Universitat Oberta de Catalunya

Memoria

Proyecto Fin de Carrera

"Sistema de vídeo vigilancia IP"

Miguel Ángel Fuentes Casanova

12/01/2015

Contenido

1. Introducción

El siguiente documento presenta la Memoria que he seguido para la realización de mi Trabajo Fin de Carrera en Ingeniería Técnica de Informática de Sistemas. Este documento recopila una síntesis del análisis, del diseño final y del plan de trabajo; además de información adicional.

Se trata de una propuesta totalmente personal no seleccionada de entre las ofertadas en el curso. Consiste en un Sistema de vídeo-vigilancia que permite a sus usuarios la monitorización de las imágenes capturadas por cámaras IP o conectadas localmente.

Surge de mi propia necesidad de tener vídeo-vigiladas ciertas partes de mi casa y para demostrar los conocimientos y destrezas adquiridos a lo largo de mi carrera estudiantil y profesional en este oficio. He desarrollado este proyecto suficientemente complejo para que sirva como Proyecto de Fin de Carrera.

Igualmente, el proyecto está orientado para una posible liberalización como software gratuito (no “open source”) y permitir a empresarios o particulares su uso. Si hubiese negocio sería como futura oferta de soporte a clientes o mejoras solicitadas del producto.

1.1. Tecnología utilizada

Se ha elegido el lenguaje C# porque es uno de los lenguajes más flexibles y cómodos que hay actualmente. No hay que olvidar que está orientado a alto nivel con lo que dispone de una amplia biblioteca (.NET) para conseguir realizar nuestros desarrollos más fácilmente, de una robusta manera de funcionar donde todo error es controlable y que disponemos de una recuperación automática de recursos. La posibilidad de realizar todo tipo de tareas gráficas es, gracias al gran editor Visual Studio también, una tarea cómoda y agradable.

1.2. Objetivo principal

El objetivo del proyecto es realizar un sistema de vídeo vigilancia en un entorno gráfico vistoso lejos del estándar típico de botones rectangulares de Windows. El sistema permitirá monitorizar las imágenes de hasta 16 cámaras conectadas localmente al ordenador o en red. Para permitir cierta flexibilidad en ciertos aspectos del entorno se ha decidido utilizar ficheros XML que permitirán:

Desde el mismo programa:

- Mantener la configuración de opciones.
- Mantener la configuración de salidas a pantallas.
- Configuración de cámaras.
- Salvaguarda de la última configuración visual: vistas, cámaras, volumen general, volumen por cámara y efectos.

De manera externa al software personalización del:

- Entorno gráfico (“skinning”)*.
- El idioma.
- Configuración de distribución de cámaras en las vistas.

*: El sistema “skinning” que yo desarrollaré permitirá personalizar vía XML distintos atributos gráficos pero no así la posición y otros aspectos de los controles ya que éstos serán determinados y guardados en tiempo de diseño. Esto no impide que se pueda alterar –indirectamente- la posición final simplemente manteniendo imágenes suficientemente grandes y desplazando en su interior el diseño.

2. Descripción del proyecto

2.1. Justificación del proyecto

2.1.1. ¿Por qué el proyecto?

Existen multitud de sistemas de video-vigilancia en el mercado, la gran mayoría basados en cámaras analógicas o digitales tradicionales por radiofrecuencia o cableadas, concentradores y aplicaciones software que permiten su gestión, todos ellos a precios generalmente elevados; a un nivel superior se compran junto a ordenadores servidores y clientes, monitores y concentradores de los mismos, sistemas de respaldo de datos y de corriente, guardias de seguridad, centralitas de atención y emergencia... Por supuesto con ello va todo un mercado que persigue que se siga consumiendo lo más posible una vez que se instala o contrata el sistema de video-vigilancia y/o seguridad: a través de mantenimientos, licencias, actualizaciones software/hardware, periféricos especializados (micrófonos, detectores de movimiento, detectores de presión, cámaras infrarrojas, etc.).

Vivimos en una época de grandes cambios tecnológicos en los que se busca embeber el “Internet de las cosas” en todo lo que nos rodea, impulsados por esa moda y por nuestro inherente gusto por el ahorro, vamos buscando alternativas a las antiguas tendencias incluso en seguridad tecnológica, acudiendo cada vez más al “DIY” (hazlo tú mismo). En tecnología es una moda que muchos seguimos incluso sin ni siquiera darnos cuenta, ocurre desde el momento en que comenzamos a preferir cubrir nuestras necesidades con opciones más económicas aun sabiendo que tendremos que hacer un pequeño esfuerzo extra para adecuarlo totalmente a nosotros.

Un claro ejemplo del “DIY” en el nivel profesional muy de moda en nuestra época, es la utilización de software libre. Así, cada vez más empresas usan distribuciones GNU/Linux como sistemas operativos en lugar de las alternativas de pago (como Windows, Macintosh o HP-Unix); sabiendo que a cambio de un nulo o menor precio tendrán que trabajar mucho más duro que con sistemas de pago donde “casi todo” ya está hecho y el soporte es completo.

Pienso que estamos volviéndonos una sociedad cada vez más capaz de afrontar la constante evolución tecnológica, y que cada vez más sabemos aprovechar mejor este acceso a toda la información a golpe de clic, lo cual nos convierte en “manitas digitales”. ¿Quién no sabe hoy en día cómo encontrar una canción concreta e incluso descargarla? Esta creciente seguridad en nosotros mismos proviene de saber que todo lo que queramos conocer o conseguir está en internet, y esto nos hace personas más capaces en el “DIY” digital.

Dado que cada vez existe un mayor mercado de cámaras IP o webcams cableadas a mucho menor precio que ofrecen una buena calidad, se hace patente que el ahorro es mucho mayor cuando pensamos en la implantación de un sistema de vídeo-vigilancia aprovechando esas cámaras además de infraestructuras de telefonía e internet preexistentes (cableadas y/o inalámbricas) en empresas y domicilios particulares.

Por tanto, siguiendo la actual corriente del “DIY” y aprovechando la infraestructura existente cualquier persona sin especial formación podría montar su propio sistema de video-vigilancia con mi software y unas cuantas cámaras. Lo ideal serían cámaras IP inalámbricas por su sencillez de instalación pero también es factible ahorrar –por ejemplo- conectando por cable USB las cámaras que ya tengamos en casa o consigamos de segunda mano a bajo precio. Incluso, en su mínima expresión, podríamos usar las cámaras integradas en nuestros dispositivos móviles convirtiéndolos[1] en cámaras IP y ubicándolos con soportes de plástico de bajo precio.

NOTA 1: Un software “open source” que permite esto es el “DroidCam Wireless Webcam” para Android, cuya dirección Web indico en la bibliografía.

2.1.2. Descripción del proyecto

Es una solución orientada a pequeñas empresas o negocios familiares donde los recursos y las necesidades son mínimos, así el programa corre en un único ordenador (por lo que no sigue la arquitectura cliente-servidor) y solo va a requerir para su instalación el uso de un sencillo instalador que desarrollaré. Si existiera el tiempo suficiente y con objeto de aprovechar otras cámaras –no IP- conectadas localmente a ordenadores en la red, desarrollaré una utilidad (servicio Windows) que permita hacer estas cámaras visibles al programa principal.

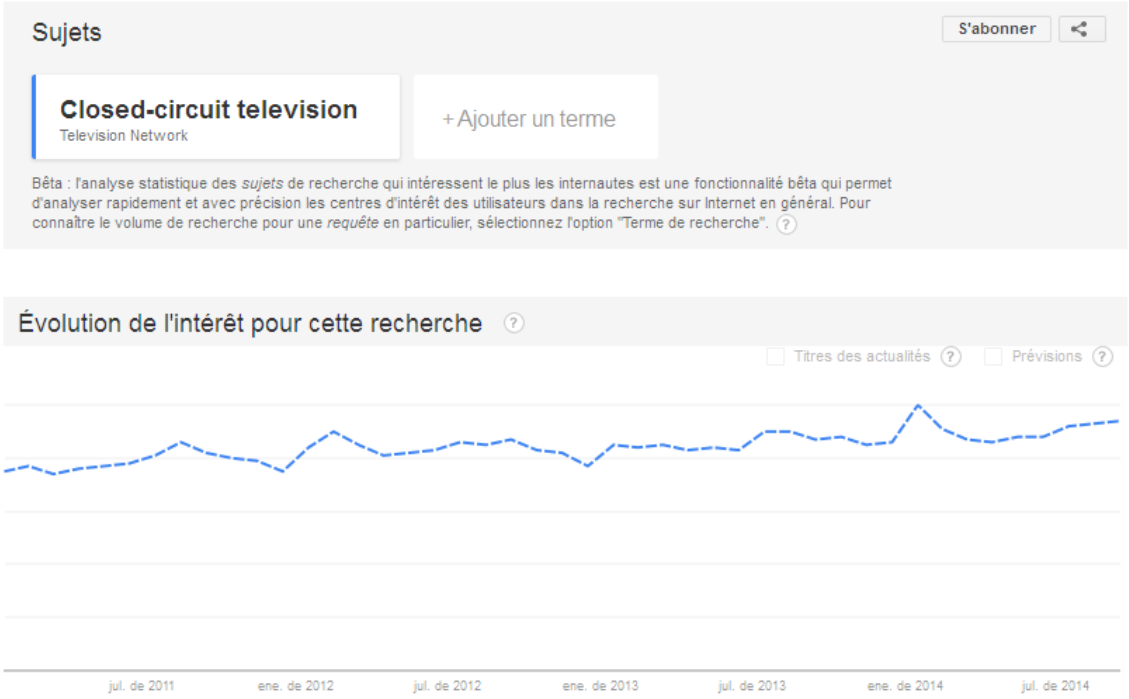
El proyecto principal permitirá la captura de vídeo de hasta 16 cámaras desde un único ordenador, lo que no impide hacer uso del mismo software en más ordenadores para ampliar fácilmente su alcance. En este caso, queda remarcar que se deberá tener en cuenta que todas esas instalaciones serían totalmente independientes no habiendo comunicación expresa entre ellas a través de dicho software.

El software está especialmente pensado para ser utilizado por cualquier usuario sin experiencia previa en este tipo de soluciones. Prueba de ello será su interfaz sencilla y gráfica con grandes botones, totalmente lista para ser usada en un monitor táctil; sin ser necesario el uso de herramientas especiales como las utilizadas por los profesionales, tales como joysticks de control de cámaras, teclados especializados o sistemas de backup.

2.1.3. Estudio de mercado

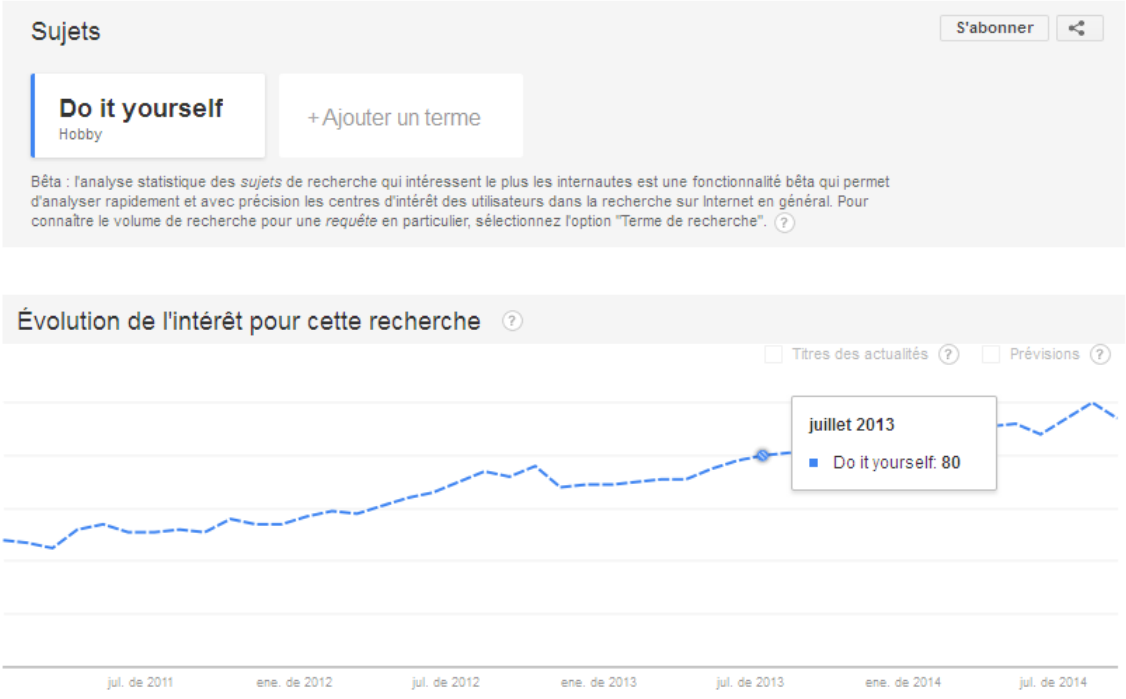
Dado que la vigilancia de nuestras pertenencias es una cuestión que siempre nos va a preocupar, parece lógico pensar que mi desarrollo es algo que siempre tendrá su nicho. Concretamos nuestro tipo de cliente como ese tipo de persona, gran parte de la población, capaz de encontrar lo que necesita y que gusta del “DIY” o simplemente por ahorrar gastos.

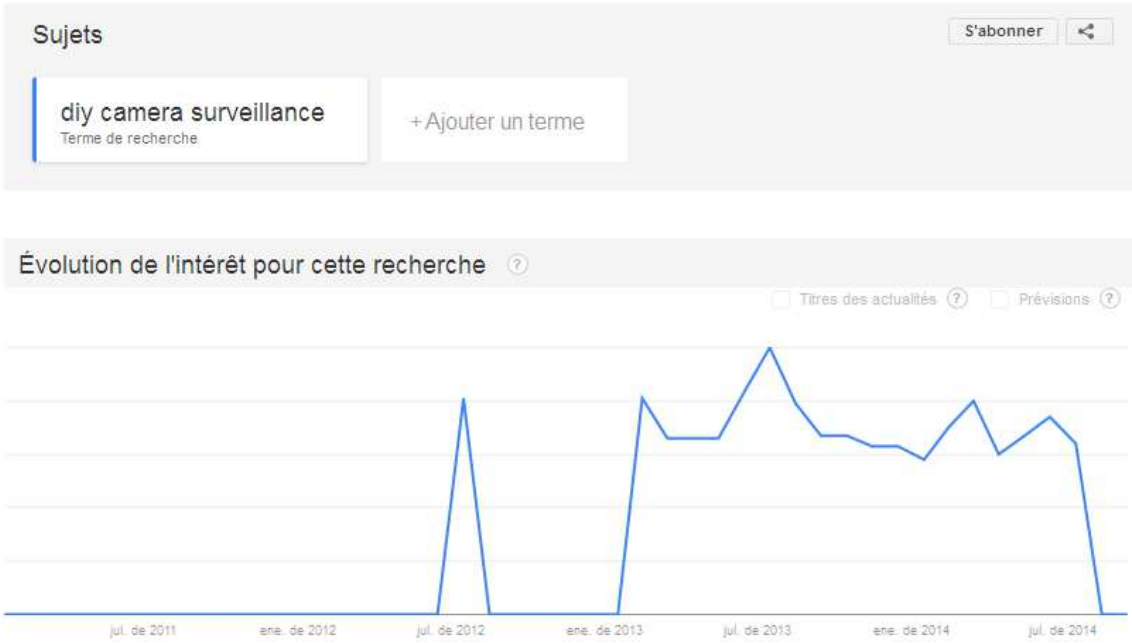
Según el prestigioso motor de búsqueda “Google”, la preocupación de la gente por buscar términos relacionados con la vigilancia a través de cámaras es bastante grande y las previsiones van al alza:




Sujets
S'abonner
camera surveillance
Terme de recherche
+ Ajouter un terme
Évolution de l'intérêt pour cette recherche
Titres des actualités
Prévisions
G
F
E
D
C
B
A
jul. de 2011
ene. de 2012
jul. de 2012
ene. de 2013
jul. de 2013
ene. de 2014
jul. de 2014

Sobre el “DIY”:


Sujets
S'abonner
Do it yourself
Hobby
+ Ajouter un terme
Bêta : l'analyse statistique des sujets de recherche qui intéressent le plus les internautes est une fonctionnalité bêta qui permet d'analyser rapidement et avec précision les centres d'intérêt des utilisateurs dans la recherche sur Internet en général. Pour connaître le volume de recherche pour une requête en particulier, sélectionnez l'option "Terme de recherche".
Évolution de l'intérêt pour cette recherche
Titres des actualités
Prévisions
juillet 2013
Do it yourself: 80
jul. de 2011
ene. de 2012
jul. de 2012
ene. de 2013
jul. de 2013
ene. de 2014
jul. de 2014


Sujets
S'abonner
diy camera surveillance
Terme de recherche
+ Ajouter un terme
Évolution de l'intérêt pour cette recherche
Titres des actualités
Prévisions
jul. de 2011
ene. de 2012
jul. de 2012
ene. de 2013
jul. de 2013
ene. de 2014
jul. de 2014

2.2. Objetivos del proyecto

2.2.1. Objetivos generales

1) Facilitar a sus usuarios realizar vídeo-vigilancias a través de cámaras IP y locales, que permitan monitorizar zonas privadas o públicas: tiendas, gasolineras, centros de enseñanza, el interior de domicilios privados, etc.

2) Realizar herramientas que permitan cierto nivel de implicación más activa en la vigilancia a través de funciones que permitan analizar las imágenes en tiempo real: como el “zoom” (cambios en la escala de la imagen –acercamientos y alejamientos-) o el “pan” (posicionar el centro de la imagen).

3) Compartir a través de red, y por tanto con el programa principal, las cámaras localmente conectadas a un ordenador, siempre y cuando disponga de suficiente tiempo para desarrollar el servicio secundario que ya comenté.

4) Permitir la grabación de contenido que provenga de las cámaras, siendo también dependiente del tiempo que disponga.

Aunque no un objetivo directo, debe quedar constancia de que se va a informar y recordar a sus usuarios que se debe respetar la legislación vigente en el país donde se quieran realizar filmaciones de vídeo y/o sonido.

2.2.2. Objetivos específicos

1) Desarrollar en lenguaje de programación C# una aplicación de escritorio para sistemas operativos Microsoft Windows que soporten la instalación del Framework .NET 4.0, capaz de conectarse a cámaras IP y locales a través de una interfaz sencilla pero completa e intuitiva, que permita no solo ser usada mediante ratón sino también a través de pantallas táctiles.

2) Desarrollar en el mismo lenguaje y para el mismo tipo de sistema operativo que el anterior, un servicio que permita que un ordenador comparta como cámaras IP sus cámaras localmente conectadas, siempre y cuando existiera tiempo suficiente para el desarrollo.

2.3. Requerimientos de la solución

2.3.1. Funcionales

Las siguientes son las funcionalidades que necesitamos alcanzar para cumplir los objetivos generales. El usuario podrá:

- Monitorizar desde 1 hasta 16 cámaras IP o conectadas localmente, concretamente el vídeo o grupo de fotografías en cadena que capturan las mismas.

- Configurar las cámaras e incluso probarlas durante el proceso in-situ.

- Configurar las opciones básicas generales del programa.

- Elegir entre ver desde ninguna hasta 16 cámaras todas a la vez o solo algunas (aunque solo en grupos y tamaños preseleccionados por mí).

- Pausar la imagen de una o varias cámaras (una por una).

- Quitar o poner sonido y volumen individualmente a cada cámara (aunque solo se podrá oír la seleccionada actualmente) y globalmente al conjunto.

- Modificar el “zoom” o el “pan” de una cámara, o incluso restituir totalmente la imagen como estaba en un principio.

- Recuperar automáticamente el espacio de trabajo tal como lo tenía en su última sesión.

- Tener toda información posible respecto a todo evento o problemas que surjan: información de estado de cada cámara, información de la hora actual, confirmación cuando se realicen cambios en la configuración, confirmación al salir de la aplicación, etc.

Si hubiera tiempo estaría muy bien añadir la opción de grabación de cámaras.

2.3.2. No funcionales

Los siguientes son los requisitos no funcionales, en términos de Hardware y Software, que debe cumplir la máquina donde se ejecutará la aplicación:

- Requerimientos mínimos software:
    - Sistema operativo Windows de escritorio compatibles con el Framework indicado: Windows XP, 2000, Millenium, Vista, 7 y 8. El sistema operativo deberá estar al día en actualizaciones de software. Se aconseja para ello el software “Microsoft Windows Update”.

    - Librerías gráficas DirectX al día. Están disponibles en la Web del fabricante:

http://www.microsoft.com/es-es/download/details.aspx?id=35

- Las cámaras locales conectadas al ordenador que se quieran utilizar, deben usar sus drivers originales en su versión más corriente (disponible en la Web del fabricante de la cámara o del ordenador si está integrada).

  Atención: Es posible que una cámara conectada físicamente no sea totalmente compatible con la captura desde mi software, especialmente si no se dispone de drivers adecuados. Un ejemplo de esto es el uso de Windows en una máquina Mac, que aún disponiendo de arquitectura Intel no tendrá un driver 100% adecuado para todos los casos).

- Requerimientos mínimos hardware serán :
  - El ordenador donde se instale el software y su tarjeta gráfica deben tener suficientes recursos y potencia para poder gestionar las cámaras y su visualización. Para casos extremos de equipos antiguos o poco potentes (menos de 1 GHz de potencia o menos de 1 GB de RAM o tarjeta gráfica integrada) se recomienda solo visualizar 1 ó 2 cámaras. En cualquier caso y si no se tienen conocimientos suficientes de los datos del hardware, se empezará por utilizar una única cámara y se irá aumentando el número de cámaras según se vea conveniente.

  - Para las cámaras IP se requiere red o cableada o WIFI.

  Atención: En el caso de redes WIFI es imprescindible que se cuente con protección en la red a través del uso de Firewalls convenientemente configurados y del uso de encriptación WPA-II como mínimo, ignorando versiones anteriores (como WEP y WPA-I).

2.4. Funcionalidades a implementar

2.4.1. Monitorización – Vigilancia de cámaras

El modo monitorización es el modo por defecto en que el programa se inicia. En él, se podrán acceder a las demás secciones de botones. Principalmente en esta vista el acceso más importante durante la vigilancia será visualizar un grupo de monitores que reflejan las imágenes de una o varias cámaras a la vez (ver sección vistas abajo).

2.4.2. Monitorización - Selección de cámaras a monitorizar

En el modo monitorización se podrá acceder a una botonera con hasta 16 cámaras donde activar o desactivar su visualización en la parrilla de vista, su estado se hará evidente a través de distintos colores (aquí a modo de ejemplo gráfico el estado “cámara activada” en verde, el “no activada” como rojo y en gris el “no configurada” –por tanto no activable-):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

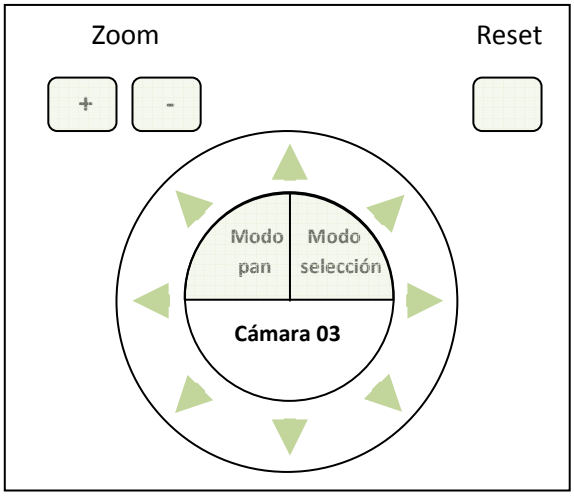
2.4.3. Monitorización – Uso de cámara

La cámara poseerá una serie de funciones a las que tendremos accesos a través de un controlador visual. Para poder realizar cualquiera de estas funciones es necesario seleccionar una cámara (haciendo clic en ella sobre la vista).

Dos modos (excluyentes entre sí):

- Modo selección. En el diseño de abajo se permite activar a través del botón “modo selección”. Permite usar las flechas –abajo en el ejemplo de diseño en color verdes- para seleccionar una cámara de entre las que haya en la vista.

- Modo “pan” (modo de efecto panorámico). Se podrá iniciar el modo a través del botón con ese nombre, donde gracias a las flechas verdes del diseño se dará la posibilidad de movernos a través de la imagen (arriba, izquierda, derecha, abajo y sus derivados básicos). Esto solo es factible si la imagen se está visualizando a un tamaño menor del que realmente tiene o si se ha realizado un zoom previamente.



Efectos:

- “Zoom”: Este modo nos permitirá alejarnos o acercarnos a la imagen.

- “Pan”: Este modo nos habilita cuando no se ve toda la imagen, desplazar la parte de la imagen que vemos.

- “Reset”: Permitirá eliminar todo efecto “zoom” o “pan”.

2.4.4. Monitorización - Selección de vista de cámaras

En el mismo modo que en el apartado anterior, tendremos acceso a una botonera que permite elegir el tipo de vista de cámaras, es decir, el tipo de agrupación y tamaño en que veremos las cámaras. Un ejemplo sería la siguiente vista que permite ver 7 cámaras siendo 3 de ellas mayores que el resto:

<table>
<tr><td rowspan="2">1</td><td colspan="2">2</td></tr>
<tr><td rowspan="2">3</td><td>4</td><td>5</td></tr>
<tr><td>6</td><td>7</td></tr>
</table>

La siguiente vista permitiría ver 4 cámaras al mismo tamaño:

1	2
3	4

Las vistas pre-configuradas serán (por número de cámaras visibles): 1, 4, 7, 9, 10, 13 y 16.

2.4.5. Configuración - Cámaras

La interfaz principal dispone de un botón desde el cual acceder a la configuración de cámaras, donde se deberá permitir configurar cada una de las 16 cámaras preexistentes (marca, modelo, URL, tamaño de caché, etc.). Esta sección de configuración aparece automáticamente en el inicio del programa siempre que no se haya configurado ninguna cámara.

En la misma ventana de configuración se mostrará una pre-visualización de la cámara configurada para poder validar que sea correcta.

2.4.6. Configuración - Opciones generales

Existirá una sección de opciones generales donde podremos modificar ciertos aspectos genéricos, como el idioma de la interfaz o una opción para evitar que el ordenador entre en modo descanso (“idle”).

2.5. Resultados esperados

El objetivo es facilitar la monitorización de zonas a través del uso de cámaras IP y/o cámaras conectadas localmente.

Para ello es vital que todas las funcionalidades estén disponibles para que nos permitan configurar las cámaras, seleccionar las cámaras que queremos ver activas, también la vista que más nos interese –según el número de cámaras a ver o el tipo de vista que queramos-, realizar ciertos efectos para estudiar las imágenes (como el “zoom” y el “pan”) y poder tener una agradable experiencia táctil lejos de las habituales interfaces cuadriculadas.

2.6. Productos obtenidos

Se obtuvo el software que se requería, en dos formatos:

- **Instalador**. Ficheros:
  - IP-VideoSurveillance-Installer.msi
  - Setup.exe

  Una vez instalado el producto en la carpeta de destino tendremos el mismo producto que en el formato de **aplicación portátil**.

- Formato de **aplicación portátil** que no necesita instalación previa. Ficheros y carpetas:
  - Librerías propias compiladas: CameraController.dll, CameraView.dll, Common.dll, Data.dll, IPCameraControl.dll y SkinnedControls.dll.
  - Librerías de terceros: AForge y VlcDotNet.
  - Aplicación compilada “IP-VideoSurveillance.exe”.
  - Carpetas:
    - Diary: contiene el diario de eventos de usuario.
    - lib: con librerías de terceros, recursos open source y licencias de ellos.
    - config: las configuraciones ya indicadas:
      - “cameraConfig”: que guarda la configuración de las cámaras configuradas.
      - “cameraDatabase”: que contiene la base de datos –de solo lectura- de cámaras disponibles en remoto.

- “cameraViews”: con las configuraciones de cámaras y efectos en las vistas.

- “languages”: donde se guardan los idiomas en que podemos tener la interfaz gráfica.

- “skins”: que almacen tanto las distintas configuraciones gráficas posibles como las imágenes asociadas.

2.7. Planificación inicial vs planificación final

La planificación en número de horas ha sido aproximadamente la esperada según el epígrafe “evaluación de costes” de la implementación que se leerá más adelante. Solo he superado en unas 8 horas las 98 horas que debía cumplir en total (este tiempo no incluye ni la confección de esta memoria, ni de la presentación virtual ni del debate). Hay que tener en cuenta que al final no he podido seguir totalmente el horario previsto para el proyecto teniendo que ajustarlo según mi tiempo libre tras mi jornada laboral diaria.

Me he ido encontrando dificultades e impedimentos que han ido retrasando puntos y no permitiendo que otros tuvieran suficiente tiempo de desarrollo en código. Estos cambios se detallan en el epígrafe “reflexiones y conclusiones”.

3. Organización del proyecto

3.1. Hardware necesario

No tengo que construir hardware ninguno, pero me gustaría indicar el banco de pruebas hardware al que voy a someter mi programa:

- Dispongo de las cámaras integradas de un par de ordenadores que poseo (con Windows 7 y 8.1).
- Cuento además con todo tipo de cámaras IP en abierto en internet.
- Intentaré instalar un software especial en mi Smartphone Android para permitir usarlo como cámara IP.

3.2. Software utilizado

La siguiente es una lista de módulos software a desarrollar y sujeta a un análisis más profundo:

- El programa para Windows que permite la monitorización de cámaras en .NET. Implementando toda la funcionalidad antes comentada relativa a:
    - Vigilancia de cámaras (monitorización normal). Estructura en general.
    - Selección de cámaras.
    - Uso de cámara.
    - Selección de vista de cámaras.
    - Configuración de cámaras.
    - Opciones generales.

  Además, los siguientes módulos que –aún formando parte de uno o varios de los anteriores- merecen una especial atención y dedicación:

    - Módulo cámara. Contendrá toda la lógica necesaria para su funcionamiento autónoma a partir de parámetros de entrada y métodos de control.
    - Controles y módulos que permitan “skinning”, o sea, controles gráficos con estética personalizada.
    - Todo trabajo de desarrollo de software conlleva cierto esfuerzo también en cuidar la interfaz a nivel estético e intentar que todo siga una organización limpia y lógica. En este proyecto hay que incluir de manera extraordinaria el diseño gráfico necesario para el sistema estético, “skinning”.

    - Persistencia de datos:
        - Histórico de acciones acaecidas (fichero secuencial).
        - Configuración general a través fichero XML.
        - Configuración de cámaras a través fichero XML.
        - Configuración estática de “skinning” (fichero XML)
        - Ficheros estáticos que guardan los distintos idiomas disponibles (XML).

- Opcionalmente y si el tiempo lo permite un servicio para Windows para el reenvío de la señal de una cámara local:
    - Módulo de gestión del servicio.
    - Módulo “forward camera” que permite capturar desde una cámara y reenviar el “streaming” como si fuese una cámara IP.
    - Interfaz gráfica en forma de icono que se instala en la bandeja del sistema (junto al reloj de Windows) desde donde se podrá configurar las cámaras.
    - Persistencia de datos:
        - Histórico de acciones acaecidas.
        - Configuración de cámaras a través fichero XML.

- Opcionalmente y dependiendo de si hay tiempo suficiente, se permitirá grabar las imágenes de las cámaras.

3.3. Arquitectura del proyecto

Los siguientes son los puntos que tendremos en cuenta en la Arquitectura del desarrollo:

- Estudio técnico y viabilidad del proyecto.
- Planificación del proyecto.
- Análisis y desarrollo de la documentación.
- Implementación, desarrollo.
- Pruebas (incluidas en el desarrollo).
- Documentación.

- Elaboración presentación virtual.

3.4. Definición de roles

Los roles principales asumidos serán: director de proyecto, analista, programador, diseñador gráfico, verificador y documentador.

En este proyecto todos los asumiré yo, no los diferenciaremos ni los agruparemos de manera especial en la planificación de tareas, debiendo en cada caso asumir cada una de las funciones de la forma en que lo haría cada especialista.

3.5. Análisis de riesgos

Hay que hacer un pequeño inciso en este desarrollo para hablar de si es realmente posible desarrollar este proyecto y en el plazo indicado.

3.5.1. Hardware

Cuando tratamos con el desarrollo de software de gestión de hardware, como es nuestro caso al tener que conectar con cámaras locales e IP, nos enfrentamos a todo un mundo de máquinas y drivers que -se suponen- siguen ciertos estándares y un buen uso de sus interfaces, pero la realidad no siempre es así.

Y es que es muy corriente hoy en día encontrar en entornos caseros o profesionales todo tipo de problemas relacionados con cámaras; por ejemplo que no funciona bien en programas tan robustos como Skype (el software de comunicación por video-conferencia por excelencia) si usas WIFI o simplemente si tu Windows 8 no acaba de tener el driver más adecuado para conectarlo… Ante ese tipo de problemas las empresas grandes, como a la que pertenece Skype, dedican grandes recursos en mejorar el producto depurando y corrigiendo uno a uno cada uno de sus fallos.

Pero yo no tengo el tiempo y los recursos de las grandes empresas, ni siquiera el de una pequeña puesto que tengo que compaginar este proyecto con mi trabajo propio que nada tiene que ver con este tipo de proyectos. Así que tengo que dejar claro que soy consciente que puede que no todas las cámaras locales o IP sean compatibles, pero gracias a que uso adrede tecnología muy robusta (VideoLan como reproductor) debería de cubrir las necesidades de la mayor parte de los usuarios. Intentaré utilizar el software en un parque amplio de equipos y cámaras para poder corregir todo lo posible.

3.5.2. Efectos gráficos

Me he decantado por utilizar una interfaz gráfica poco usual, me alejo de los controles rectangulares y me adentro en el desarrollo de controles propios que permiten transparencias y formas irregulares basadas en diseños propios. Esto tiene un coste, ya que a priori parece bastante sencillo pero no lo es tanto porque:

- Nativamente los controles son rectangulares.
- Las transparencias deben ignorar los eventos habituales y se debe permitir solo a los píxeles opacos provocarlos.
- La superposición entre objetos cuando el de arriba posee transparencias es una cuestión que se dará en ciertas ocasiones y hay que saber tratarla adecuadamente.
- Solo las imágenes de 32 bits se deben de tener en cuenta, ya que son las únicas que poseen canal alfa (el formato GIF no valdría ya que es un color en la paleta y no un canal propio).
- Los tratamientos de imágenes a bajo nivel no son bienvenidos ni compatibles entre las distintas versiones de Windows. Por ello tendré que comprobar que todo es correcto en todo tipo de Windows.
- Hay que vigilar el rendimiento que se consigue y los recursos que se necesitan en caso de tener muchos controles con efectos gráficos.

3.5.3. Recursos de cámara

Las capturas de vídeo que se capturan requieren bastantes recursos, es un tema que dependerá de los recursos del usuario. Yo me limitaré a advertirlo y dar consejos al respecto.

4. Análisis y diseño

4.1. Requerimientos funcionales

4.1.1. Módulo 1 – Monitorización (vigilancia de cámaras)

Requerimiento 1.1: El sistema se iniciará en modo monitorización de cámaras, el cual es un modo pasivo en el que realmente el usuario no necesita realizar operación alguna si las cámaras ya están configuradas. En el caso de no existir ninguna cámara configurada se mostrará la interfaz que permite configurarlas.

Requerimiento 1.2: El software y las cámaras monitorizadas se mostrarán en los monitores que se hayan configurado en las opciones generales.

Requerimiento 1.3: Se proporcionarán botones para acceder a las siguientes funcionalidades –más adelante nombradas- dentro de la misma pantalla principal: selección de cámaras a monitorizar, uso de cámara y selección de vista. También se añadirán botones para acceder a las siguientes funcionalidades que se mostrarán en otra ventana superpuesta a la principal: configuración de cámaras y opciones generales.

Requerimiento 1.4: Por defecto, el sistema no permitirá al Sistema Operativo que entre en modo de ahorro energético (suspensión energética, salvapantallas, etc.) al menos en ninguna de las pantallas donde sea visible.

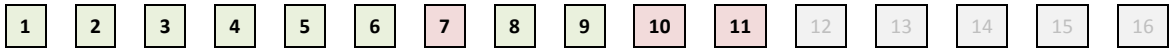
Requerimiento 1.5: El sistema principal y las cámaras asignadas a pantallas independientes se mostrarán en pantalla completa. Por ese mismo motivo se mostrará la fecha y la hora del sistema en la interfaz principal del sistema, además de un resumen del estado de las cámaras.

Requerimiento 1.6: El sistema será multimonitor permitiendo la salida a diferentes monitores de las cámaras configuradas y el propio sistema (uno de ellos por monitor sin repetir). Si el sistema encontrara que uno de los monitores no está disponible lo mostrará como error y preguntará si debe eliminar dicha configuración (y actuar en cada caso).

4.1.2. Módulo 2 – Selección de cámaras a monitorizar

Requerimiento 2.1: El presente módulo formará parte de la interfaz principal de monitorización.

Requerimiento 2.2: Se proporcionarán botones que permitan seleccionar la cámara a visualizar en la vista (ver módulo 4). Las cámaras visualizadas, no visualizadas y no configuradas aparecerán con diferencias visuales para dejar claro su estado fácilmente. Aquí se puede ver una representación de ejemplo donde el color rojo se da a las cámaras no visualizadas, el verde a las que se ven y en gris las no configuradas:



4.1.3. Módulo 3 – Uso de cámara

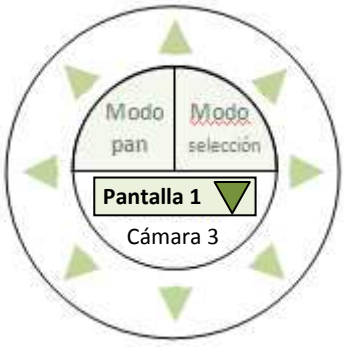
Requerimiento 3.1: El presente módulo formará parte de la interfaz principal de monitorización.

Requerimiento 3.2: Una cámara no configurada no mostrará imagen alguna y sus controles estarán desactivados.

Requerimiento 3.3: Una cámara siempre mostrará su estado sobreimpreso y –si configurado para que sea así en opciones generales- la fecha y la hora.

Requerimiento 3.4: Se proporcionarán controles que permitan:

- Seleccionar una de las cámaras de la vista activa (ver módulo 4). Esta selección se podrá realizar también simplemente haciendo clic sobre una cámara de la vista. El nombre de la cámara seleccionada aparecerá en el mismo controlador, en el ejemplo aparece de bajo aparece como “Cámara 3” y las flechas verde claro son los seleccionadores de cámara. Este modo se llamará “Modo selección”.

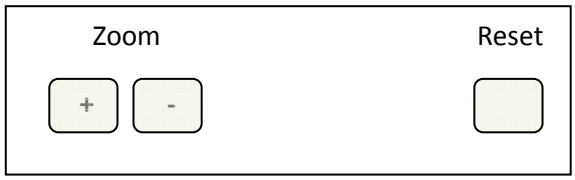


- Permutar entre el modo selección de cámara y el panorámico. Llamamos “Modo panorámico” al que permite desplazar el centro de la imagen actual a través de los selectores del modo selección (para ello se debe previamente haber realizado zoom). En el ejemplo aparecen dos botones que permiten elegir uno u otro modo.

- Seleccionar una pantalla distinta de entre las configuradas.

Requerimiento 3.5: El programa principal tendrá todos los controles habilitados excepto para las cámaras visualizadas en pantalla completa exclusiva –fuera del entorno principal - donde desaparecerán los botones selectores y el botón de modo de selección (entrando automáticamente en modo panorámico). En el ejemplo anterior se puede ver el selector de pantalla con la “Pantalla 1” seleccionada, pudiéndose elegir ahí otra pantalla como la actual.

Requerimiento 3.5: Se mostrarán controles para poder alejamiento o acercamiento de una imagen (zoom), además de otro para reinicializar la imagen y dejarla sin efectos de zoom en ella.



Requerimiento 3.6: En el modo zoom si tratamos con una cámara que pertenece a la vista del programa principal, se centrará la ventana donde se visualiza la cámara para poder verla casi a pantalla completa (ocupando la mayor parte posible de toda la zona de cámaras del programa principal).

Requerimiento 3.7: Se proveerá un controlador general de volumen que afectará a todas las cámaras.

4.1.4. Módulo 4 – Selección de vista de cámaras

Requerimiento 4.1: El presente módulo formará parte de la interfaz principal de monitorización.

Requerimiento 4.2: El usuario podrá seleccionar entre las distintas vistas de cámaras. Vista es sinónimo de estructura donde se muestran una o más imágenes capturadas por cámaras. Se permitirá seleccionar una de las vistas disponibles a través de botones que tendrán una

representación gráfica sencilla del resultado que se obtendrá. Un ejemplo éste donde se muestran 7 cámaras, siendo 3 de ellas mayores que las demás:

<table>
<tr><td rowspan="2">1</td><td colspan="2">2</td></tr>
<tr><td rowspan="2">3</td><td>4</td><td>5</td></tr>
<tr><td>6</td><td>7</td></tr>
</table>

Requerimiento 4.3: Cada vista almacena una grupo de cámaras distinta entre vistas. Así, aunque en una vista se use –por ejemplo- la cámara 01 no tendrá por qué utilizarse ésta misma en otras vistas ya que son vistas independientes.

Requerimiento 4.4: Toda la configuración realizada tendrá persistencia a través del fichero XML de configuración general.

4.1.5. Módulo 5 – Configuración - Cámaras

Requerimiento 5.1: El presente módulo formará parte de una capa que se superpondrá en la interfaz principal de monitorización e impidiendo su uso.

Requerimiento 5.2: Se podrán configurar los datos necesarios para acceder a cada una de las 16 cámaras y el nombre a mostrar. Inicialmente, el nombre mostrado por cada cámara será una numeración basada en su posición respecto de la lista de 16 cámaras disponibles, por ejemplo “Cámara 01” para la primera. El acceso configurable incluirá:

- Para cámaras IP: marca, modelo, dirección IP, número de dispositivo, puerto, protocolo, usuario, contraseña, resolución, volumen, nombre y si está activada o no. Se permitirá introducir una configuración propia.

- Para cámaras locales: cámara utilizada, resolución, volumen, nombre y si está activada o no.

Requerimiento 5.3: Se podrán configurar si se muestra la fecha y la hora en las cámaras.

Requerimiento 5.4: En la misma ventana de configuración se mostrará una pre-visualización de la cámara a configurar para poder validar que sea correcta.

Requerimiento 5.5: Toda la configuración realizada tendrá persistencia a través del fichero XML de configuración general.

4.1.6. Módulo 6 – Configuración – Opciones generales y pantallas

Requerimiento 6.1: El presente módulo formará parte de una capa que se superpondrá en la interfaz principal de monitorización e impidiendo su uso.

Requerimiento 6.2: Se podrán configurar si el Sistema Operativo puede o no entrar en modo de ahorro energético (suspensión energética, salvapantallas, etc.).

Requerimiento 6.3: El idioma por defecto del software será el del propio Sistema Operativo siempre que se disponga de traducción del mismo, sino será el inglés. Se permitirá elegir idioma desde esta interfaz. En principio se dispondrá de castellano e inglés. Los ficheros de idiomas serán externos facilitando así el mantenimiento de las traducciones y de la futura adicción de nuevos idiomas.

Requerimiento 6.4: Se podrá configurar la interfaz de usuario a través de la selección de “skin”.

Requerimiento 6.5: Se podrán configurar las pantallas donde serán visibles el software y las cámaras. Para ello se mostrará una estructura donde se representan todas las pantallas disponibles y en ellas se podrá elegir si ver una cámara de las disponibles o el sistema mismo.

Requerimiento 6.6: Se habilitará un botón que permita identificar la pantalla que estamos configurando.

Requerimiento 6.7: Toda la configuración realizada tendrá persistencia a través del fichero XML de configuración general.

4.1.7. Módulo 7 – Generalidades

Requerimiento 7.1: El sistema ofrecerá un entorno gráfico amigable especializado en pantallas táctiles.

Requerimiento 7.2: Los errores y advertencias más importantes serán mostrados al usuario.

Requerimiento 7.3: Los eventos más importantes (incluyendo los no mostrados al usuario como mensaje) se guardarán en ficheros planos de tipo CSV a fin de: permitir realizar un análisis en caso de problema, querer realizar un posible seguimiento de todo lo que ocurre o incluso dejar la puerta abierta a un eventual tratamiento automatizado de los eventos que ocurren en el mismo (ya que los ficheros CSV son fácilmente automatizables).

4.2. Requerimientos no funcionales

4.2.1. Módulo 8 – Librerías comunes

Requerimiento 8.1: El presente módulo formará parte del grupo de utilidades comunes e internos del sistema, incluyendo las llamadas comunes a librerías comunes del Sistema Operativo.

Requerimiento 8.2: Incluirá todo lo relacionado con la persistencia de datos (ficheros de seguimiento, de configuración (cámaras, opciones y vista actual incluidas), de idioma y de “skin”.

4.2.2. Módulo 9 – Librerías y componentes “skinning”

Requerimiento 9.1: Este módulo interno conformará el aspecto gráfico no estándar del sistema: botones no rectangulares, imágenes transparentes como botones de cualquier tipo de forma, gestión de eventos a través de transparencias, formas suficientemente grandes y orientadas al uso mediante pantallas táctiles, etc.

Requerimiento 9.2: Se ofrecerán por defecto un “skin” moderno, pero la librería permite que el usuario realice el suyo de manera sencilla. Se le indicará en el manual de uso cómo realizar el suyo propio, dónde ubicarlo y cómo configurar la interfaz para que use dicho “skin”.

4.2.3. Módulo 10 – Diseño gráfico

Requerimiento 10.1: El grafismo tendrá una especial consideración y se verá muy cuidado el diseño de los componentes interactivos visuales (botones, listas, formularios y otros), especialmente grandes orientados a pantallas táctiles, sin dejar de lado el uso de ratón.

Aunque parezcan el uso de ratón y del dedo en una pantalla táctil parecen sinónimos no lo son tanto debido a que en Sistemas Windows 7 y superiores se tratan de manera diferente e incluso se pueden utilizar gestos o más de un dedo. Además, los movimientos de un dedo pueden llegar a ser más sofisticados y los del ratón más exactos, por ello los diales (como los de las radios antiguas) son sencillos de usar por dedos e insufribles para ratones. Otro ejemplo sería el del típico control .NET “spin” que suele tener controladores pequeños donde un ratón no tiene mayor problema pero que un dedo -por impreciso- apenas podrá controlar.

4.2.4. Módulo 11 – Librerías y componente para cámaras

Requerimiento 11.1: Este módulo interno común incluye utilidades para el tratamiento de imágenes y protocolos de “streaming” y reproducción; además de un componente .NET propio que permitirá gestionar de manera transparente todo lo relacionado con el “streaming” entrante y visualización del mismo en él (utilizado luego en la vista de monitorización).

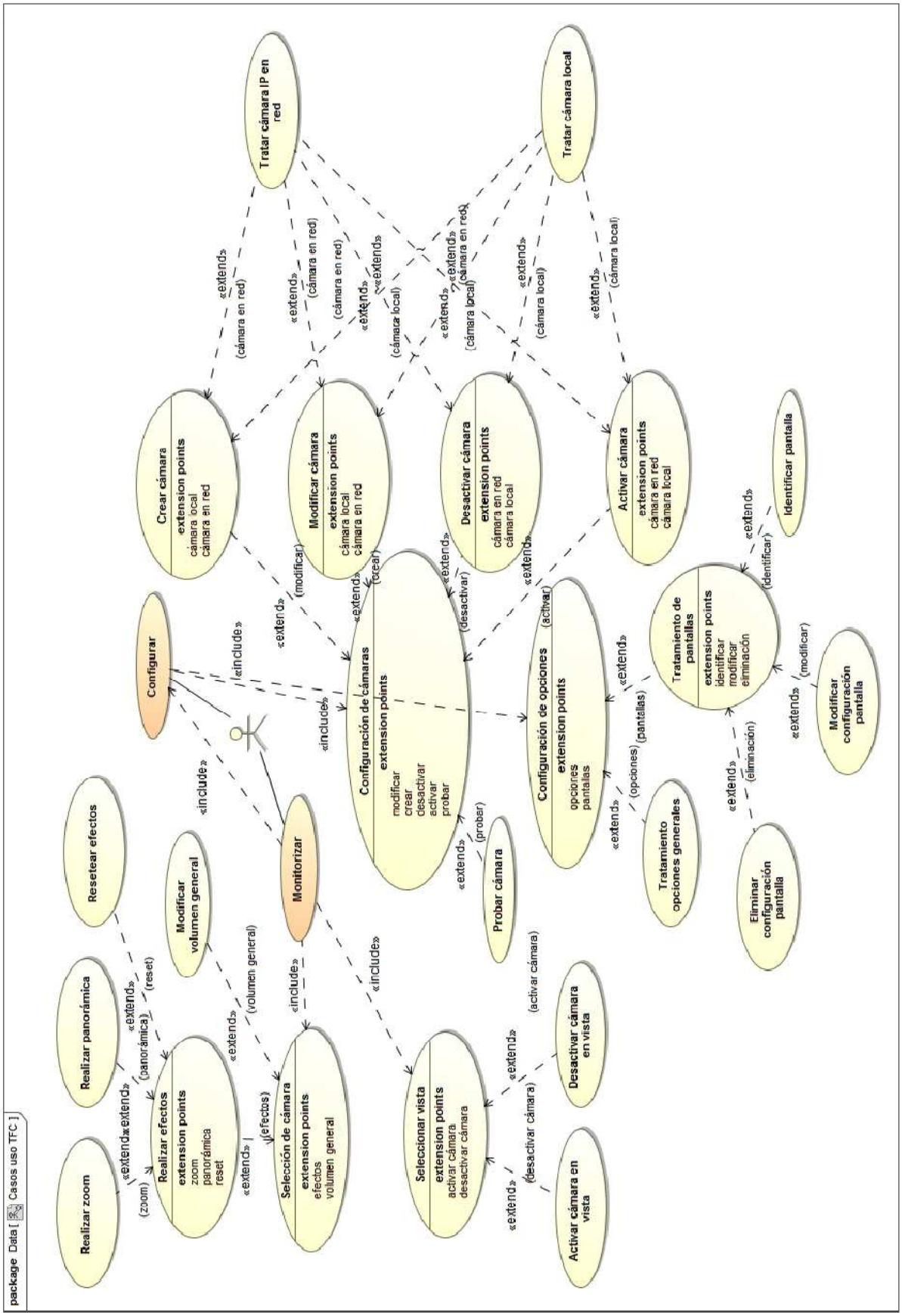
4.2.5. Módulo 12 – Documentación

Requerimiento 12.1: Se ofrecerá dos manuales especializados para simplificar el aprendizaje y su uso:

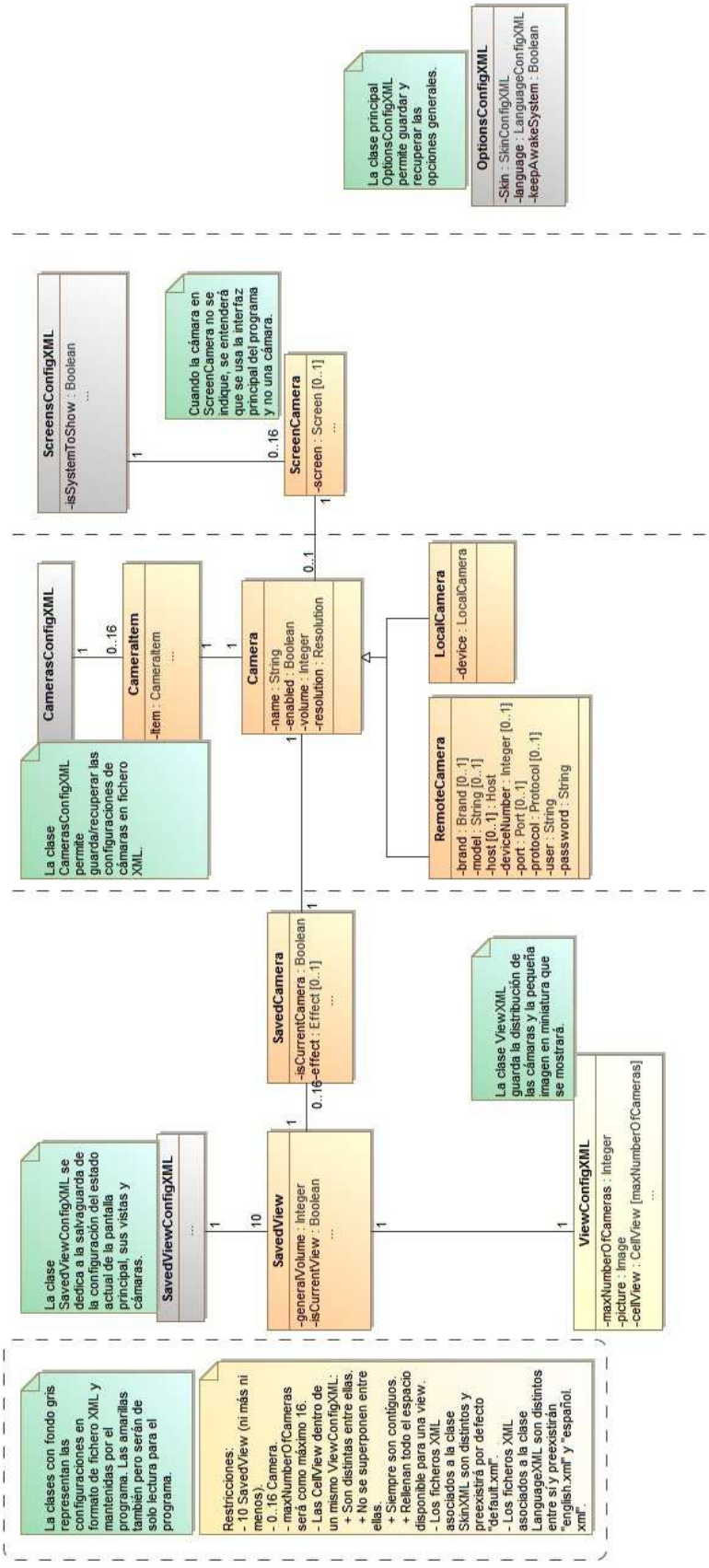
- Un manual de uso rápido orientado al usuario final del software, el usuario que monitoriza las cámaras incluyendo una guía rápida de posibles fallos y soluciones.

- Un manual más técnico orientado a la configuración del software.

4.3. Diagrama de casos de uso



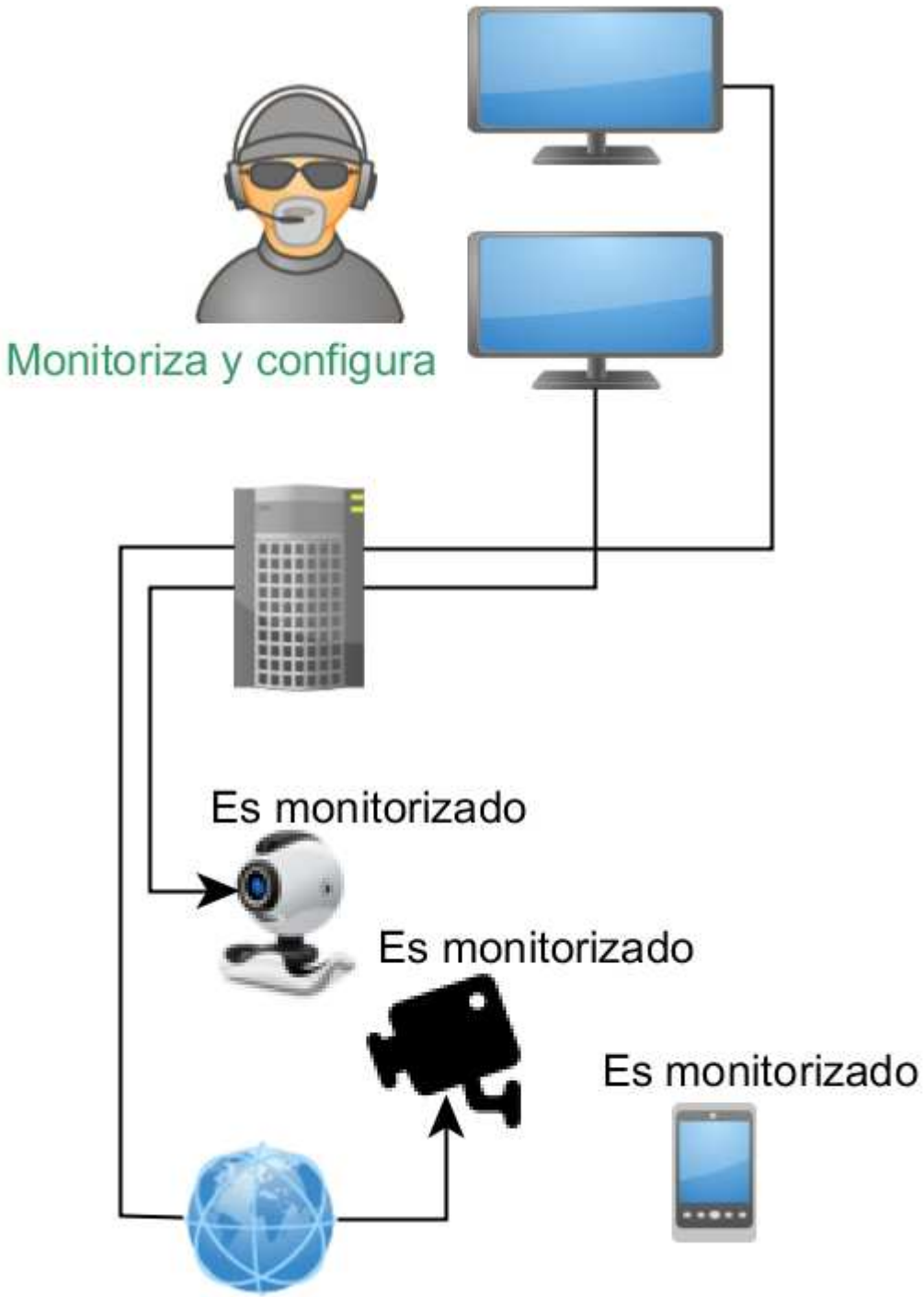
4.4. Modelo conceptual



4.5. Diagrama de arquitectura

4.5.1. Aplicación

La aplicación consistirá en un sistema software que se ejecuta sobre un ordenador con Windows y que permitirá monitorizar la captura de cámaras locales e IP a través de una o varias pantallas. Las cámaras IP debido a su sistema de comunicación serán no solo accesibles a través de red local sino también Internet.



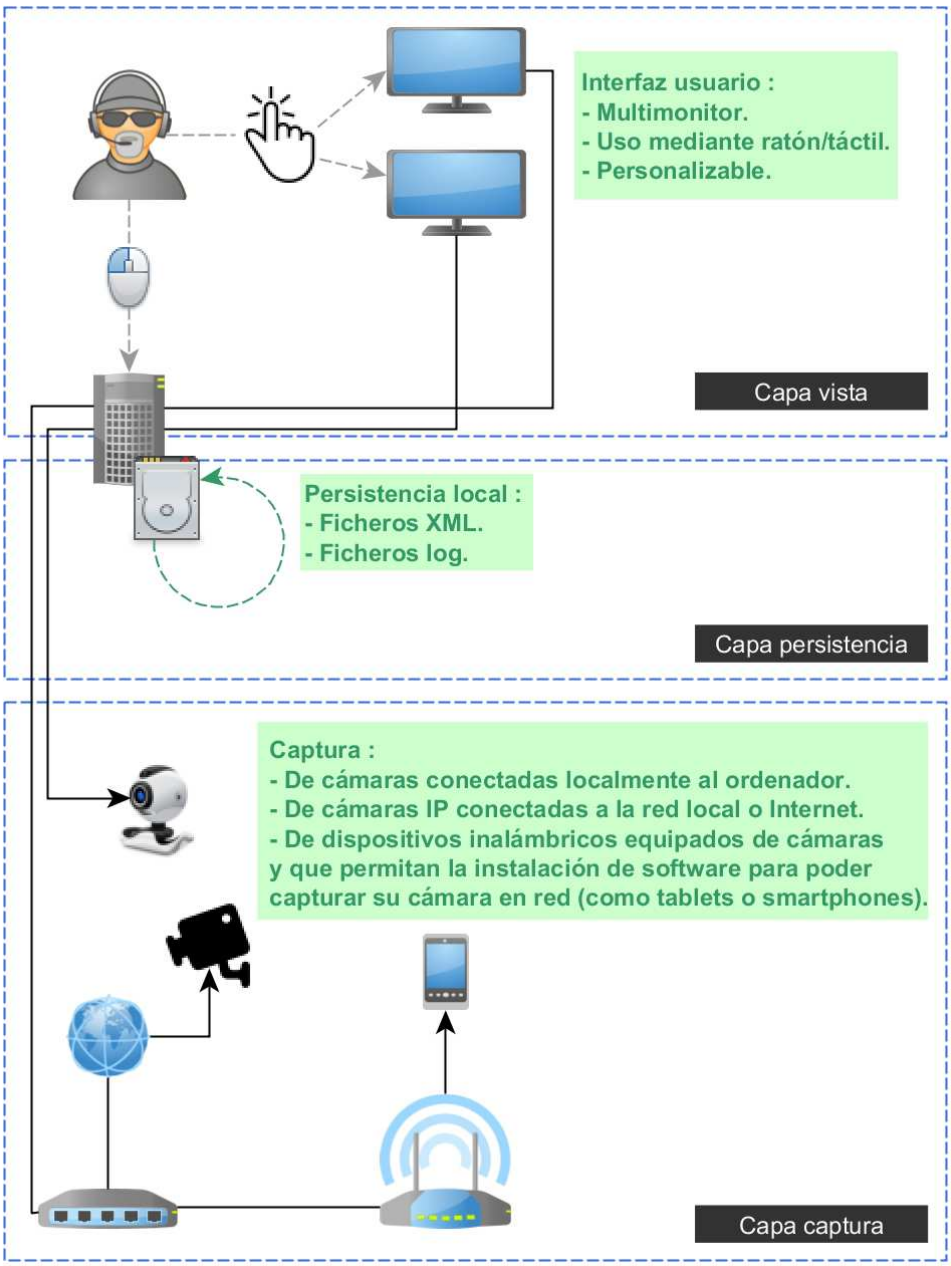
4.5.2. Software

No tratamos un modelo MVC, aún así podríamos separar en las siguientes capas software –diferenciadas también en el mismo código-:

- Vista. Es el conjunto formado por la interfaz gráfica, incluyendo componentes propios y sistema para proporcionar “skins” al entorno.

- Persistencia. Podríamos llamarla datos pero es algo menos completa pues solo incluye la parte determinada y conocida (ficheros XML y de seguimiento –o log-).

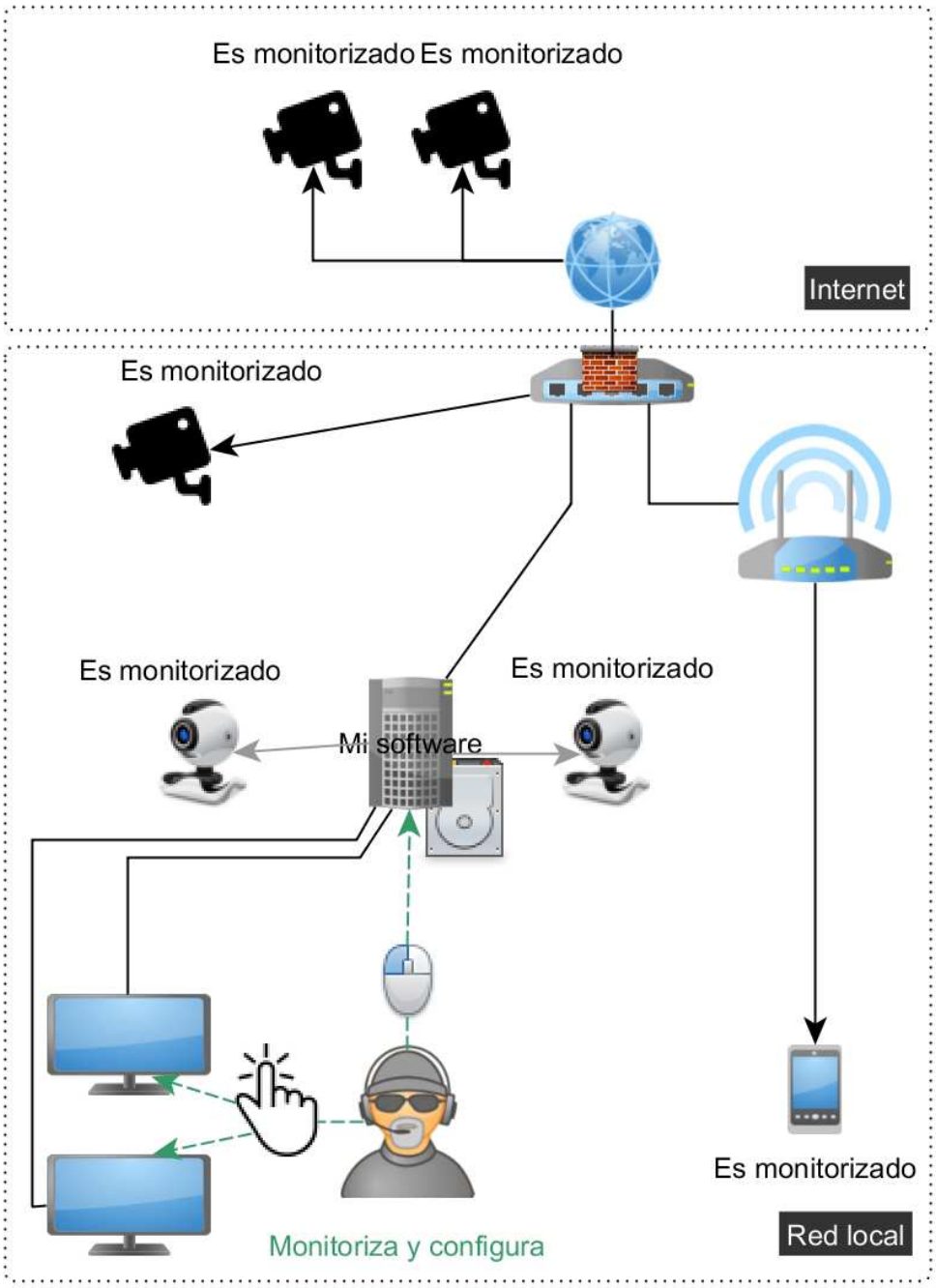
- Captura. Se refiere a la adquisición de vídeo/imágenes y/o sonido de las cámaras en tiempo real a través de RTP, RTSP, HTTP, HTTPS, MMS, MMSH o internamente (local).



4.5.3. Hardware

Desde un ordenador se podrán monitorizar en una o varias pantallas, el vídeo (imágenes y/o sonidos) capturados por toda cámara configurada localmente conectada al equipo y toda cámara accesible mediante IP a través de la red local (cableada o inalámbrica) o internet. Los dispositivos móviles son fácilmente convertibles en cámaras a través de software gratuito instalado en él. No solo se podrá usar el ratón para poder utilizar el programa sino también una pantalla táctil.

El sistema permitirá sin problemas ser utilizado en más de un ordenador pero de manera totalmente independiente entre ellos. Importante configurar debidamente sistemas Firewall.



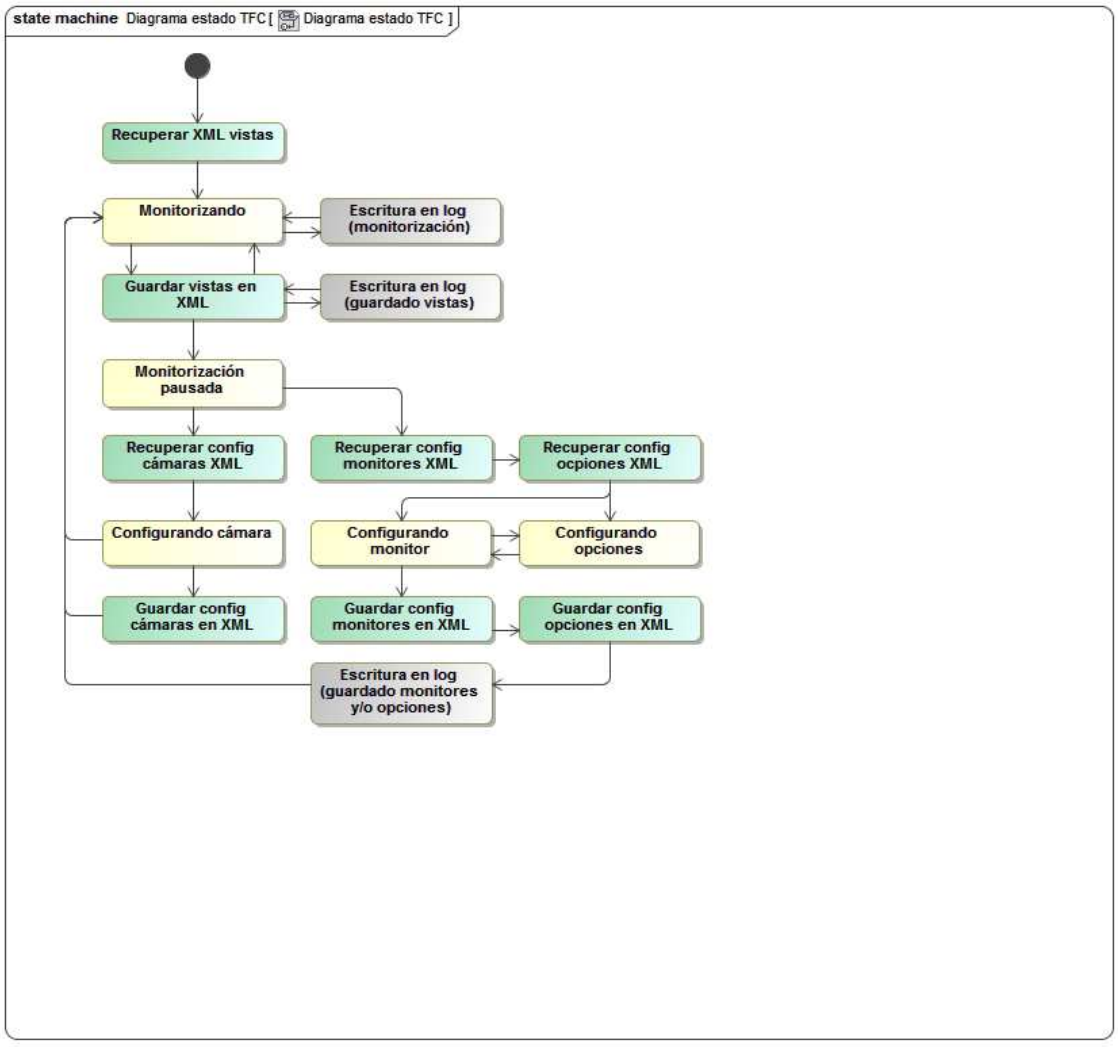
4.6. Diseño de la Base de Datos / Diagrama E-R

Esta aplicación no requiere diseño de base de datos, debido a que la persistencia se realiza en ficheros de texto. Se ha preferido ficheros y no base de datos ya que no se requieren grandes recursos, ni eficiencia ni un sistema multiusuario; además el uso de base de un gestor de base de datos sería un derroche en recursos y complicaciones que este tipo de proyectos no requiere.

Sin embargo, si en un futuro se quisieran implementar un soporte multiusuario u otro tipo de ampliaciones, sí podríamos llegar a pensar en el uso de una base de datos, aunque por supuesto siempre se tendrían que comparar con el uso de alternativas menos pesadas como las bases de datos XML que no requieran instalación.

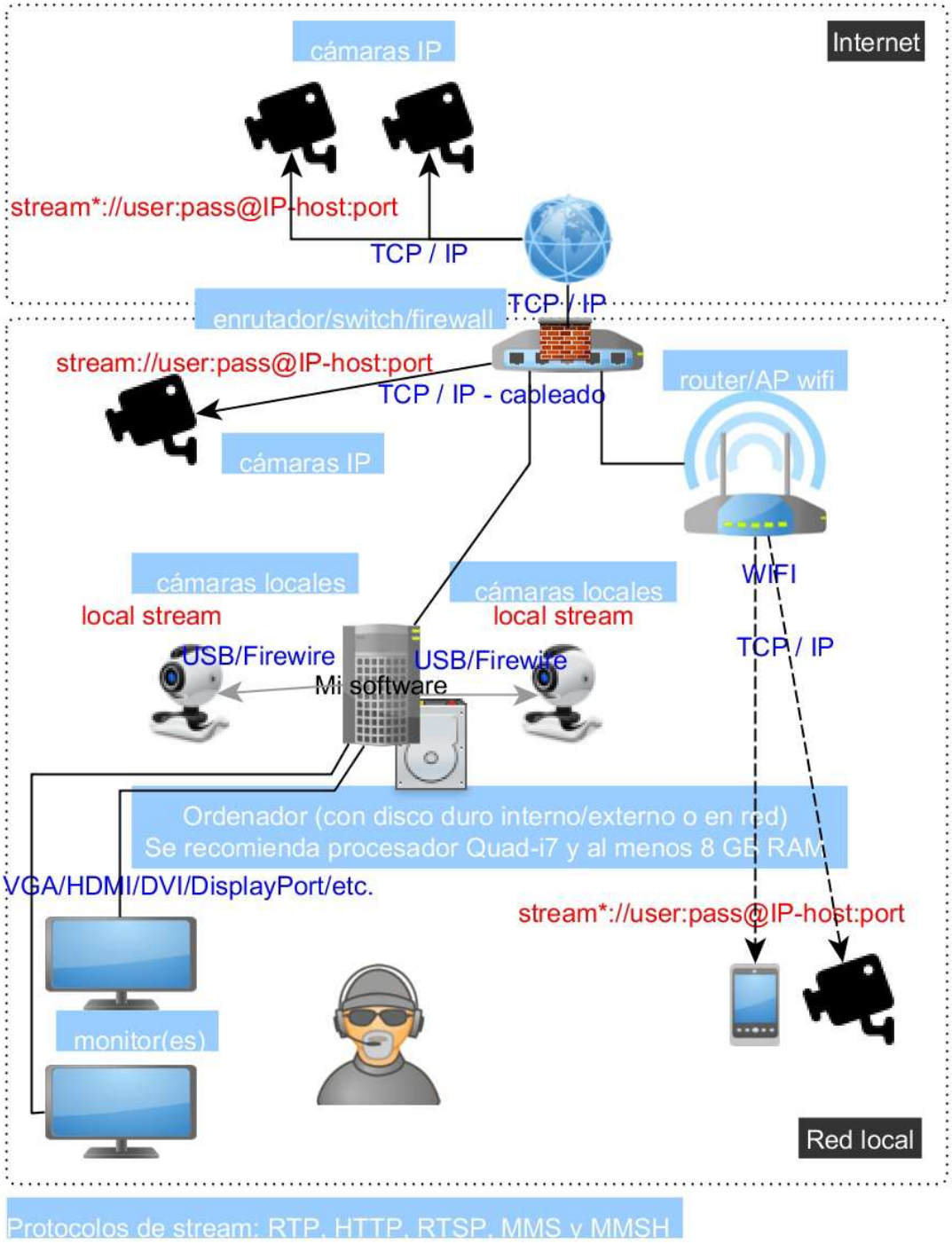
4.7. Diagrama de estado

Se presenta aquí el recorrido por estados más importantes que realizará el usuario (en amarillo), el sistema de persistencia más importante (en verde) para recuperar y guardar la configuración y el sistema de seguimiento/log (en gris).

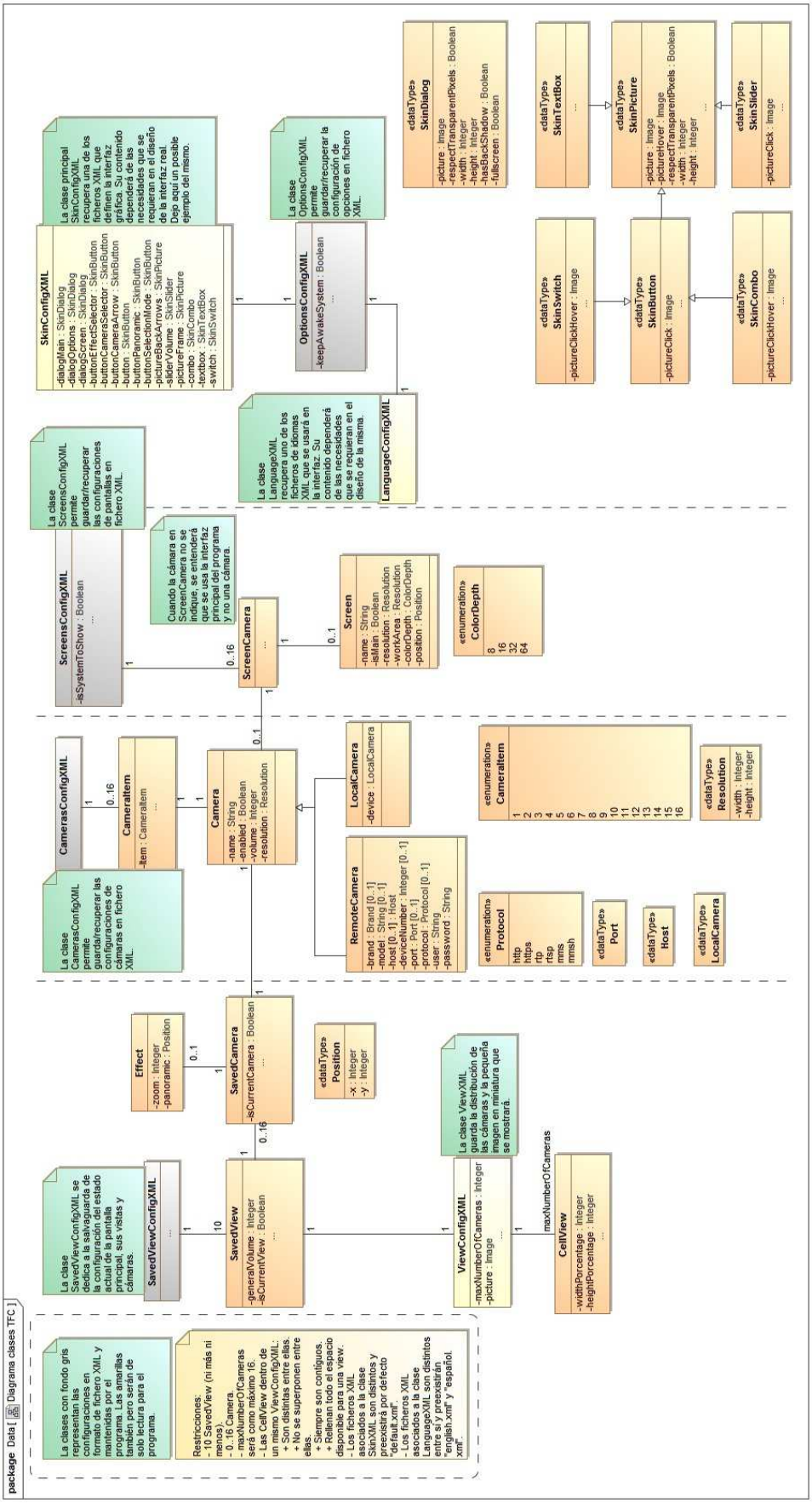


4.8. Diagrama de implementación

El siguiente representa un diagrama de implementación del flujo de comunicación y la arquitectura física del sistema.



4.9. Modelo de clases

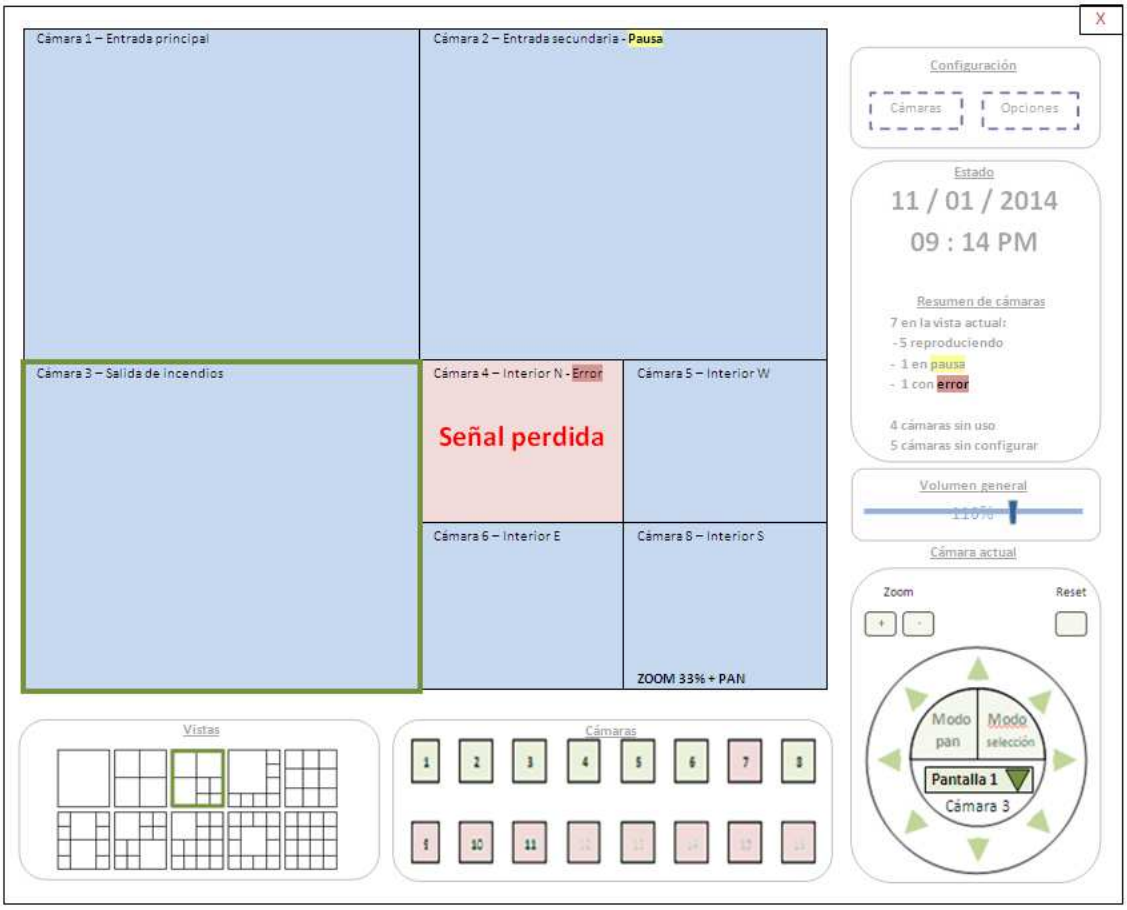


4.10. Diseño de la interfaz de usuario

Debido a que es mis interfaces no están basadas en el estándar típico de objetos rectangulares, no he podido utilizar los diseños de borradores tipo “mockups” que suelo presentar; así que aquí presento maquetas que sirven para poder tener una idea aproximada de los contenidos en un ejemplo de estructura de los elementos que componen las interfaces gráficas. El resultado final dependerá de cómo diseñaré los elementos gráficos.

4.10.1. Modo monitorización

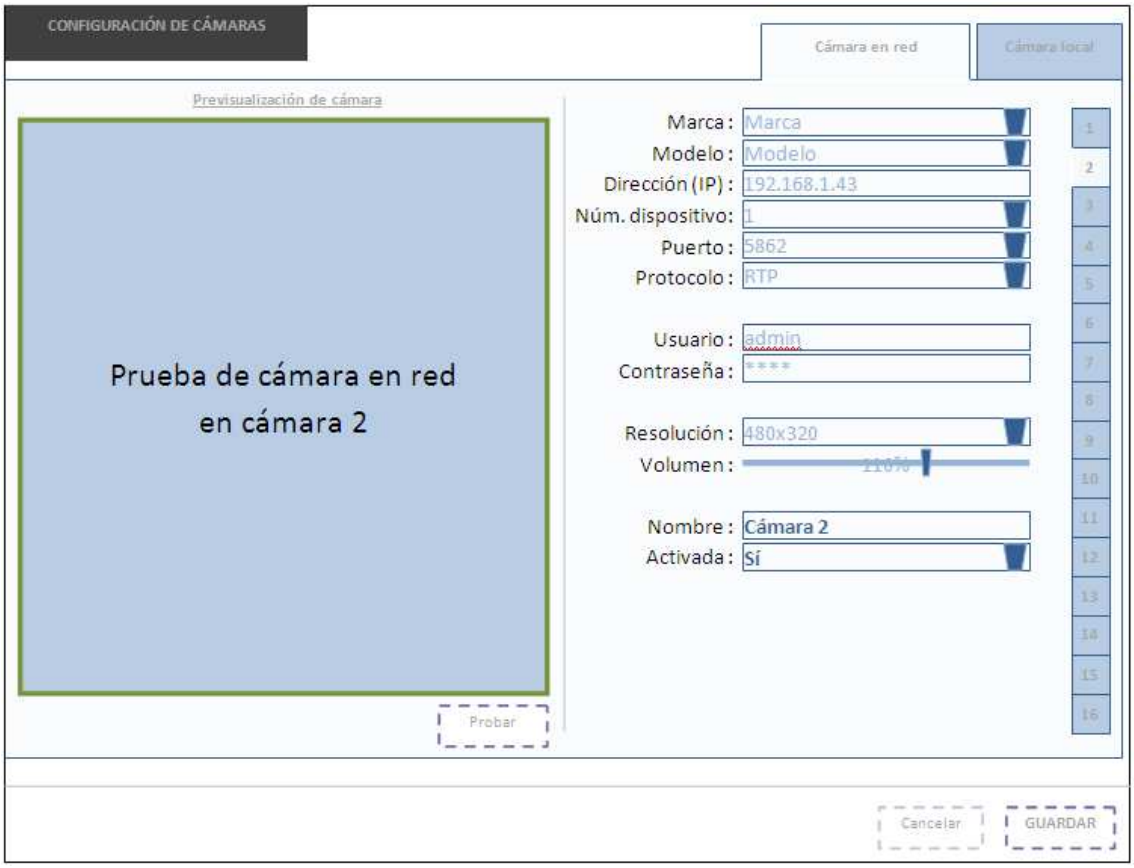
Aquí presento una maqueta del programa principal en modo monitorización, siempre que no se requiera configuración obligatoria alguna. Los recuadros azules representan cámaras en funcionamiento y el rojo con fallo.



4.10.2. Configuración de cámaras

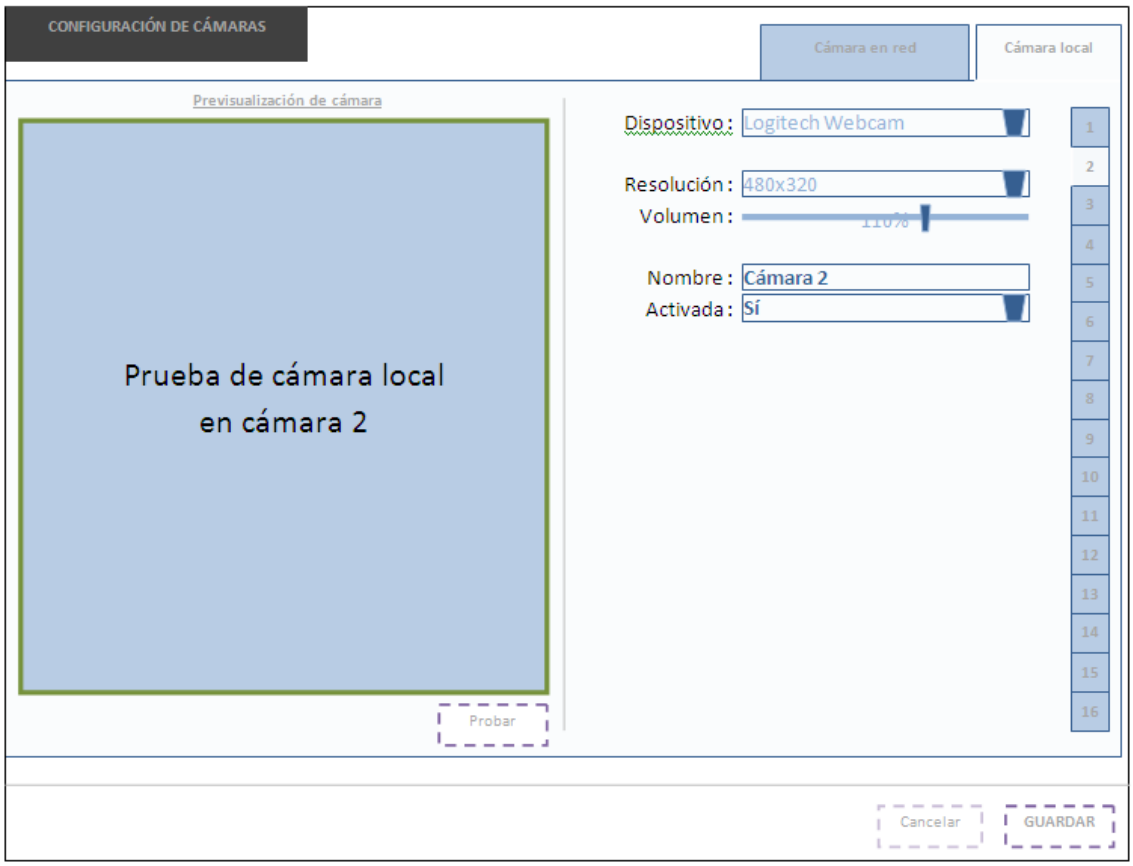
En la siguiente maqueta se representa la ventana de configuración de cámaras a la que se tiene acceso a través del botón habilitado en la pantalla de monitorización. También, es la pantalla inicial que se mostraría en caso de nunca haberse configurado ninguna cámara.

Esta ventana consta de dos grupos: el grupo de los controles que integran la pestaña “Cámara en red” y el de la pestaña “Cámara local”. Aquí se presenta la primera de ellas:



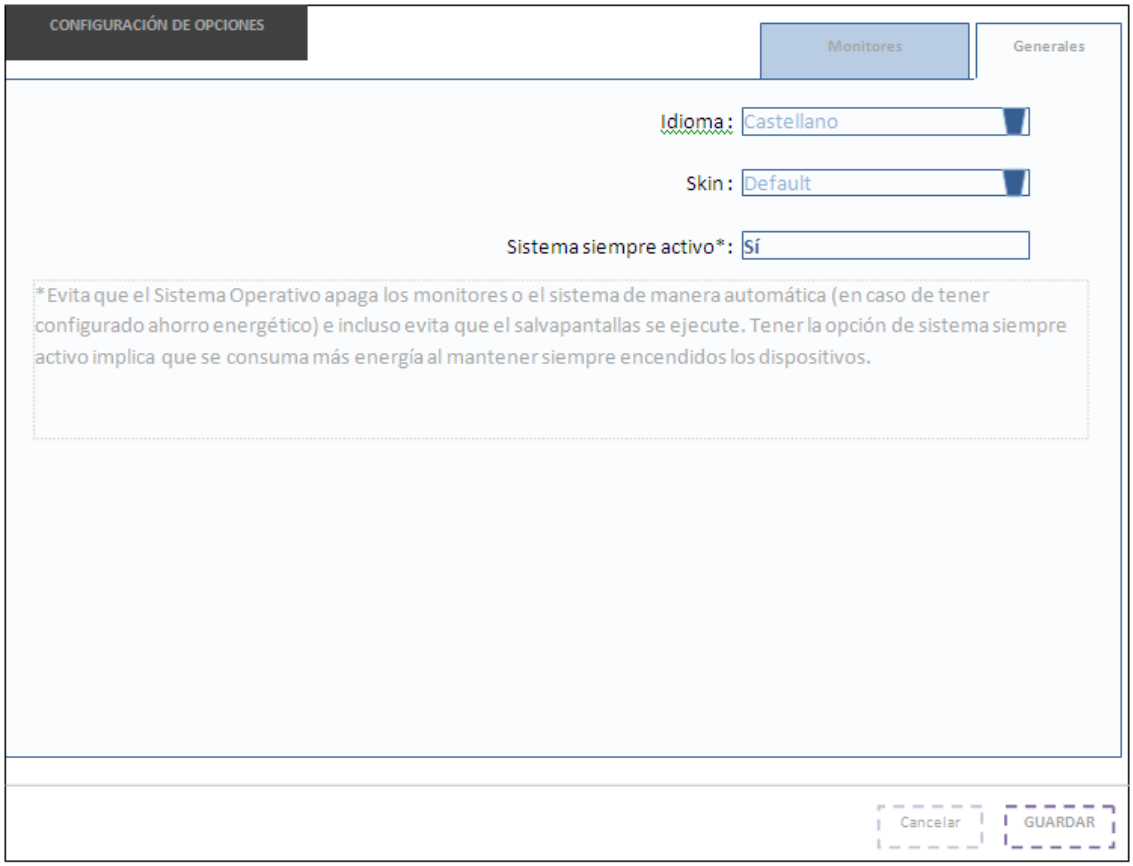
En el caso de seleccionar como marca “personalizar” se deshabilitaría el campo modelo.

Y aquí la segunda pestaña, que permite configurar las cámaras locales (aquí la conectada localmente Logitech Webcam):

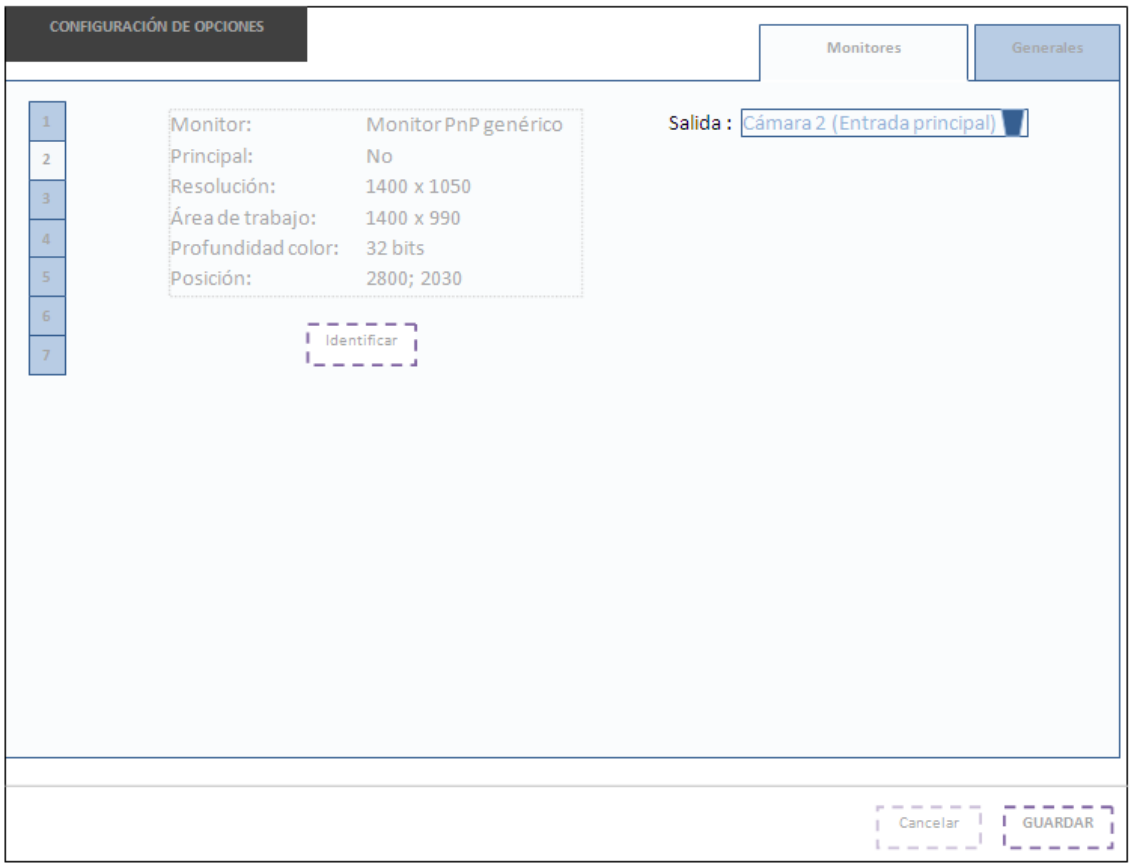


4.10.3. Configuración de opciones

La siguiente maqueta nos muestra la ventana de las opciones generales (pestaña opciones):



Y la siguiente la pestaña de configuración de monitores que permite seleccionar en qué monitores se verá el software y las cámaras:



El uso del botón identificar permitirá averiguar rápidamente con cuál de los monitores físicos estamos tratando al dibujar en dicho monitor durante unos segundos un texto grande en primer plano.

4.11. Riesgos

4.11.1. Hardware

Como ya he dicho en el análisis inicial, para mí los riesgos están marcados en el correcto tratamiento del hardware que en el mercado actual son heterogéneos, tantos modelos de cámara IP y cada uno con maneras distintas de comunicarse. Yo me basaré en el tratamiento de las imágenes (vídeo o estáticas en cadena) que comparten a través de protocolos de “streaming” conocidos, así que la dificultad estará en la recopilación de suficientes tipos de cámaras y sus maneras de conectar, aunque siempre se permitirá al usuario introducir manualmente los datos necesarios (personalizar la conexión) dejándo nuestro camino menos difícil.

4.11.2. Protección de los sistemas

Un tema importante que tendré que mencionar también en el manual de usuario, son los sistemas de protección a nivel ordenador y red: antivirus, firewalls, antimalware y todo tipo de herramientas protectoras pueden impedir o ralentizar el acceso al hardware.

4.11.3. Rendimiento

Algo también que se incluirá en el manual del usuario son los límites del sistema informático ante tanta cámara capturada y visualizada en el mismo. Es algo difícil de calcular y prever y dependerá más bien de la experiencia real.

5. Implementación

5.1. Software utilizado

Las siguientes son las tecnologías que propongo utilizar para realizar mi desarrollo de software:

- Lenguaje de programación utilizado C#.

- Librerías Microsoft Framework .NET 4.0.

- Librerías para permitir acceso a la difusión de vídeo por red (“streaming”) o local:
  - Microsoft DirectShow.
  - AviCap32.
  - VLC DotNet.
  - Reproductor Videolan (VLC).

- Librerías para la visualización de los vídeos:
  - VLC DotNet.
  - Reproductor Videolan (VLC).

- Librerías para enumerar dispositivos locales de vídeo y audio:
  - AForge .NET.

5.2. Capas de la aplicación

5.2.1. Por código

Dentro de los distintos proyectos que componen la solución Visual Studio, hemos separado las 3 capas principales de la siguiente manera:

- Vista. Compone toda la interfaz gráfica reutilizable: controles personalizados y formularios. En los proyectos donde existe esta capa, está representado por sus respectivas carpetas llamadas “View”.

- Modelo. Es la encargada de la definición de persistencia para ficheros XML en forma de XSD y clases para su gestión y de los mensajes y textos personalizables.

- Controlador. Se encarga de la gestión de las vistas, adquisición de datos XML y vía captura de medios multimedia. La carpeta utilizada para almacenar la gestión de controlador se llama –respectivamente por proyecto- “Controller”.

5.2.2. En datos

En el proyecto principal, existen las siguientes carpetas que se copian en el producto final:

- Una carpeta llamada “config” donde se almacenan ficheros XML de persistencia en las siguientes carpetas:
    - “cameraConfig”: que guarda la configuración de las cámaras configuradas.
    - “cameraDatabase”: que contiene la base de datos –de solo lectura- de cámaras disponibles en remoto.
    - “cameraViews”: con las configuraciones de cámaras y efectos en las vistas.
    - “languages”: donde se guardan los idiomas en que podemos tener la interfaz gráfica.
    - “skins”: que almacen tanto las distintas configuraciones gráficas posibles como las imágenes asociadas.

- Una carpeta “Data” que contiene los recursos open source y librerías externas utilizadas (no instalables de otra manera).

- La carpeta “diary” que contendrá el log en formato CSV a nivel usuario con los eventos principales ocurriedos en la aplicación.

- En caso de encontrarse en modo depuración el ejecutable, la carpeta que lo contiene tendrá una carpeta “log” con el log completo de eventos en formato CSV para ayudar al programador.

5.2.3. División en proyectos

- Las carpetas “API” indican la parte común accesible de cada proyecto y la manera más adecuada de encontrar los encontrar métodos más útiles.

- Este proyecto fin de carrera se ha dispuesto en varios proyectos Visual Studio para poder agrupar los recursos en ciertos proyectos y permitir –en la medida de lo posible- su reutilización en futuros proyectos; o simplemente para aislar o facilitar el desarrollo de componentes:
    - “Common”: contiene la API común a los demás proyectos. Permite evitar duplicidad de código.
    - “CommonData”: abarca desde los modelos comunes más importantes hasta la API de acceso (controladores) a ellos.
    - “CameraController”: comprende el control controlador de cámara.
    - “GridCameraView”: con el control de rejilla de cámaras.
    - “IPCameraControl”: con el proyecto que permite conectarse a las cámaras locales y remotas de tipo IP.
    - “IPVideoSurveillance”: el proyecto principal.
    - “SkinnedControls”: es la biblioteca de controles que permiten “skin”.
    - Otros proyectos secundarios:

- “Test-SkinnedControls”: permite probar la biblioteca “SkinnedControls”.
- “UpdateCameraDatabase”: permite obtener la lista de cámaras y propiedades que el usuario podrá seleccionar al configurar cámara.
- “Installer”: el instalador de la aplicación.

5.3. Evaluación de costes

5.3.1. Costes económicos

Visual Studio es un entorno de desarrollo software en el que no se ha gastado dinero en licencias ni royalties por disponer del mismo a través de la Universidad vía “Microsoft DreamSpark”. Si este entorno resultara impropio para realizar negocios se pasaría a utilizar un editor y compilador gratuito, como “SharpDevelop” con “.MONO” o incluso una versión Express (si fuera compatible).

Otras librerías que se van a utilizar tienen licencias que conceden su uso incluso comercial de manera gratuita.

Tampoco se van a realizar compras de hardware alguno.

Ya que la inversión no ha consistido en dinero sino solo en tiempo de desarrollo realizado por mí en mi tiempo libre, sea cual sea una posible futura ganancia en el pago de añadidos solicitados o soporte al cliente serán ganancias completas.

6.1.1. Redacción de actividades

Utilizando la herramienta Microsoft Project he planificado una propuesta de actividades que sigue el plan de trabajo que aquí indico. En la realización de las actividades incluimos:

Redacción del Plan de Trabajo

Para la investigación sobre el proyecto a elegir, las tecnologías a utilizar en un estudio técnico y la redacción del presente documento he empleados varios días, comenzando el día 18 de septiembre de 2014 y finalizando el 29 del mismo mes. En realidad la investigación empezó unos días antes pero oficialmente entiendo que debo contar desde la fecha en que se me pidió que hiciera el plan de trabajo. Entonces tendríamos:

- Estudio técnico.

- Planificación del proyecto.

6.1.2. Análisis y diseño UML

El lenguaje unificado de modelado (UML) me permitirá modelar el análisis que realice sobre el proyecto. Para ello dispondré del intervalo de días entre el día siguiente a la finalización del anterior hito hasta el 27 de octubre. El modelado cumple una función diseñadora y es por ello que no permanecerá totalmente estático ya que según vaya adentrándome en el desarrollo real puede llegar a modificarse en pos de mejoras o retoques:

- Comenzando por el análisis de las necesidades y requisitos.
- Diseñando diagramas de Casos de Uso con los anteriores datos.
- Elaborando un modelo de clases a partir de los elementos encontrados.
- Diseñando la interfaz de usuario.
- Diseñando Diagramas de Secuencia para poder seguir mejor el correcto orden de las funcionalidades posibles.

6.1.3. Implementación

Los siguientes serán los módulos a tratar en la implementación desde el 28 de octubre hasta el 15 de diciembre:

- Vigilancia de cámaras (monitorización normal). Estructura en general.
- Selección de cámaras.
- Uso de cámara.
- Selección de vista de cámaras.
- Configuración de cámaras.
- Opciones generales.
- Módulo cámara.
- Controles y módulos que permitan “skinning”.
- Diseño gráfico.
- Persistencia de datos.

Cada módulo requiere pruebas específicas pero no hay que olvidar las pruebas generales para comprobar que todo junto sigue correctamente:

- Pruebas generales.

Por último tendré que realizar algún tipo de manual para iniciar al usuario en el uso de la herramienta software:

- Documentación.

Documentación,

La documentación la realizaré entre los días 16 de diciembre y 12 de enero de 2015.

Consistirá en:

- Manual de instalación y configuración.
- Manual de usuario.
- Memoria.

Elaboración de la presentación virtual

Dado que la presentación virtual la realizaré entre los días 26 de enero de 2015 y el día 29 del mismo, dispondré de los días en el intervalo del 13 hasta el día 25 de enero de 2015 para elaborar la presentación virtual.

6.1.4. Intervalos de tiempos

Se indican aquí los días que se van a trabajar, en un número de horas que dependen de mi disponibilidad. Así, aunque todos los días al menos haré una hora, la mayor parte de los domingos podré dedicar 6 horas, los sábados 4 y los días de lunes a viernes entre 1 a 3 horas.

- Planificación inicial: 12 días.
    - Estudio técnico y viabilidad del proyecto: 4 días.
    - Planificación del proyecto: 8 días.
- Análisis y diseño UML: 28 días.
- Implementación: 49 días.

- Documentación: 28 días.

- Elaboración de la presentación virtual: 13 días.

TOTAL: 130 días.

6.1.5. Hitos a cumplir

Se corresponden con los intervalos entre el inicio y la fecha límite de entrega de las distintas partes del proyecto, además de la defensa de la presentación virtual a través del debate online:

- Planificación: Desde el 18/09/2014 hasta el 29/09/2014.

- Análisis: Desde el 30/09/2014 hasta el 27/10/2014.

- Implementación: Desde el 28/10/2014 hasta el 15/12/2014.

- Entrega final (documentación): Desde el 16/12/2014 hasta el 12/01/2015.

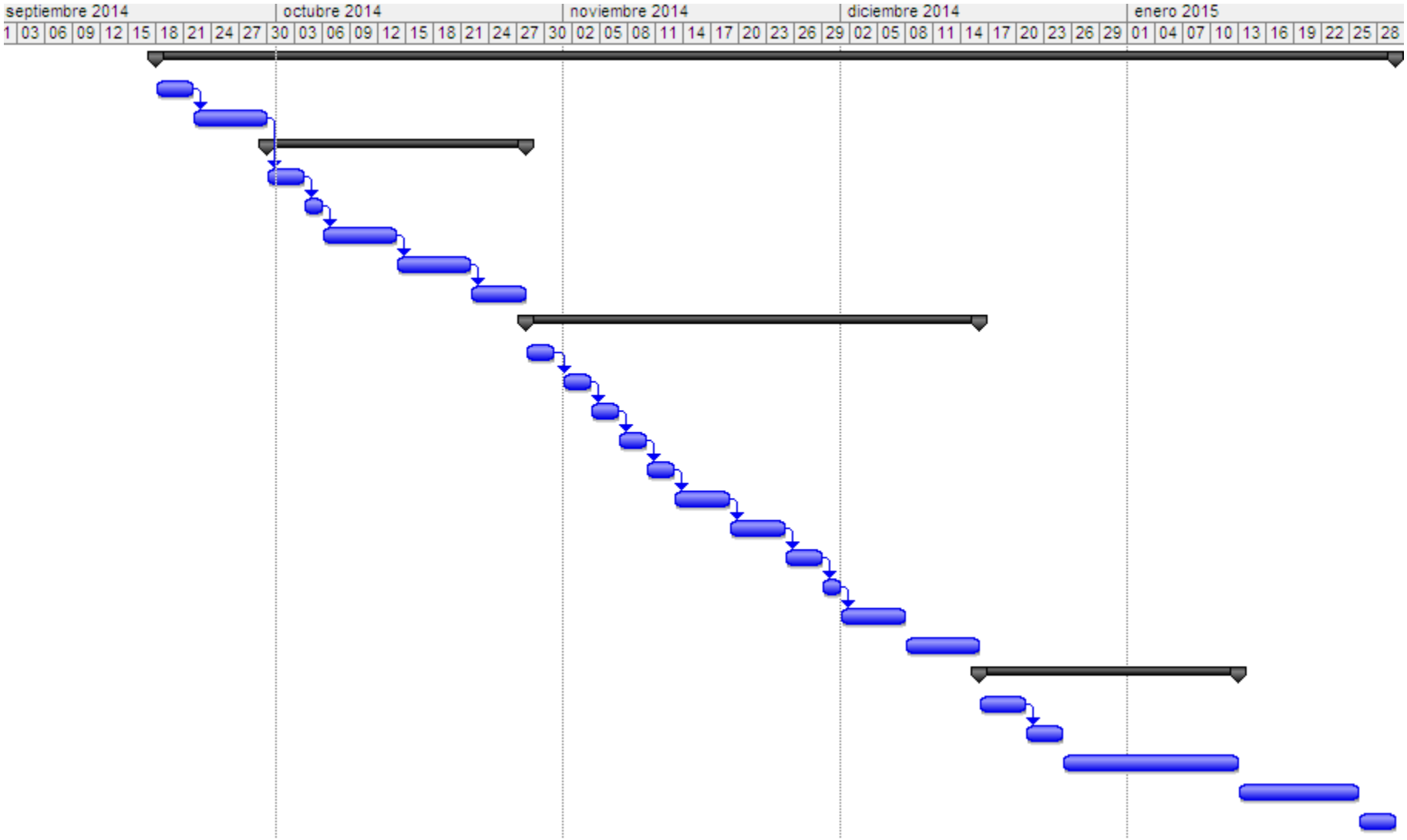
No son hitos del desarrollo del proyecto en sí mismo los siguientes puntos, los indico igualmente para mostrar la programación completa de la asignatura:

- Elaboración de la presentación virtual: Desde el 13/01/2015 hasta el 25/01/2015.

- Defensa del proyecto a través de debate: Desde el 26/01/2015 hasta el 29/01/2015.

6.1.6. Diagrama de Gantt

Nombre de tarea	Duración	Comienzo	Fin
Proyecto Sistema de Vídeo-Vigilancia IP	**134 días?**	**jue 18/09/14**	**jue 29/01/15**
Estudio técnico y viabilidad del proyecto	**4 días**	jue 18/09/14	dom 21/09/14
Planificación del proyecto	**8 días?**	lun 22/09/14	lun 29/09/14
Análisis y diseño UML	**28 días?**	**mar 30/09/14**	**lun 27/10/14**
Análisis de necesidades y requisitos	4 días	mar 30/09/14	vie 03/10/14
Casos de uso	2 días	sáb 04/10/14	dom 05/10/14
Modelo de clases	8 días?	lun 06/10/14	lun 13/10/14
Diseño interfaz gráfica	8 días?	mar 14/10/14	mar 21/10/14
Diagramas de secuencia	6 días	mié 22/10/14	lun 27/10/14
Implementación	**49 días**	**mar 28/10/14**	**lun 15/12/14**
Vigilancia de cámaras (monitorización nor	3 días	mar 28/10/14	jue 30/10/14
Selección de cámaras	3 días	sáb 01/11/14	lun 03/11/14
Uso de cámara	3 días	mar 04/11/14	jue 06/11/14
Selección de vista de cámaras	3 días	vie 07/11/14	dom 09/11/14
Configuración de cámaras	3 días	lun 10/11/14	mié 12/11/14
Opciones generales	6 días	jue 13/11/14	mar 18/11/14
Módulo cámara	6 días	mié 19/11/14	lun 24/11/14
Controles y módulos que permitan "skinnir	4 días	mar 25/11/14	vie 28/11/14
Diseño gráfico	2 días	sáb 29/11/14	dom 30/11/14
Persistencia de datos	7 días	lun 01/12/14	dom 07/12/14
Pruebas generales	8 días	lun 08/12/14	lun 15/12/14
Documentación	**28 días?**	**mar 16/12/14**	**lun 12/01/15**
Manual de instalación	5 días?	mar 16/12/14	sáb 20/12/14
Manual de usuario	4 días?	dom 21/12/14	mié 24/12/14
Memoria	19 días?	jue 25/12/14	lun 12/01/15
Elaboración presentación virtual	**13 días?**	mar 13/01/15	dom 25/01/15
Debate Tribunal	4 días?	lun 26/01/15	jue 29/01/15


septiembre 2014
octubre 2014
noviembre 2014
diciembre 2014
enero 2015
1 03 06 09 12 15 18 21 24 27 30 03 06 09 12 15 18 21 24 27 30 02 05 08 11 14 17 20 23 26 29 02 05 08 11 14 17 20 23 26 29 01 04 07 10 13 16 19 22 25 28

6. Mejoras futuras

- Añadir la capacidad de modificar el volumen por cámara a través de un control SkinnedControls.SkinnedSlider. El control por código está listo e incluso el diseño del controlador gráfico en Inkscape (tipo slider). También está lista la posibilidad de elegir audio en VLC (aunque oculta en runtime).

- Modo pantalla completa de la cámara seleccionada.

- Capacidad multi-pantalla. Además, un combo en el controlador de cámara para poder seleccionar en qué pantalla se verá en modo pantalla completa.

- ¿Existirá alguna manera de evitar la pixelación en los vídeos donde las imágenes tiene mucha calidad pero se deben de ver en pantallas muy pequeñas? Esto es algo innato a VLC pero se puede seguir buscando una solución.

- Incluir mi proyecto “UpdateCameraDatabase” en la aplicación principal y modificarlo para obtener mis datos del fichero actualizado XML que por lo visto provee la Web iSpyConnect. Esta URL se puede encontrar en el código fuente de un producto open source de la Web que se llama como la propia Web. Ahora mismo mi proyecto es muy lento parseando la Web y sería casi instantáneo si solo bajase el fichero XML de la Web.

- El fichero XML “UniqueCameraDatabase.xml”, obtenido a través del proyecto “UpdateCameraDatabase”, contiene una lista con todas las URLs de conexión posibles únicas. Esto nos permitiría añadir la funcionalidad de conexión automática de cámara IP remota (verificando una tras otra hasta encontrar alguna que funcione). En este caso, la configuración encontrada se guardaría como personalizada (ya que no contiene información alguna sobre el nombre de marca o producto –ya que éste puede ser múltiple-).

- Se podría permitir mayor nivel de personalización de colores y elementos para la aplicación, el control personalizado SkinnableComboBox e incluso para SkinnableControls.FrmMenuListForm (un formulario interno que se usa en los combos).

- Se podría intentar mejorar el soporte para imágenes JPEG en secuencia.

- Añadir sistema de guardado de vídeo (de hecho por código ya funciona excepto para imágenes JPEG en secuencia).

- Se podría añadir la captura de imagen estática (actualmente funciona por código excepto para imágenes JPEG en secuencia). He encontrado una manera de hacer incluso ese tipo de captura de capturas pero capturaría todo lo que hay sobre ella impreso (erróneo si no ha refrescado bien o se ha interpuesto otro control encima). El

proyecto-investigación del que hablo es “MouseEventsInTranspPic” (no incluido en este TFC).

- Cambiar el tipo de ubicación del “skin”, en mi proyecto “SkinnedControls”, de ubicación absoluta a relativa (propiedad “SkinPath”) para modo diseño (en “runtime” ya lo hace bien).

- Utilizar compatibilidad (alternativa con VLC) con:
  + AForge.NET (solo compatible con cámaras HTTP y cámaras locales).
  + FFMPEG.
  + ONVIF (para cámaras compatibles con ONVIF).
  + OpenCV (cvCaptureFromFile). No tan ideal debido a su considerable espacio utilizado (además habría que incluir el wrapper EMgU).

  Estos tipos distintos de tratar las imágenes los llamaría “motores de renderizado”.

- Se podrían sacar ideas del proyecto open source iSpyConnect. Por ejemplo, éste captura las imágenes–si se requiere detección de movimiento- o las modifica –si se necesita realizar un efecto sobre las imágenes-; y luego las dibuja sobre un entorno controlado como un panel (esto evitaría –por ejemplo- los problemas que he tenido con la ventana DirectX que a veces emerge como popup en ventana independiente).

- He detectado que en pantallas muy pequeñas necesitaría añadir unas barras de desplazamiento (skinnables) al sumario.

- Se podría añadir detección de movimiento. Por ahora no lo consigue porque las propiedades LogProperties.Verbosity y LogProperties.LogMessages no funcionan en VLC. Esto lo podría arreglar modificando el código original de VLC o, mejor aún, modificar el filtro de movimiento que trae VLC, "motion.c", para que me escriba en ficheros determinados la detección de movimiento para poder capturarlo.

- El configurador de cámaras podría ir directamente a configurar la cámara seleccionada (actualmente es la última por modificar, o la primera si en la sesión no se ha accedido aún a ninguna).

- En el configurador de cámaras podría añadirse la validación automática (en el método “storeConfig”) de la IP/Host. Incluso un botón para que el usuario pueda probar manualmente también.

- Añadir soporte para francés (ahora solo tengo soporte para castellano e inglés).

7. Reflexiones y conclusiones

7.1. Reflexiones iniciales

Tras varias pruebas realizadas en mi domicilio que me sirvieron para seleccionar éste proyecto por delante de otros, veo posible el desarrollo del producto. Quedaron patentes en una anterior sección, los riesgos a los que me enfrento dedicando buena parte de este desarrollo a hardware y trabajos gráficos singulares, pero gracias a la selección cuidadosa que realicé de las librerías que utilizaré creo que me aseguro un buen resultado, al menos una pequeña prueba de concepto del producto parece ir bien con las cámaras que tengo en casa.

Este proyecto ya me ha permitido aprender más de lo que sabía sobre tecnologías de transmisión de vídeo por red y cableadas además del tratamiento de imágenes (para el “skinning”), y estoy seguro que al final del mismo sabré mucho más. Porque cada proyecto es una experiencia nueva, siempre se aprende algo y no es poco, y mucho más de los errores que de tener la costumbre de ganar, porque el que no arriesga no aprende.

Existe mucho desarrollo a realizar, es bastante trabajo y, debido a que estamos tratando con hardware y elementos poco habituales en la informática de gestión que más conocemos, posee varios frentes a investigar basándome en el análisis inicial:

- Sistema que permita controles y “skins”. He estado investigando la mejor manera y tengo una propuesta alfa más o menos operacional, lo he realizado como proyecto independiente por si más adelante quisiera incluirlo en otro proyecto. Aún así, son controles complejos y densos a realizar en su totalidad así que he decidido que limitaré su funcionamiento a justamente lo que necesite en este proyecto, no mucho más. La evolución de esta librería de controles verá la luz en futuros proyectos donde quiera volver a usarlos pero por ahora tendrán funcionalidades mínimas.

- El sistema de captura de imágenes estará encapsulado en un control que lo gestione todo de manera transparente. Necesita algo más de investigación para poder realizarse completamente. Y es que había pensado utilizar VLCDotNet y dedicarme a hacerlo compatible con unas pocas marcas y modelos de cámaras IP pero he encontrado una bibliotecas interesante, la ONVIF, que haría mi trabajo mucho más sencillo, aunque sin embargo no acaba de querer funcionar por ahora con algunas cámaras online que he ido probando, así que toca seguir investigando o utilizar como había planeado VLCDotNet.

7.2. Conclusiones finales

Sí que he podido aprender mucho más sobre comunicaciones y tratamiento de la imagen, vídeo y sonido entre otras muchas cosas. Tras finalizarlo, veo de manera mucho más concreta los problemas, incidencias y soluciones aplicadas al proyecto y cómo encarar futuras mantenimientos y mejoras.

7.2.1. Cambios en el diseño o planificación e incidencias

- Tuve que añadir un proyecto nuevo que llamé “UpdateCameraDatabase” que se alimenta –a través de parsing HTML- de una sección paginada de marcas, modelos y URLs en la página Web iSpyConnect.com. Mi proyecto permite obtener toda una base de datos de cámaras IP y, a diferencia de la Web original, con nombres únicos generando un fichero XML llamado “CameraDatabase.xml” (y otro para futuros usos llamado “UniqueCameraDatabase.xml”).

- La lista de cámaras conectadas localmente no las consigue a través de API Windows ni VLC ya que ninguno es seguro ni correcto en todos los casos que he probado. Así que uso la librería AForge.NET.

- Tanto el control VLC que viene en la librería LibVLC como el wrapper para .NET que usé, VLCDotNET, tiene muchísimos errores y métodos que no funcionan. La lista de problemas que he tenido que ir arreglando es enorme, a veces las soluciones han sido realmente parches tan feos como los errores en sí mismos. Ejemplos:

    - El uso de los métodos de VLC “dispose” o “stop” pueden provocar el cuelgue de la aplicación, pero uno imposible de atrapar mediante “try-catch”. Así que mejor que usar alguno de éstos métodos prefiero eliminar el control de otras maneras e incluso me aseguro de su eliminación mediante un hilo paralelo que acaba eliminándolo en caso de que prosiga de manera inesperada.

    - El método VLC “pause” simplemente no siempre pausa (ver imágenes JPEG en secuencia).

    - La ventana DirectX de VLC a veces sale del marco del control, flotando de manera independiente sin control. He tenido que usar API de Windows para poder atajar el problema, además de un uso más correcto del cambio de tamaño de los controles –que puede llegar a afectarle-.

- Encontrados problemas con el control de audio simultáneo para varios controles VLC. Llegué a solucionarlo al menos en el control de cámara. Aquí un ejemplo de otros que encontraron el mismo problema que yo:
  http://vlcdotnet.codeplex.com/discussions/554111

7.2.2. Requisitos no conseguidos

Como ya he indicado, fueron varias las incidencias y problemas que retrasaron demasiado el proyecto, impidiendo la realización de ciertos detalles que tuve que abandonar a cambio de otros más importantes. Los siguientes fueron los puntos que no se han realizado:

- Capacidad de configurar el volumen por cámara y el general.

- Capacidad multi-pantalla (incluyendo modo pantalla completa de la cámara seleccionada).

8. Bibliografía

Wikipedia	Definición de la enciclopedia online “Wikipedia”.	http://es.wikipedia.org/wiki/Wikipedia
Wikipedia	Definición de CCTV (como circuito cerrado de televisión) para cámaras IP o tradicionales (analógicas/digitales).	http://es.wikipedia.org/wiki/Circuito_cerrado_de_televisi%C3%B3n
Wikipedia	Definición de Cámara IP.	http://es.wikipedia.org/wiki/C%C3%A1mara_IP
Wikipedia	Definición de Vídeo-vigilancia IP.	http://es.wikipedia.org/wiki/V%C3%ADdeovigilancia_IP
Wikipedia	Presentación y un poco de historia sobre “Microsoft”.	http://es.wikipedia.org/wiki/Microsoft
MSDN Microsoft	Historia e introducción de C# y .NET en la Web de Microsoft, su creador.	http://msdn.microsoft.com/es-es/library/a72418yk.aspx
Wikipedia	C# según Wikipedia.	http://es.wikipedia.org/wiki/C_Sharp
Wikipedia	Librerías Microsoft .NET según Wikipedia.	http://es.wikipedia.org/wiki/Microsoft_.NET
Wikipedia	Sobre lo económico y sencillo de la vídeo-vigilancia IP.	http://es.wikipedia.org/wiki/V%C3%ADdeovigilancia_IP
Wikipedia	El “Internet de las cosas”.	http://es.wikipedia.org/wiki/Internet_de_las_cosas
Asociación española de empresas de seguridad	Sobre las cámaras analógicas como la mayoría del parque instalado.	http://www.aesseguridad.es/boletin/36/boletin_aes_36.pdf
Wikipedia	Sobre “DIY” (“Do it yourself”)	http://es.wikipedia.org/wiki/H%C3%A1galo_usted_mismo
Google Android Store	Software para Android “DroidCam Wireless Webcam”.	https://play.google.com/store/apps/details?id=com.dev47apps.droidcam&hl=es_ES
Wikipedia	Objetivo o comando “Zoom”.	http://es.wikipedia.org/wiki/Zoom
Wikipedia	“Zoom” y “pan”.	http://es.wikipedia.org/wiki/Movimientos_de

		_c%C3%A1mara
Wikipedia	Librerías gráficas DirectX.	http://es.wikipedia.org/wiki/DirectX
Microsoft	Descarga desde el sitio de Microsoft.	http://www.microsoft.com/es-es/download/details.aspx?id=35
Wikipedia	Sobre el “streaming”.	http://es.wikipedia.org/wiki/Streaming
GitHub	Web de la MetroFrameWork.	https://github.com/viperneo/winforms-modernui
VlcDotNet (CodePlex)	Web de la VideoLan DotNet.	http://vlcdotnet.codeplex.com/
VideoLan	Web del software VideoLan (VLC).	http://www.videolan.org/
Wikipedia	MRL (media resource locator).	http://en.wikipedia.org/wiki/Media_resource_locator
Google Trends	Tendencias sobre búsquedas en Google.	http://www.google.com/trends/explore#q=camera%20surveillance%2C%20%2Fm%2F0lz3x&date=1%2F2011%2045m&cmpt=q
Wikipedia	“Skin” en software.	http://es.wikipedia.org/wiki/Skin_%28software%29
Wikipedia	Definición de UML.	http://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado
Wikipedia	Código abierto (“open source”).	http://es.wikipedia.org/wiki/C%C3%B3digo_abierto
Wikipedia **Microsoft website**	Microsoft DreamSpark.	http://en.wikipedia.org/wiki/DreamSpark https://www.dreamspark.com/
Mono project website	Equivalente gratuito de .NET llamado .MONO	http://www.mono-project.com/
Web SharpDevelop	Entorno de desarrollo SharpDevelop	http://www.icsharpcode.net/
Wikipedia	Definiciones importantes sobre los protocolos de comunicación más importantes.	http://es.wikipedia.org/wiki/Protocolo_de_comunicaciones http://es.wikipedia.org/wiki/Real_Time_Streaming_Protocol http://en.wikipedia.org/wiki/Windows_Media

		_Services
Wikipedia	Definición de “mockup” y ejemplo de uso (“Balsamiq mockups”).	http://es.wikipedia.org/wiki/Mockup
Web de “Balsamiq mockups”		http://balsamiq.com/products/mockups/

9. Vocabulario técnico básico

CCTV: Circuito cerrado (privado) de televisión. Se usa este término para designar a la vídeo-vigilancia.

Cámaras: Todos sabemos que son dispositivos capaces de capturar y a veces incluso grabar imágenes con o sin sonido. Las tradicionales utilizadas en CCTV eran analógicas cableadas, en su evolución pasaron a ser inalámbricas y actualmente conviven con otras digitales. Hoy en día las digitales que son capaces de difundir lo que capturan a través de redes informáticas de tipo IP son llamadas cámaras IP.

Visual C#: O simplemente C#, es un lenguaje de programación desarrollado por la empresa Microsoft

“**Zoom**”: Ampliación o reducción de una zona de imagen, en nuestro caso es una opción del programa que nos permite realizar dicho “zoom” con el vídeo que recogen las cámaras.

“**Pan**”: Se refiere a la panorámica de una imagen y, en nuestro caso, se trata de una opción del programa que nos permite ver una porción diferente de una imagen (que proviene de una cámara) sin cambiar de ampliación, simplemente desplazando la imagen (hacia arriba, abajo, derecha, izquierda, etc.).

“**Streaming**”: Difusión de vídeo y/o sonido en red. Actualmente Youtube.com es una de las Webs más utilizadas que utiliza esta idea como negocio principal.

“**MRL**”: Es el localizador de recursos multimedia (“Media Resource Locator”), o lo que es lo mismo, lo que la URL en internet pero para multimedia.

“**Skin**”: En entornos software, se refiere al tema gráfico que se aplica al diseño del mismo. Muchas veces se refiere a un cambio simple de colores pero otros van mucho más allá inspirándose en películas, motivos y todo tipo de ideas que provoquen inspiración.

“**Skype**”: El software más prestigioso y utilizado actualmente para realizar conferencias.

“**Open source**”: Es el término anglosajón que define al código fuente “abierto”, o sea, disponible para que otros puedan modificarlo.

“Ahorro energético”: En informática y concretamente en entornos domésticos, se refiere a los distintos medios o planes que ofrece un Sistema Operativo a sus usuarios para permitir un uso menor de los recursos hardware. Esto puede incluir: bajar la potencia o el uso de los procesadores (o sus núcleos) cuando no se requiera tanta potencia, apagar la pantalla conectada al ordenador cuando no se esté usando el mismo interactivamente (en monitores antiguos tipo CRT era vital para prolongar su vida útil), apagar los discos duros si no se usan, etc.