el cual me apunta a partir de la primera línea que he de mostrar por pantalla para así extraer las 22 que tengo que hacer visualizar.

En este caso pruebo que al tener el archivo más de 22 líneas de texto y el atributo apunta a la primera línea, al querer visualizar la siguiente parte del texto, este atributo apunta a partir de la línea 22:
public boolean down() throws IOException{
    mf.me.e.f=new File("eventosPrueba/3.txt");
    FileInputStream fi = new FileInputStream(mi.me.e.i);
    mf.me.e.txt=mi.me.e.obtenerTxt(fi);
    if(mf.me.e.down){
        mf.me.e.down();
        if(mf.me.e.textoleido==22){
            return true;
        }else{
            return false;
        }
    }
}

Después probamos lo mismo cuando intentamos visualizar texto aunque ya hayamos visualizado la parte final del archivo. Si el atributo apunta a 44 y el texto tiene 50 líneas, ya se están visualizando esas últimas 6 líneas por lo que ese atributo no puede aumentar de valor como podemos comprobar a continuación:

    public boolean down(){
        mf.me.e.down();
        mi.me.e.down();
        if(mf.me.e.textoleido==44){
            return true;
        }else{
            return false;
        }
    }

En el siguiente caso volvemos a visualizar el texto que hay a partir de la línea 22. Como anteriormente apuntaba a 44, te permite realizar la acción y por lo tanto devolverá cierto:

        public boolean up(){
            mf.me.e.up();
            if(mf.me.e.textoleido==22){
                return true;
            }else{
                return false;
            }
    }
A continuación comprobamos el caso en que apunta el atributo a la primera línea del texto e intentamos acceder a parte del texto como si aún no hubiésemos llegado al inicio del texto. En este caso el atributo no se modificará y apuntará a cero, por lo que devolverá cierto:

```java
public boolean up1(){
    mf.me.e.up();
    mf.me.e.up();
    if(mf.me.e.txtLeido==0){
        return true;
    }
    else{
        return false;
    }
}
```

Ahora probaremos el método del juego de pruebas que comprueba si funciona bien el método que devuelve si dos archivos de texto son iguales. Para ello creo dos flujos de datos, uno para cada archivo de texto de pruebas y guardo el contenido de ellos en un vector. Después les pasas esos vectores al método que te lo mira y si devuelve cierto significaría que son iguales, ya que los dos ficheros que le he pasado tienen los mismos datos.

```java
public boolean mirarNoActualizacion() throws IOException{
    mf.me.e.f=new File("eventosPrueba/1.txt");
    mf.me.e.f2=new File("eventosPrueba/1,3.txt");
    FileInputStream fi1 = new FileInputStream(mf.me.e.f);
    FileInputStream fi2 = new FileInputStream(mf.me.e.f2);
    Vector aux=mf.me.e.obtenerTxt(fi1);
    Vector aux2=mf.me.e.obtenerTxt(fi2);
    return mf.me.e.comprobarActualizacion(aux, aux2);
}
```
A continuación pruebo lo mismo pasándole dos archivos diferentes. En este caso si devuelve cierto el método del juego de pruebas, significa que el archivo que hay está actualizado respecto la anterior versión.

```java
public boolean mirarActualizacion() throws IOException{
    mf.me.c.f=new File("eventosPrueba/1.txt");
    mf.me.e.f2=new File("eventosPrueba/1,4.txt");
    FileInputStream fi = new FileInputStream(mf.me.e.f);
    FileInputStream fi2 = new FileInputStream(mf.me.e.f2);
    Vector aux=;mf.me.e.obtenerTxt(fi);
    Vector aux2=;mf.me.e.obtenerTxt(fi2);
    return !mf.me.e.comprobarActualizacion(aux, aux2);
}
```

Ahora he de comprobar el método que guarda el contenido del archivo de texto a visualizar por pantalla en memoria. Para ello simplemente guardo en un vector auxiliar el contenido del archivo de texto “1.txt”. Después ejecuto el método que me guarda en memoria el mismo fichero y utilizo el método que me compara si son iguales. Si es así, como debe ser devolverá cierto el método del juego de pruebas.

```java
public boolean guardarTexto() throws IOException{
    mf.me.e.f3=new File("eventosPrueba/00.txt");
    FileInputStream fi = new FileInputStream(mf.me.e.f);
    Vector aux=;mf.me.e.obtenerTxt(fi);
    mf.me.e.guardarTextoLegado();
    return mf.me.e.comprobarActualizacion(aux, mf.me.e.txt);
}
```

Ahora tenemos que comprobar el método que engloba a las acciones que hemos probado que funcionan correctamente. Cuando se ejecuta el método eventoActualizado() finalmente al estar más actualizada la versión que se recibe, se renombra con el nombre con el cual debe permanecer en el disco duro hasta que haya una versión más moderna. Al existir el archivo, devolverá cierto.
public boolean actualizado(){
    mf.me.e.f=new File("eventosPrueba/1.txt");
    mf.me.e.i2=new File("eventosPrueba/1,1.txt");
    mf.me.e.f3 = new File("eventosPrueba/actualizado().txt");
    mf.me.e.EventoActualizado();
    if{mf.me.e.f3.exists()}{
        return true;
    }else{
        return false;
    }
}
}

En el siguiente caso que probamos al ser los dos archivos a comparar iguales, finalizará la ejecución eliminando de disco la versión del archivo que se ha recibido y manteniendo la original:

public boolean actualizado1(){
    mf.me.e.f3.delete();
    mf.me.e.i=new File("eventosPrueba/1.txt");
    mf.me.e.f2=new File("eventosPrueba/1,2.txt");
    mf.me.e.EventoActualizado();
    if{mf.me.e.i3.exists()}{
        return true;
    }else{
        return false;
    }
}

El último caso a comprobar es que haya errores en los archivos que hay que leer. Para eso le paso dos archivos que no existen, con lo cual cuando abro un flujo para leer el contenido de estos archivos da error (queda guardado en un atributo). Si este atributo toma por valor cierto significa que el método funciona correctamente.
A continuación lo podemos mirar:

```java
public boolean errorActualizado(){
    mf.me.e.f=new File("eventosPrueba/C0.txt");
    mf.me.e.f2=new File("eventosPrueba/O.txt");
    mf.me.e.EventoActualizado();
    if(mf.me.error==true){
        return true;
    }
    else{
        return false;
    }
}
```

En otro módulo del juego de pruebas probamos el caso en que el servidor esté activo, para ello debemos inicializar el servidor. Una vez hecho esto, este método del juego de pruebas nos dice si se ha descargado el archivo de eventos del servidor al disco duro del cliente. Si existe el archivo como así debe ser devolverá cierto este método del juego de pruebas.

```java
public boolean descargaEvento(){
    mf.cl.conexionInicial();
    mf.cl.recibirDatosIniciales();
    mf.me.e.peticionEvento(mf.cl);
    mf.me.e.guardarEvento(mf.cl);
    if(mf.me.e.f2.exists()){  
        return true;
    }
    else{
        return false;
    }
}
```
-Ahora probaremos el funcionamiento de la sección descargas donde podemos visualizar el contenido descargado.

Como el funcionamiento de las acciones para regresar atrás en un menú se hace de la misma manera que en Eventos obviaremos el funcionamiento en el juego de pruebas ya que va de la misma forma.

Para probar que se guardan correctamente el nombre de los archivos de la carpeta “Descargas” en memoria, primero debemos copiar a C:\ la carpeta que tenemos con ese nombre de la carpeta del juego de pruebas. Cuando se ejecute el método del juego de pruebas si copia bien el contenido a memoria, devolverá cierto este método. Con lo cual cuando se tenga que mostrar por pantalla los archivos que hay descargados se visualizará correctamente.

```java
public boolean cargarArchivos()
{
    nf.me.ad.cargarArchivos();
    String $[] = {"114 Valera Avenue.doc", "Conceptos.txt", "m3.png", "x2amd.png"};
    int i=0;
    boolean b=true;
    While(i<mf.me.ad.listaArchivos.length && b)
    {
        if(!mf.me.ad.listaArchivos[i].equals($[i])){
            b=false;
        }
        i++;
    }
    return b;
}
```

Para probar si una imagen se carga correctamente comprobaremos si el atributo que nos indica si ha habido un error se ha modificado. Como inicialmente el valor de este atributo es falso ya que no hay ningún error, después de cargar la imagen debe de mantener el mismo valor. Por lo tanto el método del juego de pruebas devuelve cierto.

```java
public boolean cargaimagen()
{
    mf.me.ad.opcionMenuActual=2;
    mf.me.ad.cargarImagen();
    return !mf.me.ad.errorArchivo;
}
```
También hay que comprobar que cuando seleccionamos una imagen para visualizar, el atributo que te indica que se ha de mostrar por pantalla se ha de modificar para que así se ejecute el método de paint() correspondiente. En este caso devolverá cierto si es correcto.

```java
public boolean verArchivo(){
    mf.me.ad.verArchivo();
    if(mf submenu==5){
        return true;
    }
    else{
        return false;
    }
}
```

En el caso que queramos visualizar un archivo de texto el valor de ese atributo será distinto, con lo cual también debemos de comprobar que el atributo toma el valor correcto. A continuación se ve el método que hace la comprobación:

```java
public boolean verArchivo1(){
    mf.me.ad.opcionMenuActual=1;
    mf.me.ad.verArchivo();
    if(mf.submenu==4){
        return true;
    }
    else{
        return false;
    }
}
```

Para movernos entre los distintos archivos que hay descargados y para visualizar los archivos de texto, las flechas de dirección tienen el mismo funcionamiento que en el primer caso el menú inicial y en el segundo caso como en eventos. Por lo que si tienen alguna duda pueden revisar los puntos anteriores donde comento como va.
Para probar que funciona correctamente la parte cliente-servidor de la descarga de contenidos he realizado las siguientes pruebas.

Para comenzar realizaremos las pruebas con el servidor activo, por lo tanto debemos de ejecutar la aplicación servidor.

A continuación comprobaremos con este método si el servidor se ha ejecutado correctamente, para ello se intenta hacer una conexión a este. Si se ha conectado correctamente como es el caso devolverá cierto:

```java
public boolean servidorActivo(){
    boolean conectado=false;
    try{
        Socket s = new Socket("localhost", 9351);
        conectado=s.is_connected();
    }
    catch(Exception e){
    }
    return conectado;
}
```

A continuación comprobamos si al recibir los datos iniciales (que son el nombre de las carpetas y archivos que contiene la raíz de la carpeta de contenidos del servidor) son los mismos que deberíamos haber recibido mediante este método:

```java
public boolean enviarDatosIniciales() throws IOException{
    mf.cl.conexionInicial();
    mf.cl.recibirDatosIniciales();
    String s[]= {"Documentos", "Imagenes", "Videos"};
    int i=0;
    boolean b=true;
    while(i<mf.cl.v.size() && b){
        if(!mf.cl.v.get(i).equals(s[i])){ \
            b=false;
        }
        i++;
    }
    return b;
}
```
Ahora he de comprobar si las peticiones que he realizado para visualizar un archivo o carpeta se realizan correctamente. Para ello primero he de guardar en el vector la supuesta opción que he seleccionado y comprobar que se ha guardado bien.

Con el siguiente método lo compruebo:

```java
public boolean anadirOpcion() throws IOException{
    int i=2;
    Integer inte = new Integer(i);
    mf.cl.tc.AnadirOpcion(mf.cl, inte);
    inte= (Integer)mf.cl.vc.get(0);
    i=inte.intValue();
    if(i==2){
        return true;
    }else{
        return false;
    }
}
```

Lo siguiente que realizamos es enviar este vector al servidor para que sepa que opción ha elegido el cliente y enviar una respuesta a este (un String que le dice que la petición en este caso es un directorio) para que su aplicación sepa que la opción marcada por él era una carpeta y pueda ejecutar lo que tenga que hacer. En este caso al cumplirse devolverá cierto.

```java
public boolean peticionCarpeta() throws IOException{
    mf.cl.tcEnviarPeticion(mf.cl);
    String t="";
    t = mf.cl.tc.tratarRespuesta(mf.cl);
    if(t.equals("Directorio")){
        return true;
    }else{
        return false;
    }
}
```
Seguidamente lo que se ejecuta es el método de añadir el directorio. Cuando se ejecuta se debe de guardar en un vector el nombre de las carpetas o archivos que contenía la carpeta que seleccionó el usuario anteriormente. En este caso como sabemos el nombre de los archivos que contenía esa carpeta y se comparan con los que hemos guardado en memoria. Si todo es correcto como es el caso devolverá cierto.

```java
public boolean guardarCarpeta() throws IOException{
    int i=2;
    Integer inte = new Integer(1);
    mf.cl.tc.AnadirDirectorio(mf.cl, inte);
    String s[] = {"RAIKKONEN.jpg", "WEIGHTLIFTING.avi"};
    i=0;
    boolean b=true;
    while((i<mf.cl.v.size()) && b){
        if(!mf.cl.v.get(i).equals(s[i])){
            b=false;
        }
        i++;
    }
    return b;
}
```

Ahora probaremos de enviar una petición para descargar un archivo. Funciona todo igual con la única diferencia que por parte del servidor recibiremos un String llamado “Archivo” para indicarle a la aplicación del cliente que el archivo seleccionado era un fichero y se disponga a prepararse para comenzar la descarga del fichero. En este caso debe devolver cierto ya que la petición hecha era de un archivo.

```java
public boolean peticionArchivo() throws IOException{
    int i=1;
    Integer inte = new Integer(1);
    mf.cl.tc.AnadirOpcion(mf.cl, inte);
    mf.cl.tc.EnviarPeticon(mf.cl);
    String t="";
    t = mf.cl.tc.tratarRespuesta(mf.cl);
    if(t.equals("Archivo")){
        return true;
    }
    else{
        return false;
    }
}
```
Aquí probamos el método para volver hacia atrás dentro de los subdirectorios que pueda tener el servidor en su carpeta de contenidos. Para ello hemos puesto el caso en que estemos en una subcarpeta de la raíz de la carpeta de contenidos del servidor. En este caso se coge del vector del cliente donde se guardan las opciones que ha ido seleccionado el usuario y se borra la última. Entonces se envía ese vector, con lo cual se recibirá el nombre de los archivos o carpetas donde se encuentra el directorio padre. Después en el método del juego de pruebas me comprueba si se ha guardado correctamente la información. En este caso me debe devolver cierto.

```java
public boolean esc(){
    mf.cl.conexionInicial();
    mf.cl.recibirDatosIniciales();
    mf.cl.tc.peticionCliente(1, mf.cl);
    mf.cl.esc();
    String s[] = {"Documentos", "Imagenes", "Videos"};
    int i=0;
    boolean b=true;
    while(i<mf.cl.v.size() && b){
        if( mf.cl.v.get(i).equals(s[i]) )
            b=false;
    }
    i++;
    return b;
}
```

En el momento que estamos en la raíz de la carpeta de contenidos del servidor y queremos ir al menú inicial tenemos que comprobar que el atributo que indica que menú se ha dibujado se hace correctamente.

```java
public boolean esc1(){
    mf.cl.conexionInicial();
    mf.cl.recibirDatosIniciales();
    mf.cl.esc();
    if(mf.menu==0){
        return true;
    }
    else{
        return false;
    }
}
```
Una vez hecho este último paso la conexión te debe de cerrar la conexión que hay establecida con el servidor.

Para comprobar que se ha hecho correctamente disponemos de este método:

```java
public boolean conexionOff(){
    mf.cl.cerrarConexion();
    if(mf.cl.socket.isClosed()){ return true; }
    else{
        return false;
    }
}
```

A continuación probaremos los métodos sin tener conexión al servidor. Lo único que debemos comprobar es si el atributo que nos indica si hay un error equivale a cierto, lo que indicaría que nos llevará al menú inicial y nos mostrará un mensaje con el error sucedido.

Primero probamos la conexión con el servidor y recibir los datos iniciales. En este caso devolverá cierto ya que el mensaje de error de conexión se activa y el atributo que indica en que menú estamos coge el valor para visualizar el menú inicial.

```java
public boolean envioDatosIniciales() throws IOException{
    mf.menu=1;
    mf.cl.conexionInicial();
    mf.cl.recibirDatosIniciales();
    if(mf.error==true && mf.menu==0){ return true; }
    else{
        return false;
    }
}
```
Después comprobamos la petición de la carpeta al servidor. Al no haber conexión se activará el atributo que hace mostrar el mensaje de error y también se modificará el atributo que indica en qué menú estamos, con lo que volveremos al menú inicial de la aplicación.

```java
public boolean peticionCarpeta() throws IOException{
    mf.error=false;
    mf.menu=1;
    int i=2;
    Integer inte = new Integer(i);
    mf.cl.tc.AnadirOpcion(mf.cl, inte);
    mf.cl.tc.EnviarPeticion(mf.cl);
    String t="";
    t = mf.cl.tc.tratarRespuesta(mf.cl);
    if(mf.error==true && mf.menu==0){
        return true;
    }
    else{
        return false;
    }
}
```

Después probamos el caso en que se haya perdido la conexión en el momento de recibir los datos de la carpeta que habíamos pedido al servidor. En este caso los dos atributos a mirar cogerán el mismo valor que en el método anterior, por lo que volverá a dar de resultado cierto.

```java
public boolean guardarCarpeta() throws IOException{
    mf.error=false;
    mf.menu=1;
    int i=2;
    Integer inte = new Integer(i);
    mf.cl.tc.AnadirDirectorio(mf.cl, inte);
    if(mf.error==true && mf.menu==0){
        return true;
    }
    else{
        return false;
    }
}
```
Si perdemos la conexión con el servidor mientras realizamos la petición de un archivo, nos debe de devolver el método cierto, lo que demostraría que se han modificado correctamente los valores de los atributos que deben cambiar.

```java
public boolean petecionArchivo() throws IOException{
    mf.error=false;
    mf.menu=1;
    int i=1;
    Integer inte = new Integer(i);
    mf.cl.tc.AñadirOpcion(mf.cl, inte);
    mf.cl.tc.EnviarPeticion(mf.cl);
    String t="";
    t = mf.cl.tc.tratarRespuesta(mf.cl);
    if((mf.error==true && mf.menu==0)){
        return true;
    }else{
        return false;
    }
}
```

Ahora nos toca comprobar cuando queremos ir hacia atrás en un directorio de los que disponga la carpeta de contenidos del servidor. Para ello como tenemos que hacer la petición, si hay un fallo de conexión también se modificarán los dos atributos de antes para que se nos muestre el menú inicial y un mensaje de error para que el usuario sepa donde está. Con este método del juego de pruebas se puede mirar si estos atributos toman los valores que les corresponden.

```java
public boolean esc(){
    mf.menu=1;
    mf.cl.conexionInical();
    mf.cl.recibirDatosIniciales();
    mf.cl.tc.peticionCliente(1, mf.cl);
    mf.cl.esc(1);
    if((mf.error==true && mf.menu==0)){
        return true;
    }else{
        return false;
    }
}
```
Ahora toca comprobar la descarga de un archivo si se realiza correctamente. Como sabemos el nombre del archivo que debe haberse descargado, simplemente miramos si existe ese fichero. Si todo es correcto este método del juego de pruebas debe de devolver cierto.

```java
public boolean descargaActiva(){
    mm.cl.conexionInicial();
    mm.cl.recibirDatosIniciales();
    mm.cl.v.e.addElement(1);
    mm.cl.tc.peticionCliente(0, mm.cl);
    try{
        while(mm.cl.tc.m.isAlive()){}
    }
    catch (Exception e){
    }
    File f=new File("C:/Descargas/Blu Ray M3.jpg");
    if(f.exists()){ return true; }
    else{
        return false;
    }
}
```

Ahora probaremos de descargar un archivo sin el servidor activo. En este caso se activa el atributo que nos mostrará un mensaje de error por pantalla y el atributo que hace que se tenga que visualizar el menú inicial.

```java
public boolean descargaInactiva(){
    mm.menu=4;
    mm.cl.tc.m = new DescargaFichero(mm.cl);
    mm.cl.tc.m.start();
    while(mm.cl.tc.m.isAlive()){}
    if(mm.error & mm.menu==C){
        return true;
    }
    else{
        return false;
    }
}
```
Con todas estos test he finalizado el juego de pruebas. He intentado abarcar todos los casos disponibles para asegurar el correcto funcionamiento de la aplicación. Gracias a este juego de pruebas he podido conseguir una aplicación lo más compacta y robusta posible.
7. Tutorial

7.1 Aplicación cliente

7.1.1 Requisitos

- Requisitos de hardware

Los requisitos mínimos para el funcionamiento de la aplicación son:

-Procesador:

Intel® Core™ Duo T2400 1.83 GHz.
AMD Athlon 64 X2 3800+ 2.0 GHz.

-Memoria:

512 MB de memoria RAM o superior.

-Tarjeta gráfica:

Asegúrese que soporta aceleración de hardware.

128 MB VGA VRAM o superior.

NVIDIA GeForce® 7600, ATI Radeon™ X 1600.

Los requisitos recomendados para el óptimo funcionamiento de la aplicación son:

-Procesador:

Intel: Core™ Duo T2500 2.0 GHz, Core Duo E6400 2.13 GHz, Core 2 Quad Q6600 2.4 GHZ, Core 2 Duo 5500 1.66 GHz, o superior.
AMD: AMD Athlon 64 FX-60 2.6 GHz, AMD Athlon 64 X2 4200+ 2.2 GHz, AMD Turion™ 64 X2 TL-60 2.0 Ghz, o superior.
-Memoria:

1 GB de memoria RAM o superior.

-Tarjeta gráfica:

NVIDIA GeForce 8400, ATI Radeon HD 2400 o superior.

-Versión de los drivers de la tarjeta gráfica:

Para asegurarse de que pueden disfrutar de plena calidad de Blu-ray, por favor, visite la páginas web de ATI y NVIDIA para descargar los controladores más recientes.

-Disponer de conexión a internet.

- Requisitos de software

Estos son los siguientes requisitos de software que necesita la aplicación para funcionar:

- Sistemas operativos soportados: Windows XP con Service Pack 2 o Windows Vista.

- Java SE Runtime Environment 1.4.2 o superior. Si no lo tiene en su ordenador lo puede descargar desde http://www.java.com

- Requisitos del simulador de Blu-ray: InterVideo WinDVD 8.
7.1.2 Funcionamiento

En este apartado describiré como el usuario puede utilizar las diferentes funcionalidades de que dispone la aplicación.

- Visualizar la película

Una vez inicializada la aplicación vemos la pantalla del menú inicial. Para visualizar la película tenemos que acceder a la opción de “REPRODUCIR PELÍCULA” de este menú. Para poder desplazarnos entre las diferentes opciones disponibles del menú hay que utilizar las flechas de dirección y para entrar a un submenú de este pulsar el “enter”.

Una vez dentro, estaremos en el menú de video con las distintas opciones de reproducción que son el “PLAY”, “PAUSE” y “STOP”.

En este caso para comenzar la reproducción de la película pulsaremos el botón del “PLAY”. Como podremos observar el menú desaparecerá cuando le demos a esta opción.

-Pausar la película

Una vez que estamos visualizando la película tenemos que pulsar la tecla “esc” para visualizar de nuevo el menú con las opciones de reproducción. En este caso solo nos aparecerán las dos únicas opciones disponibles que son el “PAUSE” y el “STOP”.

Una vez aparecido el menú pulsamos “enter” en la opción “PAUSE” para pausar la película.

-Quitar la pausa a la película

Para realizar esta acción simplemente debemos volver a pulsar con la tecla “enter” en la opción “PAUSE” y se volverá a reproducir la película en el punto donde la habíamos dejado y el menú donde nos encontramos desaparecerá.
-Detener la reproducción de la película

Primeramente debemos regresar al menú donde aparecen las opciones de reproducción. Para ello pulsamos la tecla “esc”. Volveremos a visualizar este menú:

Después pulsamos “enter” en la opción “STOP” de este menú y conseguiremos nuestro objetivo de detener la reproducción de la película. Una vez hecho esto en el menú volverá a aparecer la opción de reproducción del “PLAY” para volver a comenzar a visualizar la película.

-Cambiar el modo de audio

Para ello primero debemos estar reproduciendo un video. Una vez que estamos visualizándolo debemos pulsar la tecla “esc” para ir al menú de opciones de reproducción.

Después volvemos a darle a la tecla “esc” para ir al menú inicial de la aplicación donde nos aparece la opción “AUDIOS” para ir al menú de audios. Para ello pulsamos “enter” en esa opción.

Una vez en el menú de audio nos aparecen dos modos disponibles para escuchar el video. Disponemos de “ESTEREO PCM” y “5.1 DOLBY DIGITAL”. Seleccionamos el modo que queremos utilizando la tecla “enter” y una vez hecho esto desaparecerá el menú y seguiremos con la visualización de la película con la opción elegida.

-Visualizar galería de imágenes

Para acceder a esta opción primeramente debemos acceder al menú extras desde la opción del menú inicial llamado “EXTRAS” utilizando la tecla “enter”.

Para visualizar la galería fotográfica pulsaremos sobre la opción “GALERIA DE IMÁGENES” pulsando la tecla “enter”. Una vez hecho esto veremos la galería de fotos.
Para visualizar sobre que imagen nos encontramos, nos aparece un sombreado gris de fondo sobre ella. Para desplazarnos sobre las diferentes imágenes disponibles utilizaremos las teclas de dirección. Cuando queremos visualizar una imagen en tamaño original, la seleccionamos pulsando “enter”.

Una vez vista la imagen a tamaño original si queremos volver al estado anterior para seguir viendo la galería con las imágenes disponibles, le daremos a la tecla “esc” con lo que conseguiremos volver al estado anterior.

-Visualizar trailer

Para ello debemos acceder al menú extras desde la opción “EXTRAS” que se encuentra en el menú inicial.

Una vez hecho esto pulsaremos la opción “TRAILER” del menú de opciones de extras con la tecla “enter”.

Después desaparecerá el menú y se comenzará a reproducir el trailer. Para poder pausar o detener la reproducción de este simplemente tenemos que seguir los mismos pasos que cuando estamos reproduciendo la película de la aplicación.

-Visualizar eventos

Para ello debemos acceder al menú extras desde la opción “EXTRAS” que se encuentra en el menú inicial.

Una vez hecho esto pulsamos sobre la opción de “EVENTOS” dentro del menú de extras para poder visualizar eventos que se organizan derivados de la película.

Directamente intentará conectar al servidor para descargar la última versión que disponen y comprobar con la que tenemos nosotros cual está más actualizada. Dependiendo del resultado se nos mostrará una versión de los eventos. Podremos visualizar el texto a través de la pantalla.
Dependiendo de la cantidad de texto que tenga esta sección no se podrá visualizar todo por pantalla por cuestiones de espacio. Para poder ver el resto del contenido utilizaremos las flechas de dirección para visualizar el texto que falta por ver. Si queremos volver al menú anterior simplemente deberemos pulsar la tecla “esc”.

-Visualizar archivos descargados

Para ello debemos acceder al menú extras desde la opción “EXTRAS” que se encuentra en el menú inicial:

Una vez hecho esto pulsamos sobre la opción de “DESCARGAS” dentro del menú de extras para poder visualizar una lista con los archivos que hemos descargado con la aplicación.

Después si queremos visualizar un archivo simplemente debemos pulsar la tecla “enter”. Tenemos que tener en cuenta que la aplicación solo puede visualizar imágenes con extensión .jpg o .png y archivos de texto .txt. En caso de no ser así no se podrá visualizar el contenido de ellos por problemas de compatibilidad. Para visualizar el resto de archivos se tendrá que utilizar programas compatibles con ese tipo de ficheros.

-Visualizar contenido adicionales

Para ello debemos seleccionar la opción del menú inicial “DESCARGA DE CONTENIDOS”:

Si tenemos un problema de conexión a internet se nos notificará por pantalla en cualquier momento de la conexión, con lo cual apareceremos en el menú inicial de la aplicación.

En el caso que la conexión haya funcionado, apareceremos en una nueva pantalla en la cual nos aparecen directorios para navegar entre ellos o archivos para poder descargar.
Para navegar por subdirectorios del servidor utilizaremos el botón “enter”. Para ir hacia estados anteriores tendremos que irle dando al “esc”.

-Descargar un archivo del servidor

En el momento que queramos descargar un archivo pulsamos “enter” sobre él y se nos aparecerá otra pantalla.

En ella podremos ver el estado actual de la descarga en tiempo real. Como podemos ver en cualquier momento durante la descarga tenemos la posibilidad de cancelarla dándole al botón “enter”, el cual nos mostrará de nuevo el contenido del directorio donde nos encontrábamos anteriormente.

Una vez descargado el archivo nos mostrará una opción para aceptar la descarga. Entonces pulsamos “enter” y volvemos al directorio de donde elegimos descargar el archivo que seleccionamos para poder seguir interactuando con el contenido disponible.

Para la ejecución de la aplicación utilicé una imagen de un disco de Blu-ray que me han prestado en la empresa Infinia. Así puedo visualizar mi autoría.
7.2 Aplicación servidor

7.2.1 Requisitos

- Requisitos de hardware

Los requisitos mínimos para el funcionamiento de la aplicación son:

- Procesador:
  Intel Pentium 4 o AMD Athlon™.

- Memoria:
  512 MB de memoria RAM.

- Tarjeta gráfica:
  Integrada en placa base.

- Disponer de conexión a internet.

Los requisitos recomendados para el óptimo funcionamiento de la aplicación son:

- Procesador:
  Intel Core 2 Duo o AMD Athlon™ 64 X2 Dual-Core o superior.

- Memoria:
  1024 MB de memoria RAM o superior.
-Tarjeta gráfica:

NVIDIA GeForce® 6800, ATI Radeon™ X 850 o superior.

-Disponer de conexión a internet.

- Requisitos de software

Estos son los siguientes requisitos de software que necesita la aplicación para funcionar:


-Java SE Runtime Environment 1.4.2 o superior. Si no lo tiene en su ordenador lo puede descargar desde http://www.java.com
7.2.2 Funcionamiento

Se crea una carpeta y en ella introducimos los ficheros Servidor.class, HiloServidor.class y PeticionesCliente.class, después se tiene que crear una carpeta en ese directorio llamada “Contenido”. En esta carpeta introduciremos todo el contenido adicional que el cliente puede descargar. También crearemos en ese directorio otra carpeta llamada “Eventos” en la cual pondremos el archivo de texto “Eventos.txt” con la última versión disponible para que la pueda descargar el cliente.

Después ejecutamos la consola de Windows y navegamos hasta el directorio donde tenemos los .class y ejecutamos este código “java Servidor”. Poniendo de ejemplo que tengamos el código en esta carpeta:
“D:\Programas\Java\jdk1.6.0\bin”

Primero ejecutamos la consola ms-dos. Para ello pulsamos “Inicio”, “Ejecutar” y escribimos en la pantalla que se nos muestra el comando “cmd”. Se aparecerá esta pantalla:

Fig 46. Consola de MS-DOS

Después ejecutamos en consola el comando:
cd Programas\Java\jdk1.6.0\bin
Con esto apareceremos en la carpeta donde hemos depositado los .class como se puede ver a continuación:

Fig 47. Ejemplo de pantalla después de acceder al directorio de la aplicación

Más tarde ejecutamos el comando:
java Servidor
En ese momento comienza la puesta en funcionamiento de la aplicación como podemos observar en la siguiente imagen:

![Fig 48. Pantalla de ejecución de la aplicación](image)

Una vez hecho esto ya la aplicación comienza su ejecución y no termina hasta que la cerremos nosotros mismos. Podemos detener la aplicación utilizando “Control+c” o utilizando la pestaña “X” de la ventana ms-dos desde la cual estamos ejecutando la aplicación.
8. Conclusiones

Después de la realización del proyecto creo que se han conseguido todos los objetivos que me propuse en el inicio del mismo.

- Se ha conseguido una aplicación que incluya las nuevas características del perfil 2.0 de Blu-ray para descargar contenidos adicionales descargados de un servidor.

- Creo que he conseguido una interfaz gráfica estéticamente bonita y agradable a la vista. Además es muy intuitiva ya que el contenido está bien distribuido y permite al usuario diferenciar las diferentes partes de la aplicación con gran facilidad lo que le permite moverse con comodidad.

- En la medida de lo posible creo que he conseguido una aplicación reusable. Al haber conseguido tener el código de la aplicación bien estructurado ayuda a su entendimiento por parte de algún programador que intente implementar una mejora.

- He creado una aplicación robusta que tolere errores que puedan surgir.

- Se ha conseguido que la aplicación funcione en cualquier reproductor de Blu-ray para ordenador.

- La aplicación es portable entre un ordenador y otro, ya que no requiere complicados procesos de instalación y su tamaño es pequeño.

Durante la realización del proyecto ha habido algunos retrasos en alguna de las fases que lo forman. Para implementar la descarga de contenidos hubo un retraso para su implementación que se tradujeron en 20 horas más de trabajo. También en la fase del juego de pruebas debido a la dificultad de solucionar algunos errores se aumentó el tiempo de trabajo en 6 horas.
A continuación podemos ver el nuevo diagrama de Gantt después de los retrasos:

<table>
<thead>
<tr>
<th></th>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Estudio de la propuesta del proyecto</td>
<td>7 días</td>
<td>jue 21/06/08</td>
<td>vie 29/06/08</td>
</tr>
<tr>
<td>2</td>
<td>Familiarización con el lenguaje y el entorno de programación</td>
<td>40 días</td>
<td>lun 01/07/08</td>
<td>vie 24/10/08</td>
</tr>
<tr>
<td>3</td>
<td>Análisis y especificación</td>
<td>15 días</td>
<td>lun 27/01/08</td>
<td>vie 14/11/08</td>
</tr>
<tr>
<td>4</td>
<td>Diseño y casos de uso</td>
<td>24 días</td>
<td>lun 17/11/08</td>
<td>jue 18/12/08</td>
</tr>
<tr>
<td>5</td>
<td>Diseño de la interfaz gráfica</td>
<td>3 días</td>
<td>vie 19/12/08</td>
<td>mar 23/12/08</td>
</tr>
<tr>
<td>6</td>
<td>Implementación de la interfaz gráfica</td>
<td>118 días</td>
<td>vie 19/12/08</td>
<td>mar 02/03/09</td>
</tr>
<tr>
<td>7</td>
<td>Implementación de la descarga de contenidos</td>
<td>61 días</td>
<td>vie 19/12/08</td>
<td>vie 13/03/09</td>
</tr>
<tr>
<td>8</td>
<td>Implementación de las opciones de video</td>
<td>8 días</td>
<td>lun 16/03/09</td>
<td>mié 25/03/09</td>
</tr>
<tr>
<td>9</td>
<td>Implementación de las opciones de audio</td>
<td>6 días</td>
<td>jue 26/03/09</td>
<td>jue 02/04/09</td>
</tr>
<tr>
<td>10</td>
<td>Implementación de los extras</td>
<td>43 días</td>
<td>vie 03/04/09</td>
<td>mar 02/05/09</td>
</tr>
<tr>
<td>11</td>
<td>Pruebas</td>
<td>14 días</td>
<td>mié 03/06/09</td>
<td>lun 22/06/09</td>
</tr>
<tr>
<td>12</td>
<td>Redacción de la memoria</td>
<td>171 días</td>
<td>lun 27/01/08</td>
<td>lun 22/06/09</td>
</tr>
</tbody>
</table>

Fig 49. Diagrama de Gantt final
Después de los retrasos en la realización del proyecto, he vuelto a calcular el coste económico:

<table>
<thead>
<tr>
<th>Etapa</th>
<th>Precio/hora</th>
<th>Horas</th>
<th>Coste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiarización con el lenguaje y el entorno de trabajo</td>
<td>30 €</td>
<td>64</td>
<td>1920 €</td>
</tr>
<tr>
<td>Análisis y especificación</td>
<td>40 €</td>
<td>32</td>
<td>1280 €</td>
</tr>
<tr>
<td>Diseño y casos de uso</td>
<td>40 €</td>
<td>50</td>
<td>2000 €</td>
</tr>
<tr>
<td>Diseño de la interfaz gráfica</td>
<td>30 €</td>
<td>10</td>
<td>300 €</td>
</tr>
<tr>
<td>Implementación</td>
<td>30 €</td>
<td>140</td>
<td>4200 €</td>
</tr>
<tr>
<td>Pruebas</td>
<td>30 €</td>
<td>30</td>
<td>900 €</td>
</tr>
</tbody>
</table>

Como podemos ver el coste final del proyecto es de 10600 €. El coste que estimé al iniciar el proyecto era de 9820 €.

Como podemos observar el coste ha aumentado en 780 € después de los retrasos que hemos sufrido durante la realización del proyecto.
8.1 Aprendizaje personal

Con la realización de este proyecto he aprendido bastante acerca del proceso desde que una idea surge hasta que esa idea se concreta en una aplicación informática. En primer lugar, he podido poner en práctica los conocimientos aprendidos en las diferentes asignaturas, de tal forma que pude aplicar los conocimientos que adquirí sobre la fase de análisis de un proyecto informático, los conocimientos necesarios para realizar un buen diseño y los conocimientos que me permitieron implementar la aplicación con el lenguaje de programación que escogí. También pude practicar la gestión de un proyecto informático y la forma como los recursos deben repartirse y asignarse. Además, pude trabajar y diseñar una aplicación con una interfaz gráfica de usuario que requería que ésta fuera lo más cómoda y fácil de usar posible. En segundo lugar, gracias a este proyecto, he podido aprender bastante sobre el lenguaje de programación Java de Sun Microsystems. Estoy seguro de que estos conocimientos me serán de gran utilidad en mi futuro laboral y personal. En último lugar, he de decir que estoy muy agradecido de poder haber trabajado y conocido la tecnología Blu-ray, la cual está en pleno auge después de imponerse este formato creado por Sony al HD DVD de Toshiba. Gracias a este proyecto, tengo una visión más cercana del proceso de postproducción del cine a su grabación en un disco, así como de las características técnicas, funcionamiento y estructura del mundo Blu-ray.

8.2 Futuras mejoras

Cuando terminé de realizar la aplicación me di cuenta que me hubiera gustado implementar más funcionalidades de las que he realizado, pero por falta de tiempo no pude llegar a hacerlo. Preferí realizar una aplicación con todas sus funciones terminadas y sin errores en su ejecución.
Para una nueva versión de la aplicación he pensado que se podrían incorporar las siguientes mejoras:

- Añadir compatibilidad con reproductores de Blu-ray de sobremesa.
- Poder visualizar los subtítulos de la película.
- La aplicación se pueda ejecutar en varios idiomas y no solo en castellano como está ahora.
- Añadir interfaz gráfica a la aplicación servidor.
- Posibilidad de cambiar la visualización gráfica de la aplicación cliente con los colores deseados, a modo de skins.
- Realizar compras de productos on-line.
9. Bibliografía

Para realizar este proyecto se han tenido en cuenta las siguientes fuentes bibliográficas:

Libros:


Internet:

- [1] Blu-ray
  http://es.wikipedia.org/wiki/Blu-ray
  Fecha de acceso: 1/9/08

- [2] Xlet
  http://www.cesnavarra.net/cesdigital/Lists/Noticias%20CESDigital/DispFormCES.aspx?List=5ec0dfc7-7911-470b-8b6b-71ba72783fdd&ID=23
  Fecha de acceso: 15/9/08
  http://www.xlet.net/
  Fecha de acceso: 18/9/08
• [3] Alta definición
  http://www.zonadvd.com/modules.php?name=Sections&op=viewarticle
  &artid=608
  Fecha de acceso: 5/9/08

• Foro Java
  http://forums.sun.com/index.jspa
  Fecha de acceso: 4/10/08

• Tutorial Java
  http://www.programacion.com/java/tutoriales/
  Fecha de acceso: 7/10/08
  http://www.cica.es/formacion/JavaTut/
  Fecha de acceso: 12/10/08

• [4] BD-Live
  http://www.zonadvd.com/modules.php?name=Sections&op=viewarticle
  &artid=858
  Fecha de acceso: 23/9/08
  http://www.xataka.com/hd/bd-live-el-perfil-20-de-blu-ray
  Fecha de acceso: 25/9/08

• [5] J2ME
  http://grasia.fdi.ucm.es/j2me/_J2METech/index.html
  Fecha de acceso: 30/9/08

PFC:

  FIB.