Gestión de cámaras de seguridad georreferenciadas en el campus de la ULPGC

Proyecto Fin de Carrera

Autor: Cristian Ramírez Santana

Tutor: Agustín Trujillo Pino

Titulación: Ingeniería en Informática

1. Agradecimientos

Deseo, en primer lugar, darle las gracias a mi familia por todo su apoyo y paciencia que han ofrecido siempre, tanto en momentos sencillos, como en momentos más duros.

En segundo lugar, agradecer a mi tutor Agustín Trujillo Pino por otorgarme la oportunidad de realizar el PFC con él. A pesar de estar la titulación desde hace años en extinción, le agradezco también su tiempo, paciencia y motivación a la hora de prestarme su ayuda en la ejecución del proyecto.

En tercer lugar, agradecer a Francisco Pérez Rosales, quien ostenta el cargo de Director de Seguridad en la ULPGC, por habernos prestado tanto el material al comienzo del proyecto, como garantizarnos el acceso a un servidor videograbador y a cámaras.

En cuarto lugar, quisiera agradecer a Antonio José su orientación y ayuda prestada sobre Capaware, la cual ha sido fundamental para el desarrollo del proyecto.

Por último, agradecer tanto a compañeros de la facultad, algunos de los cuales los considero grandes amigos, así como a amigos externos al mundo universitario, por su apoyo y motivación todos estos años.

2. Introducción

La industria de la videovigilancia representa uno de los sectores económicos más potentes del mercado en la actualidad. Países como China han llevado a cabo grandes inversiones en sistemas de videovigilancia para controlar a su propia población, combinando este sistema además con otros de reconocimiento facial [1].

Hoy en día, es imposible concebir un negocio, independientemente de su tamaño, que no disponga de un circuito cerrado de cámaras o todo un sistema de gestión de videovigilancia (en el caso de grandes superficies con disposición de personal de seguridad).

La mayoría de sistemas de gestión de videovigilancia constan, como mínimo, de elementos comunes tales como: árbol de jerarquía de cámaras, streamming de vídeo de alguna cámara previamente seleccionada, información sobre el estado de las conexiones a servidores o sobre el estado de las cámaras, entre otros.

Sin embargo, estos sistemas muestran ciertas carencias cuando se trata de gestionar un considerable número de cámaras, o bien, se tiene un determinado número de cámaras con una distribución muy espaciada. La razón de ello, es debida a que el personal de seguridad debe conocer la localización de una cámara por la descripción de la misma en el árbol de jerarquía, o al acceder a información de ella, es decir, debe memorizar dicha distribución de cámaras en última instancia.

Actualmente, la universidad de las Palmas de Gran Canaria cuenta con un sistema propio de videovigilancia basado en un circuito cerrado de televisión o CCTV en sus siglas comúnmente extendidas. El CCTV permite el acceso a servidores videograbadores que a su vez permiten el acceso a cámaras de seguridad de todo tipo (compatibles con controles PTZ, visión noturna…). Las cámaras están distribuidas por los diferentes edificios pertenecientes a la universidad, incluidas aquellas zonas fuera del campus de la ULPGC. Las cámaras suelen controlar los accesos a los edificios y las distintas plantas, además de zonas sensibles. Por motivos de seguridad no es conveniente detallar la distribución exacta de dichas cámaras. La gestión de este número considerable de cámaras requiere una férrea tarea de etiquetado de cámaras y zonas para localizar cámaras concretas.

El servicio de seguridad del campus de Tafira cuenta además con otros sistemas que apoyan las labores de videovigilancia. El control de acceso de los usuarios a los diferentes edificios del campus, además del sistema de control de incendios son parte de las labores de control del personal de seguridad. Todos estos sistemas son independientes entre ellos, por lo que un usuario nuevo, debe emprender una labor de aprendizaje algo costosa.

La idea del proyecto, trata de averiguar si el realizar la gestión de cámaras de videovigilancia en un framework geográfico 3D puede llegar a mejorar dichas tareas de gestión, teniendo la capacidad de visualizar las cámaras según su posición geográfica, además de disponer del susodicho árbol de jerarquía de cámaras.

Capaware [2] es un framework de desarrollo de entornos 3D geográficos multicapa. Fue una iniciativa del Gobierno de Canarias para su uso como visor 3D propio en temas de gestión de emergencias. Capaware permite la interacción con terrenos virtuales 3D con precisión cartográfica, distribuida bajo una licencia GNU GPL [3]. Permite conectarse a servidores externos con protocolos OGC para obtener datos para su uso propio.

Está desarrollado en lenguaje de programación C++ y utiliza como motor gráfico OSG (OpenSceneGraph). Posee además una arquitectura de plugins que le permite crecer en funcionalidades adicionales. Integra además otros componentes de software libre para su ejecución: wxWidgets, Curl y Boost.

3. Objetivos

El objetivo fundamental del proyecto es desarrollar sobre el motor geográfico Capaware, un plug-in que permita realizar tareas de gestión de videovigilancia como, por ejemplo, visualizar la imagen de una cámara concreta según su ubicación en el terreno.

Las tareas necesarias para concluir el proyecto son las siguientes:

- Generar un complemento en el motor geográfico Capaware de forma que el usuario pueda elegir una cámara a visualizar según su posición en el terreno o en la propia interfaz.
- Acceder de forma remota al contenido de Vídeo de las cámaras utilizando la API del videograbador.
- Visualizar el resultado dentro de Capaware.

4. Estado del arte

En esta sección se analizará el sistema actual de videovigilancia empleado por el personal de seguridad en el campus de la ULPGC. Se describirán los diferentes aspectos de dicho sistema, desde los equipos servidores utilizados, hasta la aplicación que usa el personal para realizar las tareas de videovigilancia.

4.1 Sistema de videovigilancia empleado

Actualmente en el campus de la ULPGC se dispone de una red de cámaras IP conectadas a servidores de grabadores de video digital (DVR en sus siglas en inglés). Se cuenta con varias versiones de dichos equipos de la serie DX de Pelco [4], debido a la compra discontinuada de ellos a lo largo del tiempo. Se analizará en concreto la versión dx8100, de la cual dispone el alumno una dirección IP conocida de un servidor concreto, además de acceso a dos cámaras exteriores.

4.1.1 Grabador de video digital DX8100

Un grabador de video digital o DVR es un dispositivo de grabación de vídeo en formato digital. Un DVR se compone, por una parte, del hardware, que consiste en un disco duro de gran capacidad, un microprocesador y los buses de comunicación; por otro lado, del software, que proporciona diversas funcionalidades para el tratamiento de las secuencias de video recibidas, acceso a guías de programación y búsqueda avanzada de componentes.

Se podría calificar a un DVR como una computadora especializada en el tratamiento de imágenes digitales. Así como su predecesora la videograbadora (HVR en sus siglas en inglés), no solo es capaz de realizar las tareas de almacenar de forma pasiva las imágenes de una o varias cámaras IP, la capacidad de rebobinar o pausar; sino que además proporciona la capacidad de almacenar información fundamental como fechas, horas o estadísticas de transacción.

Los dx8100 son una versión bastante antigua, como podrá verse más adelante en el documento en la resolución de las imágenes de video. Es distribuida por Pelco, una gran empresa del sector para uso doméstico o arquitecturas de tamaño medio (Imagen 1).

Ilustración 1: Servidor de videograbación DX8100

4.1.2 Carácterísticas de grabadora DX8100

Las principales características del sistema DX8100 son las siguientes:

- Capacidad entre 8 y 32 entradas analógicas para cámaras.
- Capacidad hasta 480/400 IPs de NTSC/PAL.
- Exime de la necesidad de pagos de licencias siempre que se usen cámaras Pelco IP, Sarix IXS0 y Axis Standard Definition IP.
- Monitorización del estado del sistema.
- Conectividad remota desde el cliente a un máximo de 200 servidores.
- Capacidad hasta un máximo de 8 TBs.
- Notificación por email en caso de alarma.

4.1.3 Aplicación de gestión DX8100

Con la adquisición de estos equipos, se ofrece un cliente estándar en CD/DVD. Dicho cliente es una aplicación de escritorio en Windows y ofrece las características típicas de un sistema de videovigilancia. En la imagen 2, se muestra la ventana de la aplicación.

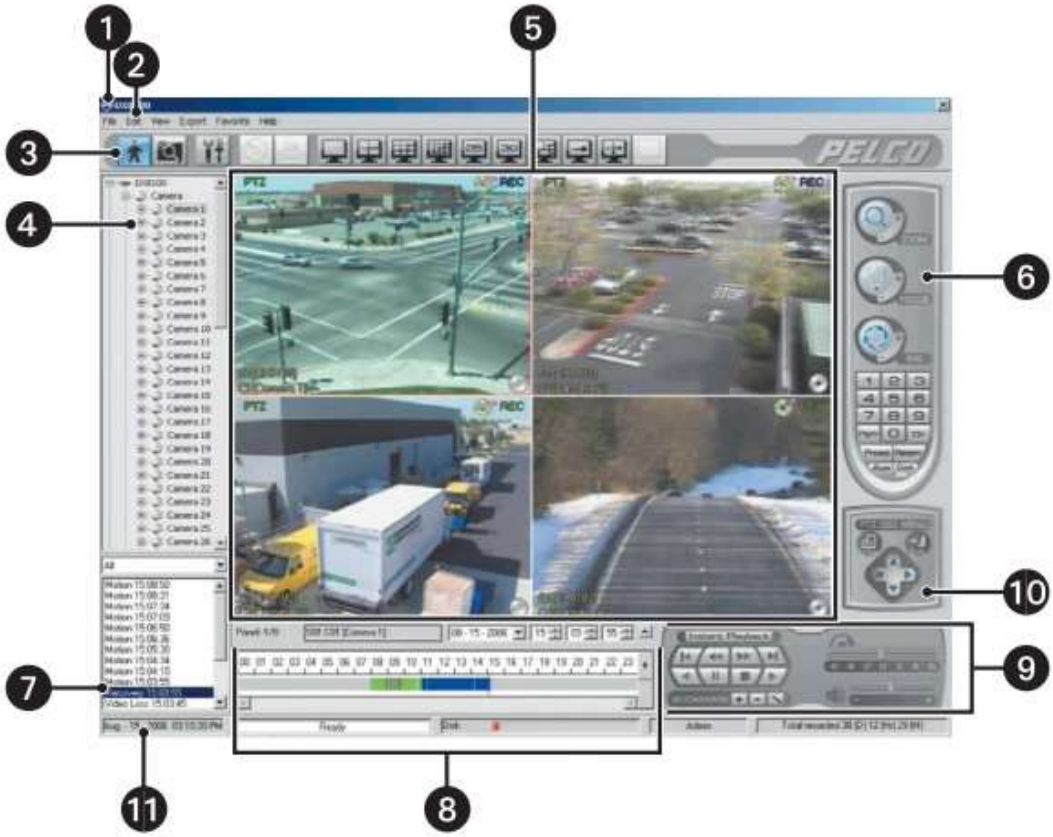

Ilustración 2: Ventana del cliente de grabadora DX8100

A continuación, se describen las diferentes secciones del cliente:

1. Barra de título.
2. Barra de menú.
3. Barra de herramientas: en ella, se dispone de acciones como mostrar, configurar, exportar y buscar.
4. Árbol de cámaras: sección clave de todo sistema de videovigilancia. En ella se construye una jerarquía de cámaras a criterio del administrador o de la distribución de la red de cámaras. Siempre debe construirse de la manera más intuitiva posible.
5. Sección de visualización.
6. Panel de control PTZ: a través de este panel, se puede enviar los distintos comandos PTZ a una cámara compatible con esa tecnología.
7. Lista de eventos: contiene una lista de posibles alarmas, o eventos programados por los propios usuarios. Requiere de permisos adicionales para verse.
8. Barra de tiempo: muestra una barra de tiempo de un periodo de 24 horas y proporciona a los usuarios básicos la capacidad de acceder en histórico en dicho periodo de tiempo.
9. Panel de histórico: proporciona a los usuarios con los permisos adecuados, el acceso en histórico a un momento del tiempo concreto. Ofrece la posibilidad de mostrar video en "playback".
10. Panel de Zoom digital.
11. Barra de estado del sistema.

4.2 Sistema de videovigilancia actual

El mercado ofrece una serie de posibilidades que no se disponían anteriormente. A la hora de seleccionar un DVR específico, debe tenerse en cuenta una serie de aspectos previos con respecto a las cámaras que se deseen utilizar:

- Cobertura en la vigilancia: dependiendo de cuanto ángulo se desea controlar, será necesario adquirir mayor o menor cantidad de cámaras. Este factor es importante, ya que un gran número de cámaras, implica mayor cantidad de servidores y, a su vez, mayor coste.
- Calidad de resolución de cámaras: una resolución de 1080p supone una calidad decente en la imagen de vídeo, pero a su vez será necesario almacenar mayor cantidad de información en los discos duros de los servidores.
- Capacidad de almacenamiento de los DVR.
- Acceso desde dispositivos móviles: aspecto esencial para uso doméstico de vivienda, pero puede ser útil en algún campo de investigación si se trata de universidades o centros de investigación.
- Cámaras con tecnología Wireless.
- Cámaras con tecnología PTZ.
- Cámaras con tecnología de ahorro de energía.

4.2.1 Sistema de gestión de video VideoXpert de Pelco

VideoXpert [5] es un sistema de gestión de video que abarca las necesidades desde un uso doméstico, como puede ser la vivienda de un particular o un pequeño negocio, hasta las

necesidades de centros educativos, almacenes, sedes de gobierno, entre otros. Presenta una serie de características interesantes como son:

- Reproducción síncrona.

  Permite sincronizar el vídeo de manera que añade dinámicamente nuevas cámaras a grupos que ya están sincronizados.

- Gestión de incidencias.

  El gestor de incidencias permite un modo de investigación ágil que permite localizar y organizar rápidamente el vídeo y crear listas de reproducción. El sistema permite observar la escena desde varios ángulos posibles (siempre que sea posible), además de la exportación y almacenamiento de dichas listas.

- Etiquetado.

  Las etiquetas son una herramienta útil que permite al sistema clasificar de manera óptima las diferentes cámaras que están operando. Se permite la búsqueda de las mismas ya sea por nombre, o por la asignación de una ID numérica.

- Mapping interactivo embebido.

  El sistema de gestión permite: importación de archivos de CAD, exportación de archivos de CAD con una nueva capa de cámaras y mapas múltiples para la representación de áreas críticas.

- Plugins de terceros.
- Configuración de seis monitores.

4.3 Motor geográfico 3D Capaware

Capaware es una plataforma, basada en software libre, para el desarrollo de aplicaciones geográficas 3D. Cuenta con una licencia GPL, que permite la visualización realista y en 3D de terrenos, compatible con capas Open Geospatial Consortium (OGC) [6]. El proyecto se origina a partir de una iniciativa del Instituto Tecnológico de Canarias (ITC) [7] en colaboración con la Universidad de las Palmas de Gran Canaria.

Capaware proporciona una interfaz fácil de usar y flexible que simplifica el desarrollo de nuevas aplicaciones, permitiendo crear entornos virtuales con múltiples capas de información sobre el terreno.

Con las capacidades clásicas de un Sistema de Información Geográfica (SIG), capaware permite la carga de capas WMS sobre entornos 3D, añadir elementos 3D sobre el terreno y visualizar elementos dinámicos, ofreciendo una perspectiva de la información analizada.

Ha sido desarrollado en el lenguaje de programación C++, utilizando el motor gráfico de código abierto Open Scene Graph [8], y la librería de interfaz libre WxWidgets [9]. Con un sistema de plugins, permite que cualquier desarrollador pueda complementar el sistema con nuevas funcionalidades específicas.

El uso de lenguajes primarios en la plataforma permite a hardware de gama medio-bajo su ejecución y evita la dependencia de equipos caros. Gracias a la oportunidad de crear un terreno 3D propio, se ofrece la posibilidad de trabajo offline (Imagen 3).

Ilustración 3: Ventana de Capaware

4.4 Sistemas de Información Geográfica actuales

Un sistema de información geográfica (SIG o GIS en sus siglas en inglés) es un conjunto de herramientas que integra y relaciona diversos componentes que permiten la organización, almacenamiento, manipulación, análisis y modelización de grandes cantidades de datos procedentes del mundo real que están vinculados a una referencia espacial, facilitando la incorporación de aspectos socio-culturales, económicos y ambientales que conducen a la toma de decisiones de una manera más eficaz.

Es cualquier sistema de información capaz de integrar, almacenar, analizar, editar, compartir y mostrar la información geográficamente referenciada. En resumen, los SIG son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones.

Una de las representaciones comunes de estas SIG, es la creación por parte de empresas u otros de un software que represente a la Tierra o a un planeta. Estos softwares son conocidos normalmente con el nombre de Globos Virtuales. Los globos virtuales permiten a los usuarios moverse libremente alrededor del globo, cambiando su posición y ángulo de vista. Los principales globos virtuales se detallan a continuación. En el trabajo de tesis doctoral de Santana [10] puede verse una relación más amplia y detallada sobre globos virtuales.

4.4.1 Google Earth

Google Earth [11] es el globo virtual más conocido en el mundo, el cuál ha sido desarrollado por Google [12]. El software comenzó su desarrollo bajo otro nombre en 2004, por la compañía Keyhole Inc. Más tarde fue adquirido por Google y pasó a llamarse oficialmente Google Earth. El software fue desde el principio código cerrado y dirigido a todo tipo de usuarios.

Existen varias versiones de la aplicación que van desde una licencia gratuita, hasta diferentes versiones de pago según la necesidad del usuario final. Cada versión ofrece diferentes funcionalidades y conjuntos de datos adicionales.

Algunas de las principales características de Google Earth son:

- Facilidad de uso.
- Amplia variedad de herramientas para navegación, búsqueda, medida, etc.
- Diferentes opciones de visualización como son la niebla, luz solar, efectos atmosféricos o reflejo de luz en superficies.
- La inclusión de grandes conjuntos de datos proporcionados por Google como pueden ser modelos de elevación, modelos de ciudades, edificios 3D, entre otros.

Al no ser un código libre per se, presenta inconvenientes a la hora de integrar conjuntos de datos personalizados por usuarios o de crear aplicaciones third-party.

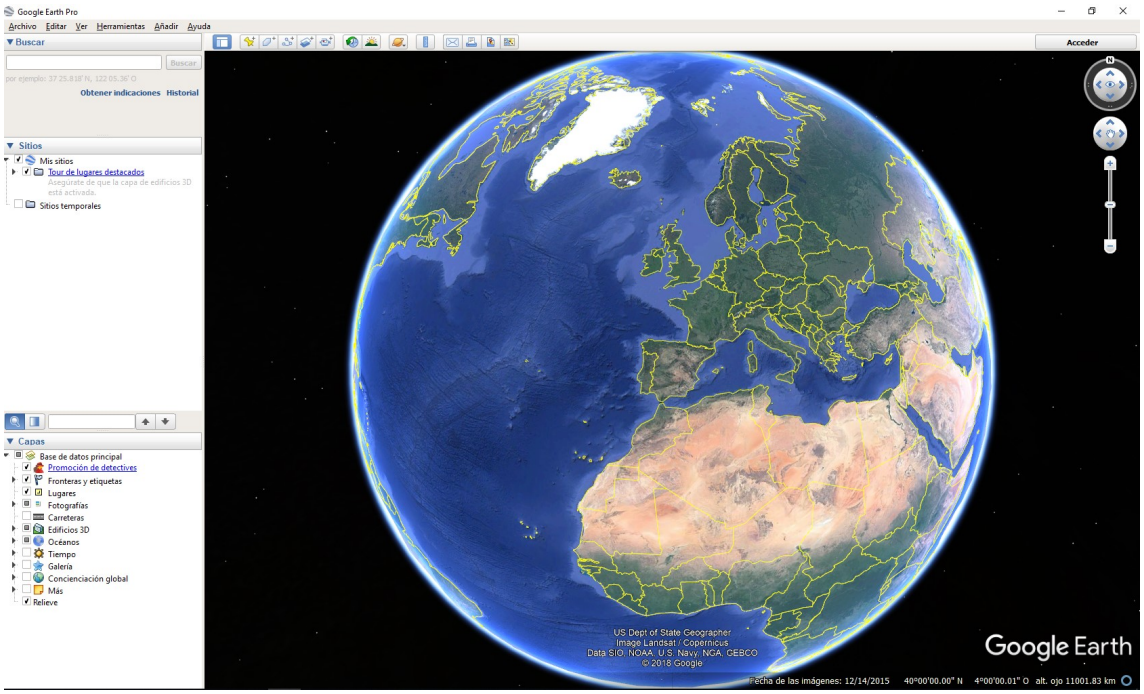

Ilustración 4: Aplicación Google Earth Pro en Windows 10

4.4.2 World Wind

World Wind [13] es un globo virtual, presentado en 2004 por la NASA [14], con el objetivo de ofrecer uno de los primeros softwares de código libre para la muestra de datos georreferenciados en un entorno 3D. Es todavía un referente en el campo de la investigación y de los SIG. Está disponible para escritorio, móvil y web (Imagen 5).

Superpone imágenes de satélite de la NASA fotografías aéreas del United States Geological Survey (USGS) [15] sobre modelos tridimensionales de la tierra. El usuario puede interactuar con el planeta seleccionando, rotando y ampliando zonas. Además, se pueden superponer topónimos y fronteras, entre otros datos a la imagen. Existen múltiples ampliaciones de World Wind que expanden su funcionalidad, por ejemplo, la capacidad de medir distancias u obtener datos de posición desde un GPS.

World Wind se presenta como un motor de globo virtual accesible a través de una API. Esto implica que el motor obliga a los usuarios a programar los elementos de la escena, permitiendo una amplia personalización. Algunas de sus características principales son:

- Integración sin restricciones de aplicaciones third-party.
- Multiplataforma (Windows, Linux, Android y Web).
- Repositorio de imágenes de la NASA.
- Herramientas de anotación.
- Interfaces estándar de bases de datos y servicios SIG.

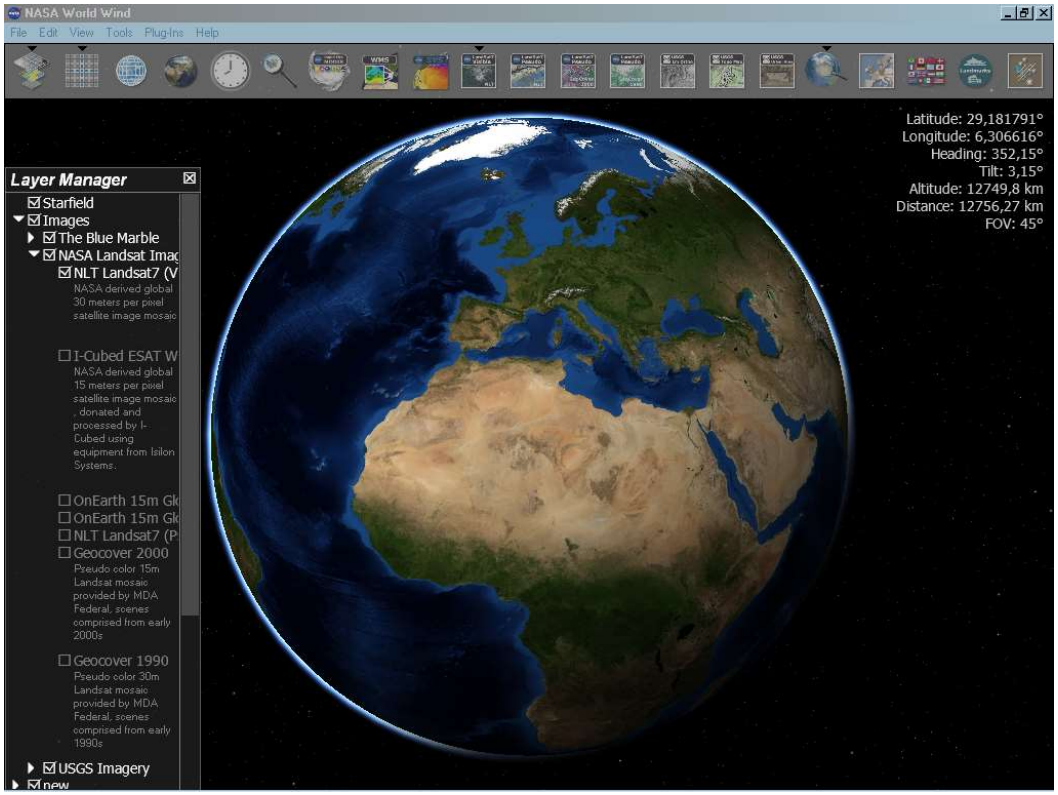

Ilustración 5: Ventana de World Wind. Fuente: Wikipedia.org

4.4.3 Cesium

Lanzado en 2012 como un proyecto de código libre, Cesium [16] se ha convertido en los últimos tiempos uno de los globos virtuales más usados. Su éxito viene dado por ser lanzado bajo una licencia de Apache 2.0, la cual otorga a cualquier aplicación un uso gratuito de la misma (Imagen 6).

Cesium está desarrollado como una librería Javascript para la generación de mapas 2D y 3D, únicamente usando tecnologías web. De hecho, los gráficos de Cesium están implementados en WebGL.

La composición de la escena se realiza mediante la API del globo, la cual ofrece al usuario una gran capacidad en la personalización y los resultados finales. Algunas de las principales características de Cesium son:

- Diferentes formas y tamaños para el planeta.
- Lenguaje propio para la definición de escenas dinámicas (CZML en sus siglas en inglés).
- Compatibilidad con muchos formatos de modelos 3D como Collada.
- Multitud de efectos de escena o iluminación.
- Diferentes esquemas de control de cámara.

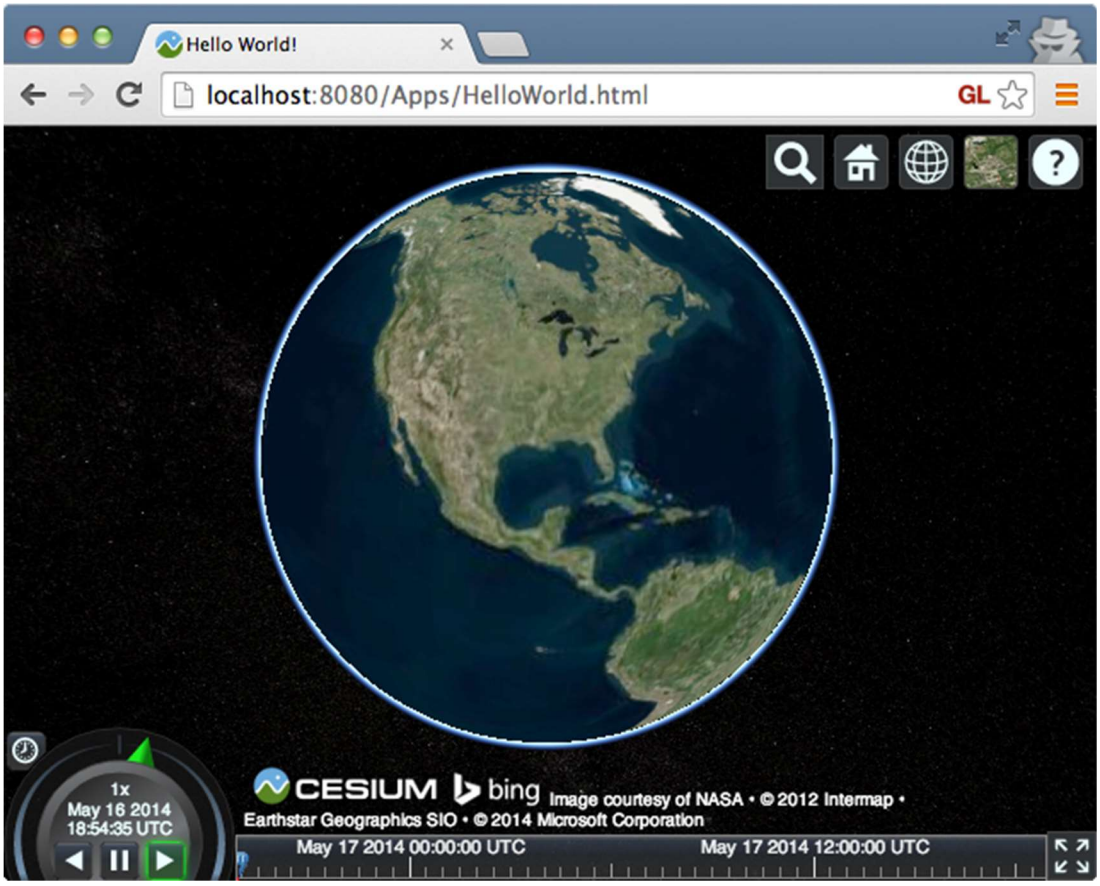

Ilustración 6: Ejemplo de aplicación web desarrollado en CesiumJS. Fuente: https://cesiumjs.org/tutorials/cesium-up-and-running/

4.4.4 Marble

El proyecto Marble [17], parte del proyecto de educación KDE, ofrece un motor de globo virtual gratuito y de código abierto bajo los términos de la licencia GNU LGPL 2. Ha sido desarrollado en C++, proporcionando un motor con menor carga que cualquier solución en Javascript. Además, usa la librería Qt4, por lo que para ejecutarse necesita un sistema operativo como Windows, Linux o Mac OS (Imagen 7).

La plataforma permite al usuario escoger entre los mapas del planeta Tierra, Venus, Marte y otros planetas para ser mostradas en un modelo 3D. Integra soporte para mapas online como

OpenStreetMap [18] y Blue Marble [19] de la NASA. Algunos efectos de visualización también están incluidos en Marble como son la iluminación del sol o la sombra proyectada por las nubes. Algunas de las principales características son:

- Multitud de fuentes de mapas incluídas.
- Búsqueda de dirección, enrutamiento y navegación.
- Posicionamiento y localización.
- Mediciones.
- Eficiente en espacio de disco duro y memoria.

Por contrapartida, la plataforma no puede considerarse realmente 3D, ya que la cámara no puede ser rotada ni inclinada y no se proporciona datos de elevaciones.

Ilustración 7: KDE Marble, centrado en Europa.
Fuente: http://edu.kde.org/marble/

4.4.5 Glob3 Mobile

El proyecto Glob3 Mobile (G3M) [20] es un framework para el desarrollo de mapas 2D o 3D en dispositivos móviles. En un principio, solo estuvo disponible para sistemas Android y iOS. Con la aparición en el tiempo de estándares como HTML5 y WebGL, pudo desarrollarse una versión Web de G3M, proporcionando la posibilidad de creación de aplicaciones para escritorio.

Los fundamentos de G3M se basan en:

- Elaboración de un código fuente eficiente para su uso en dispositivos de bajas prestaciones, así como el aprovechamiento al máximo de las capacidades hardware de los mismos.

- El framework está elaborado de manera que no existan dependencias a otras third-partys, facilitando la inclusión de G3M en sus aplicaciones. Un ejemplo de ello son los gráficos, ya que están soportados por una versión sencilla de OpenGL en el caso de móviles y WebGL en el caso de web.
- Ofrecer total libertad al usuario final con la herramienta. G3M es código abierto y permite a los desarrolladores elaborar sus propios módulos y modificar el núcleo del framework.

Este producto ha sido desarrollado en una colaboración entre la empresa IGO Software y la Universidad de las Palmas de Gran Canaria.

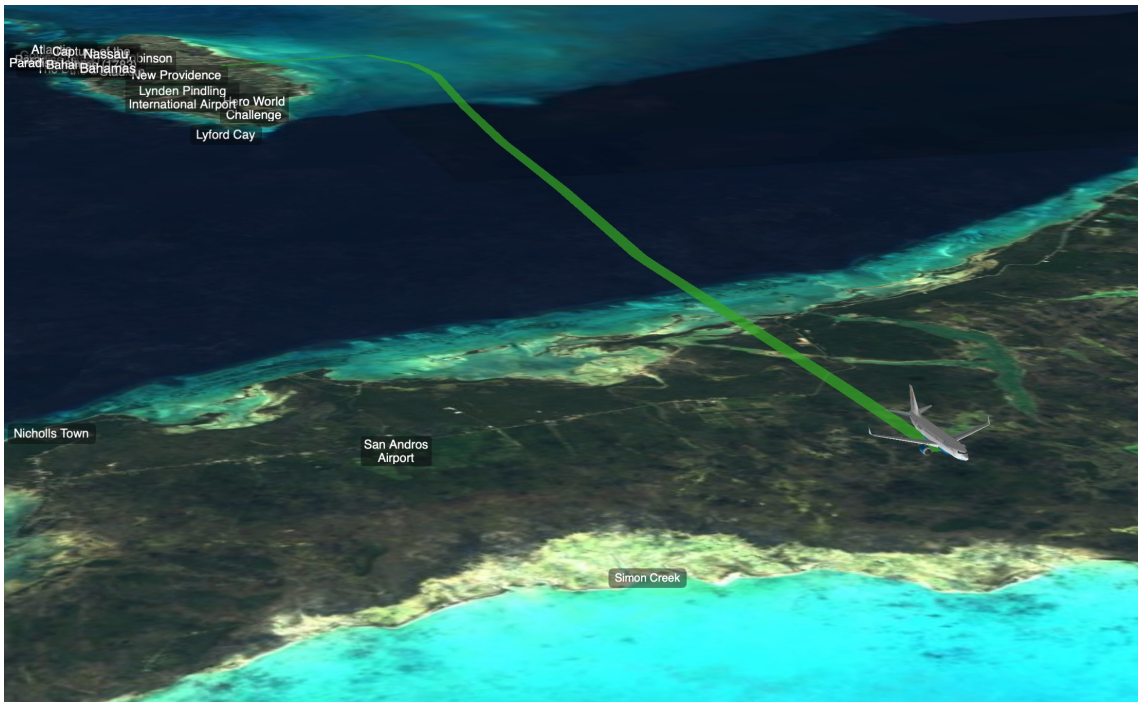

Ilustración 8: G3M, simulación de un plan de vuelo

5. Metodología

5.1 Modelo de proceso de software

La metodología empleada para el desarrollo del proyecto ha sido el desarrollo en espiral. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. En cada versión, se realiza un conjunto de actividades que definen al propio modelo.

El modelo en espiral se divide en un número de actividades llamadas regiones. Cada región esta a su vez dividida por un conjunto de tareas que pueden variar de un proyecto a otro e incluso entre iteraciones (imagen 9).

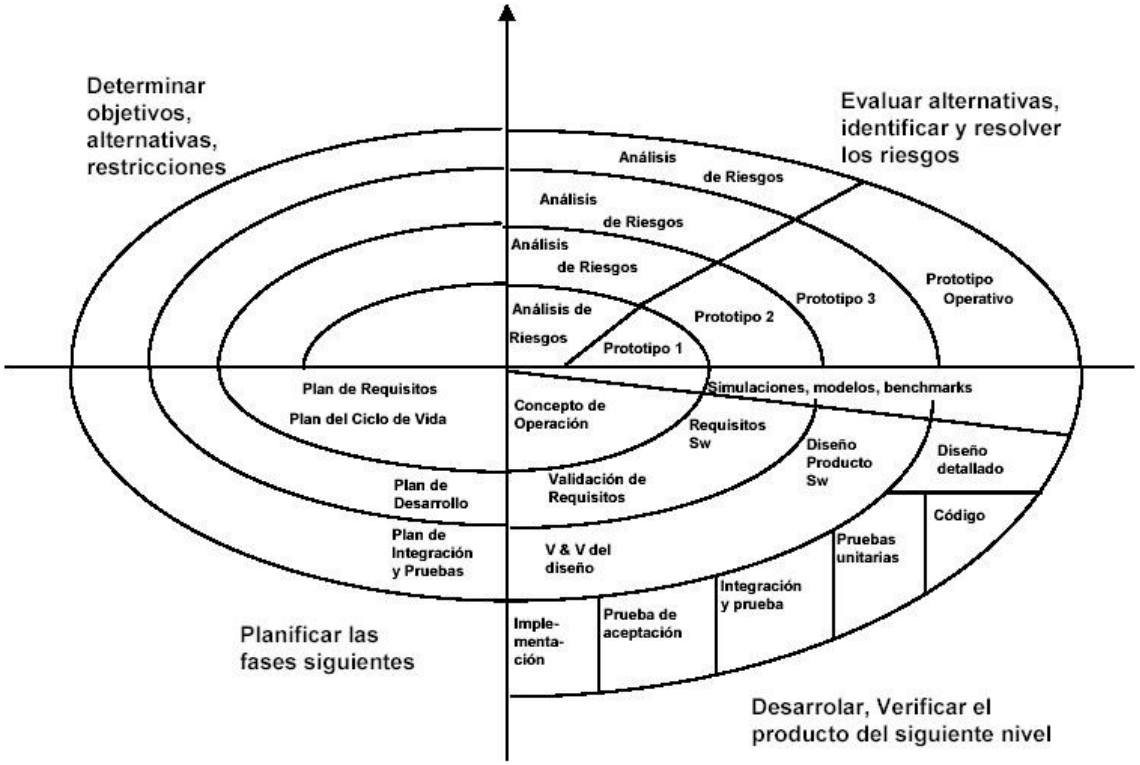

Ilustración 9: Modelo de desarrollo en espiral

Las regiones son las siguientes:

- Determinar objetivos: fijar requisitos, especificaciones y restricciones.
- Análisis del riesgo: estudio de causas de riesgos y alternativas.
- Desarrollar y probar: tareas de la actividad propia y de prueba.
- Planificación: se evalúan los resultados y se planifica la siguiente iteración.

Las principales ventajas del modelo son:

- Reduce riesgos del proyecto.
- Incorpora objetivos de calidad.

- Integra el desarrollo con el mantenimiento.

Las principales desventajas son:

- Largos tiempos de desarrollo.
- Modelo costoso debido al número de versiones.
- Requiere experiencia en la identificación correcta de riesgos.

5.2 Lenguaje de modelado

El lenguaje de modelado utilizado en el proyecto es el Lenguaje Unificado de Modelado (siglas en inglés UML) [21]. Es un lenguaje estándar gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un modelo del sistema que incluye aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones del lenguaje de programación, esquemas de bases de datos, entre otros.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. Los tipos de diagramas pueden ser:

- Estructurales: muestran la estructura estática de los objetos en el sistema. En ella se contemplan diagramas de clases, componentes, etc.
- De comportamiento: muestran el comportamiento dinámico de los objetos en el sistema. En ella se contemplan diagramas de actividades, casos de uso, etc.
- De interacción: muestran la interacción de los objetos en el sistema. En ella se contemplan diagramas globales de interacción, comunicación, tiempo, etc.

Hay que tener en cuenta que UML no se trata de un lenguaje de programación, únicamente representa la realidad de una utilización de un requerimiento.

6. Planificación temporal

En toda propuesta de proyecto fin de carrera se realiza una planificación que corresponde con una estimación temporal de las distintas actividades del proyecto. En el caso de este proyecto, existe un desfase temporal entre la estimación de la propuesta de PFC y el tiempo real consumido por la realización del mismo. Este desfase se debe principalmente por la poca experiencia del alumno a la hora de planificar un proyecto de grandes características. Por ello, se detalla en la tabla a continuación (Tabla 1) una comparación entre la planificación y el tiempo real:

Fases/Actividades	Horas estimadas	Horas reales
Fase 1: Análisis		
Actividad 1.1: Documentación y herramientas	80	100
Actividad 1.2: Análisis de requisitos de usuario	40	40
Actividad 1.3: Análisis de requisitos de software	60	60
Fase 2: Diseño		
Actividad 2.1: Estudio de herramientas	20	40
Actividad 2.2: Diseño de lógica del Plug-in	60	60
Actividad 2.3: Diseño de datos del Plug-in	40	40
Actividad 2.4: Diseño de interfaz de Capaware	40	40
Fase 3: Implementación		
Actividad 3.1: Implementación de lógica	280	310
Actividad 3.2: Implementación de datos	80	80
Actividad 3.3: Implementación de interfaz gráfica en Capaware	80	100
Fase 4: Validación del PFC		
Actividad 4.1: Test de validación	40	40
Actividad 4.2: Validación del usuario final	20	0
Total horas	840	910

Tabla 1: Comparativa entre estimación de horas inicial y tiempo real del PFC

Las principales causas de este desfase son las siguientes:

- Se infravaloró el tiempo necesario a la hora de analizar la documentación de la API de videograbador, así como la documentación proporcionada por el motor geográfico 3D Capaware.

- La familiarización con los entornos tanto de desarrollo como de uso (en concreto, Capaware), ha requerido de mayor tiempo a la hora de diseñar el plugin.
- La implementación de la lógica del plugin ha sido algo más costosa de lo esperado, en especial con la muestra de video.
- No se tuvieron en cuenta ciertos aspectos de wxWidgets a la hora de crear la interfaz gráfica en Capaware, lo que resultó en un mayor esfuerzo.

7. Recursos necesarios

A continuación, se describen los recursos que han sido utilizados en el desarrollo del PFC. Están divididos en dos secciones: hardware y software.

7.1 Recursos hardware

- Equipo de desarrollo: ordenador portátil Macbook Pro propiedad del alumno.
- Equipo de pruebas: PC proporcionado por el tutor en el laboratorio 2 de Sistemas.
- Acceso a servidor de cámaras

Para realizar las pruebas del proyecto era necesario acceder a los servidores (al menos a uno) de cámaras de videovigilancia. Como disponer de un servidor, o el acceso físico al mismo, resultó ser una opción inviable, se llegó a un acuerdo para disponer de un acceso desde un equipo del laboratorio 2 de Sistemas de la escuela de ingeniería informática.

7.2 Recursos software

- Entorno de desarrollo

  A la hora de crear el plugin de Capaware, ha sido necesario descargar los binarios de dicha aplicación y compilarlos. En la documentación de Capaware se recomienda realizar dicha compilación mediante el entorno de desarrollo integrado (IDE en sus siglas en inglés) Microsoft Visual Studio Express [22]. Dicho entorno es gratuito y está orientado principalmente a estudiantes, aficionados a la programación y principiantes.

  A pesar de poder utilizar cualquier versión del mismo, se recomendaba utilizar la versión de 2008 con SP1. Se optó por utilizar esta versión y así evitar futuros problemas con compilaciones tanto de capaware, como del propio PFC.

- Procesador de texto

  En un principio se ha usado Google Docs para las fases iniciales del proyecto. Finalmente se ha trasladado toda la información a Microsoft Word.

- Aplicación para modelado UML

  Los diagramas UML han sido creados en Visual Paradigm Community Edition [23]. Dicho entorno puede descargarse desde su página web oficial de forma gratuita siempre que no sea con fines comerciales.

- Binarios de Capaware

Los binarios de Capaware han sido proporcionados por el alumno, ya que disponía de ellos gracias a un curso impartido en la ULPGC sobre Capaware. Sin embargo, dichos archivos pueden ser descargados desde la página web oficial de la aplicación.

- API de Videograbador

El tutor del PFC ha proporcionado al alumno la API de Videograbador DX8100. Dicha API consiste en librerías de enlace dinámico (DLL en sus siglas en inglés). La API viene acompañada de su correspondiente documentación que describe las diferentes estructuras de datos y funciones necesarias para poder acceder a un servidor de cámaras de videovigilancia DX8100.

8. Análisis

En esta sección se realizará el análisis de la situación actual en la tarea de videovigilancia, además de la API de videograbador, el análisis de requisitos de usuario para la elaboración del PFC y el análisis de requisitos de software.

8.1 Análisis de la situación actual

A la hora de analizar un sistema de videovigilancia, debe tenerse en consideración los aspectos legales de la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD). Aunque en dicha ley no se contemple la palabra videovigilancia, sí que se habla sobre la grabación de imágenes de personas identificadas o identificables.

8.1.1 Ley de protección de datos en la videovigilancia

La grabación de imágenes de personas con fines de vigilancia mediante cámaras, videocámaras o dispositivos similares, constituye un tratamiento de datos personales sometido a la norma de protección de datos. Existen excepciones, como puede ser el tratamiento de imágenes en el ámbito personal o doméstico, en cuyo caso no está sometido a la normativa. Además, no se puede captar imágenes de seguridad de baños, vestuarios o lugares análogos, así como de espacios ajenos.

El responsable debe informar a quien tenga acceso a las imágenes sobre sus obligaciones de seguridad (reserva, confidencialidad y sigilo) y de su deber de guardar secreto. Deben existir medidas que eviten a personal ajeno o no autorizado acceder a las imágenes de seguridad.

El sistema de grabación se ubicará en un lugar vigilado o de acceso restringido. A las imágenes grabadas accederá solo la persona autorizada mediante la introducción de un par de códigos usuario/contraseña.

Destacar que las imágenes captadas por cualquier dispositivo de vigilancia y que sean almacenadas en un videograbador o similares, deben permanecer almacenadas un plazo máximo de un mes. Posteriormente se procederá a borrarlas. Cuando se produjese la grabación de un delito o infracción administrativa, deberán conservarse las imágenes hasta que se dispongan a disposición de las autoridades sin que puedan ser utilizadas para otro propósito.

En resumen, el resultado final deberá contar con un sistema de Autentificación de usuario, que permita la introducción de códigos de usuario y contraseña. Dicha autentificación es única para cada usuario. Por otro lado, todas las imágenes almacenadas en el videograbador, independientemente de que se borren en el plazo estricto o no establecido por la LOPD, no deben ser permitidas su acceso si tienen una antigüedad de más de un mes.

8.1.2 Cliente de videograbador

El cliente de videograbador presenta las clásicas características de todo sistema de videovigilancia:

- Jerarquía de cámaras.

La jerarquía de cámaras es un espacio del cliente reservado a mostrar una representación de la organización de las cámaras. Dicha organización es creada por el administrador que se rige por diferentes criterios, siendo la distribución geográfica la más común. Por ejemplo, si se hiciese una representación del campus de Tafira, las facultades estarían en lo más alto de la jerarquía, seguidas de plantas y accesos a edificios y finalmente las cámaras. Existen otros espacios, como parkings o zonas comunes que pueden ser representadas como un equivalente a facultad.

Otra de las jerarquías más comunes, es la representación de la distribución de los equipos videograbadores. Por ejemplo, un equipo videograbador es capaz de proporcionar cámaras suficientes para la vigilancia de varios edificios, por lo que una sola IP corresponde a varias localizaciones del campus.

- Visualización de las cámaras.

El espacio de visualización es otro elemento común de todo cliente de videovigilancia. Cumple el propósito final de mostrar las imágenes de seguridad de una cámara previamente seleccionada, o de una selección variada de cámaras (mosaico de cámaras). La resolución de las imágenes, así como la tasa de frames, depende tanto de las cámaras utilizadas, como de los equipos videograbadores.

Otro de los modos de visualización, es el acceso en histórico a las imágenes de videovigilancia. Acceder a un momento de tiempo del pasado reciente es uno de los usos más comunes en la videovigilancia. Como ya fue analizado en el apartado anterior, las imágenes de vigilancia de un periodo superior al mes, no pueden ser mostradas por el cliente, independientemente de si dichas imágenes han sido borradas o no, para el cumplimiento de la LOPD.

- Barra de tiempo.

La barra de tiempo permite a los usuarios del cliente de videograbador controlar el modo de acceso en histórico. Permite la introducción de una fecha concreta, reproducir o parar el video, entre otras funcionalidades.

- Panel de control PTZ.

Para todas aquellas cámaras instaladas que cuenten con el soporte de comandos PTZ, suele proporcionarse un panel que permite ejecutar una serie de acciones como pueden ser: mover la cámara según varios ejes, zoom in o zoom out, etc.

Otro de los aspectos fundamentales, son las conexiones con los distintos videograbadores a los que están conectados las cámaras. En el caso del videograbador dx8100, se realizan conexiones a servidores videograbadores, y a través de ellos, se acceden a las cámaras. Cada videograbador dx8100 ofrece la posibilidad de conectar 32 cámaras. Por tanto, el rango de canales por cada IP será siempre desde el canal 0 al canal 31. Por seguridad, no se mencionará el rango de direcciones IP asignada a los equipos videograbadores en el documento. En este caso práctico, el alumno ha podido acceder a una dirección IP y a dos cámaras concretas.

8.1.3 API de videograbador

Una interfaz de programación de aplicaciones o API (del inglés *application programming interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

Una interfaz de programación representa una interfaz de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

Por ejemplo, se puede ver la tarea de escribir "Hola Mundo" sobre la pantalla en diferentes niveles de abstracción:

1. Haciendo todo el trabajo desde el principio:
    1. Traza, sobre papel milimetrado, la forma de las letras (y espacio) "H,o, l, a,M,u, n, d, o".
    2. Crea una matriz de cuadrados negros y blancos que se asemeje a la sucesión de letras.
    3. Mediante instrucciones en ensamblador, escribe la información de la matriz en la memoria intermedia ("buffer") de pantalla.
    4. Mediante la instrucción adecuada, haz que la tarjeta gráfica realice el volcado de esa información sobre la pantalla.
2. Por medio de un sistema operativo para hacer parte del trabajo:
    1. Carga una fuente tipográfica proporcionada por el sistema operativo.
    2. Haz que el sistema operativo borre la pantalla.
    3. Haz que el sistema operativo dibuje el texto "Hola Mundo" usando la fuente cargada.
3. Usando una aplicación (que a su vez usa el sistema operativo) para realizar la mayor parte del trabajo:
    1. Escribe un documento HTML con las palabras "Hola Mundo" para que un navegador web como Google Chrome, Mozilla, Firefox, Opera o Internet Explorer pueda representarlo en el monitor.

Como se puede ver, la primera opción requiere más pasos, cada uno de los cuales es mucho más complicado que los pasos de las opciones siguientes. Además, no resulta nada práctico usar el primer planteamiento para representar una gran cantidad de información, como un artículo enciclopédico sobre la pantalla, mientras que el segundo enfoque simplifica la tarea eliminando un paso y haciendo el resto más sencillos y la tercera forma simplemente requiere escribir "Hola Mundo".

Sin embargo, las APIs de alto nivel generalmente pierden flexibilidad; por ejemplo, resulta mucho más difícil en un navegador web hacer girar texto alrededor de un punto con un contorno parpadeante que programarlo a bajo nivel. Al elegir usar una API se debe llegar a un cierto equilibrio entre su potencia, simplicidad y pérdida de flexibilidad.

La API de videograbador dx8100 consiste en una serie de DLLs, que contienen un conjunto de funciones y estructuras de datos. Mediante el uso de dichas estructuras de datos y funciones, se garantiza el acceso a un servidor videograbador y se permite el acceso a una serie de servicios. A continuación, se mencionan las funciones más importantes para el PFC, ya que permiten realizar las tareas de un cliente de videovigilancia común:

- DxConnect: permite conectarnos a un servidor.
- DxClose: termina una conexión con el servidor previamente conectado.
- DxBeginLiveStream y DxBeginLiveStreamWithAudio: permite la visualización en directo de una cámara seleccionada.
- DxStopLiveStream: detiene la visualización de una cámara en directo.
- DxSearchStart: permite la visualización en histórico de una cámara seleccionada.
- DxSearchStop: detiene la visualización en modo histórico de una cámara.

Las estructuras de datos más relevantes de la API y que mayor interés supone para la elaboración del PFC son:

- Frame_item: estructura que contiene información relativa a un frame concreto de vídeo. Además de contener el frame para un momento concreto del tiempo, proporciona información relativa como un time stamp, formato de video, entre otros.
- Live_vid_conf: estructura que se rellena para acceder a una imagen de video en concreto. Contiene campos como el canal de la cámara y el formato de video.
- St_server_base_info: estructura que se rellena para establecer la conexión con un servidor videograbador. Contiene campos como IP del servidor, usuario y contraseña.

Existen otras estructuras que, por ejemplo, muestran información relativa a una cámara (Imagen 10). Si se desea obtener mayor información de la API, consultar el manual de referencia de la API DX8100.

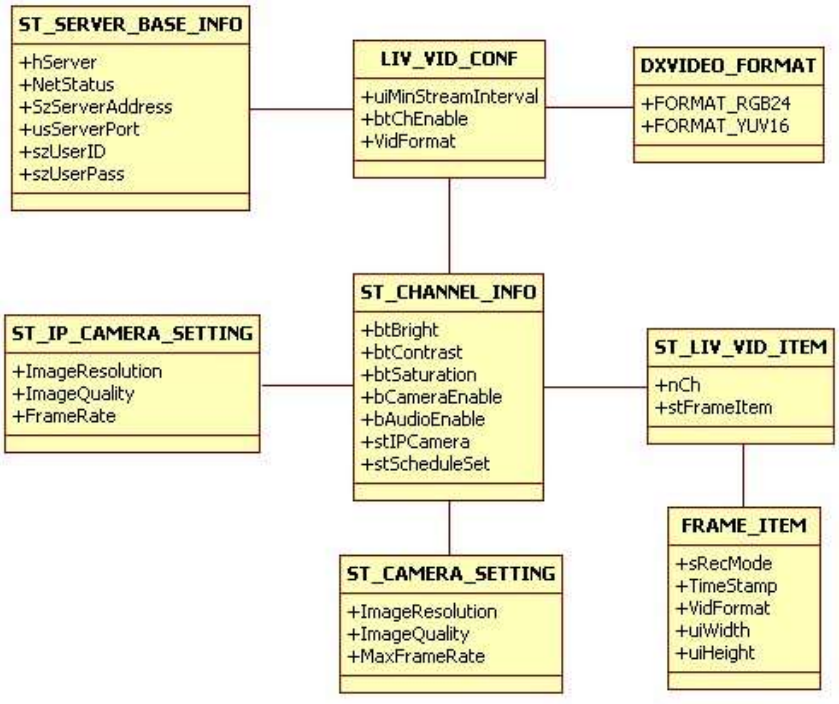

Ilustración 10: Representación de las estructuras de datos de la API

8.1.4 Formato YUV

En la imagen 10 de la anterior página, se puede observar que la estructura de datos DXVIDEO_FORMAT viene preparada para dos tipos de datos: formato rgb24 (8 bits para cada canal) o formato yuv16. Es importante señalar que, a pesar de venir preparado para ambos formatos, solo está implementado el formato yuv16 para el videograbador DX8100 Tiene sentido pensar que esta estructura de datos vino preparada para futuros videograbadores, los cuales darían soporte a otros formatos más modernos.

Por ello, es conveniente realizar un pequeño análisis de YUV y así poder planificar adecuadamente las conversiones a otros formatos. Considerando que muchas librerías abiertas soportan el formato RGB24 clásico, y no tantas soportan el formato YUV, es necesario estudiar las consecuencias de llevar a cabo una conversión.

El formato de vídeo YUV es un espacio de color usado como parte de un sistema de procesamiento de imagen en color. Se codifica teniendo en cuenta la percepción humana, lo que permite un ancho de banda reducido para los componentes de crominancia. Los errores de transmisión o las imperfecciones de compresión se ocultan más eficientemente al ojo humano que con una representación RGB.

El modelo Y'UV define un espacio de color en términos de una componente de luma (luminosidad de la imagen), representada por la Y o Y', y dos componentes de crominancia (información del color), representada por la UV. El símbolo Y' denota la señal de luma con corrección gamma.

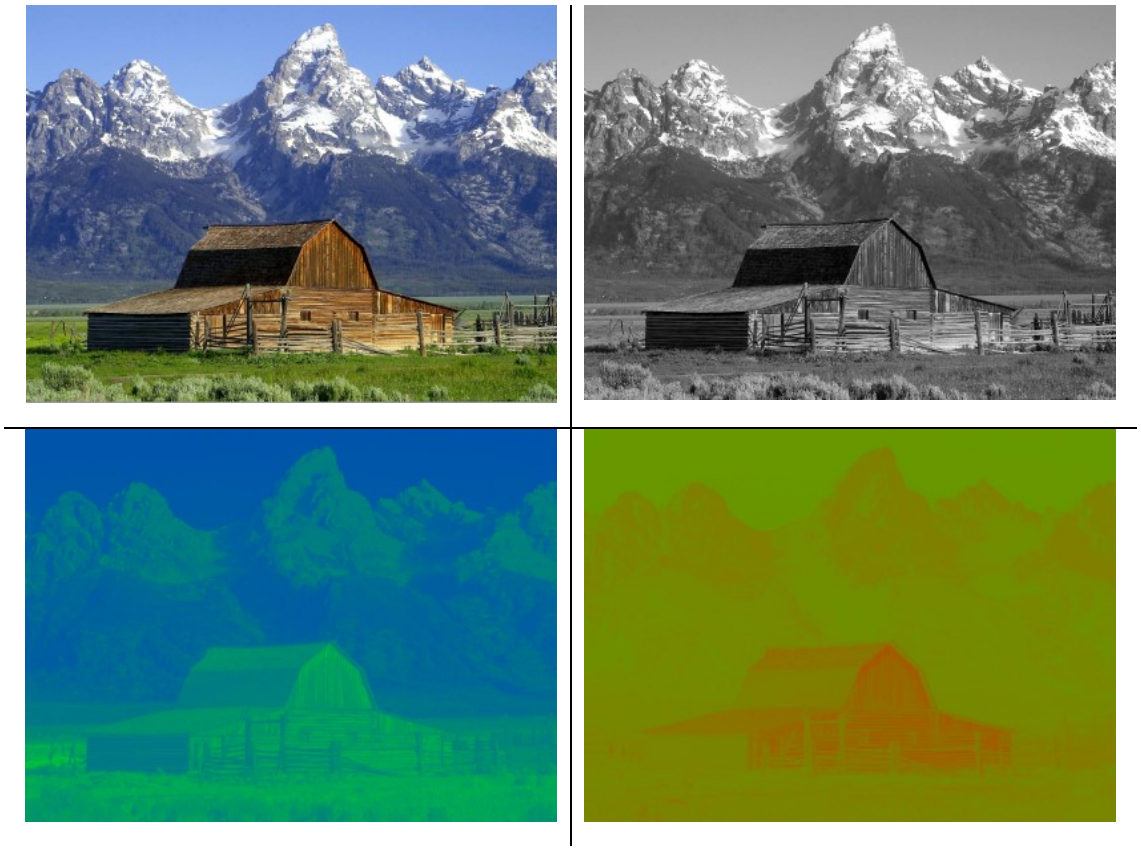

Ilustración 11: Imagen original junto a sus componentes Y' U y V. Fuente: https://commons.wikimedia.org/wiki/File:Barn-yuv.png

Es posible encontrarse con las siglas YPbPr o YCbCr. El espacio de color YPbPr usado en video componente analógico y su versión digital YCbCr se derivan de Y'UV.

El formato YUV puede ser codificado por 12, 16 o 24 bits por píxel. Otra forma de representarlo suele ser como: Y'UV422, YUV411, Y'UV422 o Y'UV420p. En el caso particular del videograbador DX8100, se trabajará con el formato Y'UV422, es decir, un píxel viene representado por dos bytes de información. La codificación del espacio de color Y'UV422 se realiza agrupando cuatro bytes de información para representar dos pixeles:

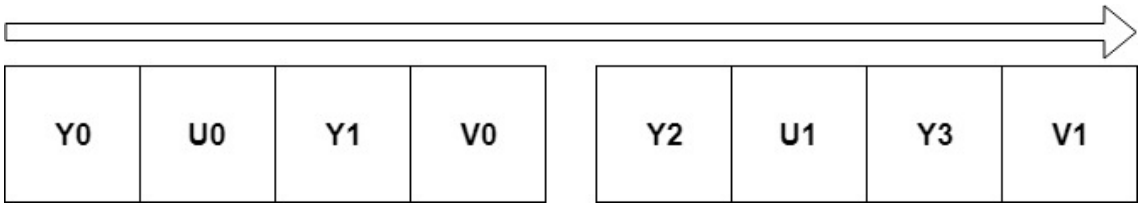

Ilustración 12: Codificación de formato Y'422 representado en FourCC YUY2

Cada cuadrado representa un byte de información. De esta forma se agrupan dos pixeles en cuatro bytes, compartiendo las componentes de crominancia.

Para realizar la conversión a formato RGB24, se leen cuatro bytes de entrada y se producen seis bytes de salida (RGBRGB). Para llevar a cabo dicha conversión, es necesario las siguientes ecuaciones en expresión matricial:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1{,}14 \\ 1 & -0{,}396 & -0{,}581 \\ 1 & 2{,}029 & 0 \end{pmatrix} \begin{pmatrix} Y' \\ U \\ V \end{pmatrix}$$

Ecuación 1: Expresión matricial de conversión de Y'UV422 a RGB24

La aritmética de punto flotante puede llegar a impactar negativamente en el rendimiento, dependiendo de la resolución de imagen a tratar. Para acelerar el proceso, se puede renunciar a algo de precisión y usar las siguientes expresiones:

$$R = clamp\big((298C + 409E + 128) \gg 8\big)$$

$$G = clamp((298C - 100D - 208E + 128) \gg 8)$$

$$B = clamp((298C + 516D + 128) \gg 8)$$

Ecuación 2: Conversión de Y'UV a RGB24 sin punto flotante

La función clamp realiza una poda del valor resultante de R, G y B, acotando el valor al rango [0,255]. Los parámetros C, D y E son el resultado de transformar las componentes de Y'UV de la siguiente forma:

$$C = Y' - 16$$

$$D = U - 128$$

$$E = V - 128$$

Ecuación 3: Transformación de las componentes de Y'UV

Los componentes U y V son señales bipolares que pueden ser positivas o negativas, representando el valor cero los grises. La codificación digital YCbCr escala los valores al rango [16-235] o [0,255], convirtiendo los canales Cb y Cr en valores sin signo, donde el valor 128 representa los grises.

Por último, es importante tener en cuenta que el rango de color y brillo de RGB es bastante inferior al permitido por YUV. Al convertir los valores YUV a RGB es frecuente obtener valores que superen el rango [0,255] (valores inferiores a 0 o valores superiores a 255).

En las expresiones anteriores, se usa una función clamp que acota los valores al rango. Sin embargo, esto produce un cambio en la tonalidad del color. Es importante, una vez obtenida la transformación de formatos, realizar un estudio de otros procedimientos para intentar mejorar la calidad de la imagen resultante, por ejemplo, disminuyendo la saturación del color.

8.2 Análisis de requisitos de usuario

Uno de los mayores inconvenientes del sistema actual para el personal de seguridad es gestionar un número considerable de cámaras con facilidad. Los usuarios deben tener una idea mental o visualizar un mapa estático de la distribución geográfica de las cámaras por el campus. Esto hace que el etiquetado de las cámaras en el cliente de videograbador sea no solo importante, sino además tedioso. Sin embargo, una representación gráfica en 3D de la escena podría simplificar este hecho.

El criterio del administrador a la hora de crear la jerarquía de cámaras puede complicar la tarea a los usuarios en igual o mayor medida. Con un SIG es posible simplificar esto mismo, creando una representación jerárquica de la escena basada en zonas y posiciones de la escena.

A continuación, se detallan las diferentes funciones que ofrecerá el plugin para simular una gestión en la videovigilancia.

8.2.1 Usuarios del sistema

Los usuarios de la plataforma son dos: administradores y personal de seguridad. Los administradores serán los responsables de seguridad del campus de la ULPGC que tienen vínculo directo con la universidad. El personal de seguridad serán aquellos usuarios que desempeñan las labores de videovigilancia. En el caso actual, la universidad utiliza los servicios de la empresa Seguridad Integral Canaria.

8.2.2 Autentificación en el sistema

El plugin deberá contar con la opción de autentificarse en el mismo. A través de un sencillo formulario de login, el usuario introducirá el par de códigos usuario/contraseña otorgados por el administrador. Una vez completado el proceso, el usuario tendrá acceso a las características del plugin.

8.2.3 Añadir servidor

Para un manejo más sencillo, sobre todo a la hora de añadir cámaras, el plugin debe proporcionar una lista de servidores. Dichos servidores se representan mediante un nombre clave, como podría ser “Edificio Informática”, y la dirección IP del servidor videograbador. La

lista debe poder mostrarse visualmente y ser gestionada intuitivamente. Añadir un campo de búsqueda será vital en caso de contar con un número alto de servidores.

Además de las opciones típicas de gestión de listas como modificar o eliminar, acelerar la tarea de añadir un servidor a la lista, sin tener que operar con demasiadas ventanas emergentes puede suponer una característica interesante.

8.2.4 Añadir carpeta contenedora

Para una gestión más eficiente de la escena, la capacidad de añadir elementos "intermedios" como carpetas contenedoras que nos permitan representar una jerarquía de la escena será vital. Estas carpetas no serán elementos visuales en la escena, simplemente mejoran la capacidad de entendimiento del usuario en la jerarquía de cámaras.

8.2.5 Añadir cámara

Una de las operaciones funcionales clave del plugin. La adición de cámaras a la escena debe ser lo más intuitiva posible. Un método sencillo para lograr este fin, puede ser mediante una ventana formulario con diferentes pestañas, que solicite la información necesaria por partes. Evitar la masificación de información ayudará a crear un proceso más intuitivo.

El formulario pedirá al usuario lo siguiente:

- Posición en la jerarquía: Se mostrará una lista con la escena "simplificada", quitando otras cámaras ya previamente añadidas. El usuario elegirá sobre que carpeta contenedora deberá añadirse la cámara.
- Servidor: se mostrará la lista de servidores y el usuario elegirá a que servidor corresponde la cámara a añadir.
- Canal: el canal al que está conectado físicamente la cámara en el videograbador.
- Nombre: indicar el nombre de la cámara.

Finalmente, cuando se vaya a añadir la cámara, se le pedirá al usuario que señale la posición geográfica donde se añadirá la misma. Dicha acción se llevará a cabo mediante el clic del ratón sobre la escena.

8.2.6 Mostrar u ocultar todas las cámaras

Aunque no se trate de una función vital en el plugin, puede resultar interesante añadir a las opciones de vista del sistema, la opción de ocultar (y por contrapartida mostrar) todas las cámaras.

8.2.7 Conectar a servidor / Desconectar de servidor

Cada cámara tendrá la opción de conectarse o desconectarse del servidor. Mediante el proceso de adición de cámaras, toda la información necesaria para llevar a cabo este proceso ya estará guardada por la cámara.

8.2.8 Mostrar video en directo

Cada cámara tendrá la opción de mostrar el video en tiempo real. Se mostrará una ventana con el video correspondiente a dicha cámara.

8.2.9 Mostrar video en modo histórico

Como sucede con el modo directo, las cámaras contarán con la opción de mostrar video en un momento pasado del tiempo. En el caso de intentar mostrar video relacionado con un momento de tiempo superior al plazo de un mes, el plugin deberá mostrar un error.

8.2.10 Mosaico de cámaras

Una de las opciones de visualización más interesantes es la del mosaico de cámaras. El plugin mostrará un formulario donde aparecerá la lista de todas las cámaras disponibles. El usuario irá añadiendo a otra lista, las cámaras que desee que se muestren. Para este caso práctico, se le otorgará al usuario la posibilidad de seleccionar tres opciones diferentes: cuatro cámaras, seis cámaras o nueve cámaras simultaneas. Esta característica estará disponible solo para el modo directo.

8.3 Análisis de requisitos de software

8.3.1 Actores

Los actores son los siguientes:

- Usuario con permiso de acceso: cualquier persona con acceso autorizado a las imágenes de videovigilancia.
    - Administrador: responsable de seguridad de la ULPGC.
    - Personal de seguridad: persona que desempeña la tarea de videovigilancia.

8.3.2 Casos de uso

En la siguiente tabla (Tabla 2), se muestran las diferentes acciones de cada usuario.

Actor	Caso de uso
Administrador	Iniciar sesión
	Añadir servidor
	Añadir carpeta contenedora
	Añadir cámara
	Mostrar todas las cámaras
	Ocultar todas las cámaras
	Conectar a servidor
	Desconectar de servidor
	Mostrar información de conexiones
	Mostrar video en directo
	Mostrar video en histórico

	Mosaico de cámaras
Personal de seguridad	Iniciar sesión
	Añadir servidor
	Mostrar todas las cámaras
	Ocultar todas las cámaras
	Conectar a servidor
	Desconectar de servidor
	Mostrar información de conexiones
	Mostrar video en directo
	Mostrar video en histórico
	Mosaico de cámaras

Tabla 2: Casos de uso por actores

El caso de uso añadir servidor permite al personal de seguridad visualizar la lista de servidores. Además, se le concede el permiso de añadir servidores a la lista, ya que no causa efectos en la escena.

En esta parte del documento, se mostrarán tres ejemplos de casos de uso. El resto de diagramas pueden consultarse en la sección Anexos (en concreto, sección 14.1 de este documento).

- Añadir cámara.

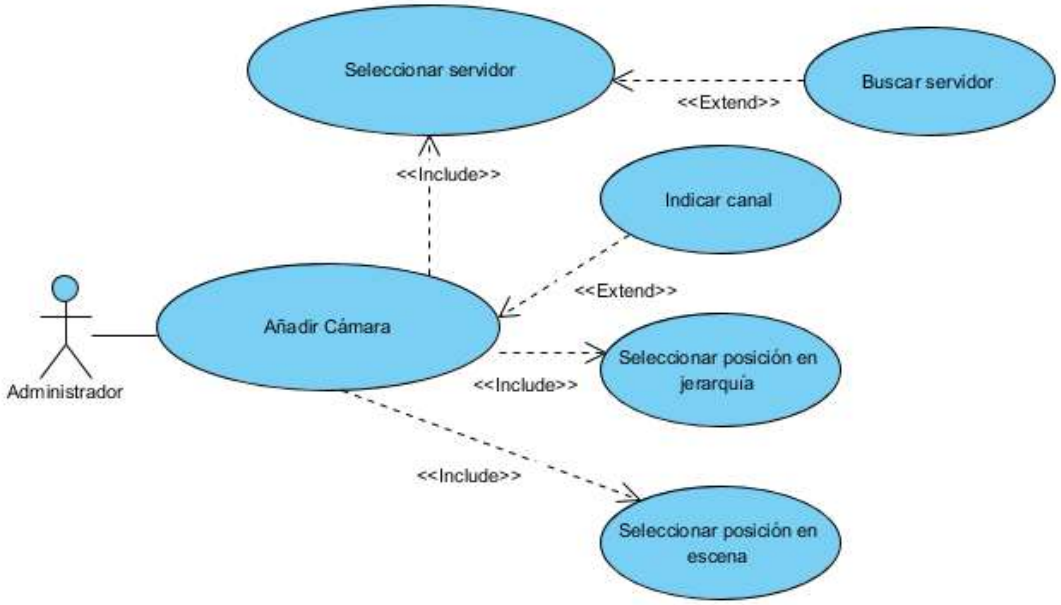

Ilustración 13: Diagrama de caso de uso de añadir cámara

Añadir cámara	
Descripción	Proceso mediante el cual el usuario añade en la escena la representación en 3D de una cámara.
Precondiciones	Usuario identificado en el sistema.
Parámetros	Dirección IP de servidor. Canal. Posición en el árbol de la escena. Coordenadas en la escena.
Comportamiento	El usuario irá rellenando un formulario donde se le irá pidiendo los parámetros necesarios. Si toda la información es correcta, se añadirá la cámara a la escena.
Postcondiciones	Cámara añadida.

Tabla 3: Definición de caso de uso de añadir cámara

- Conectar a servidor.

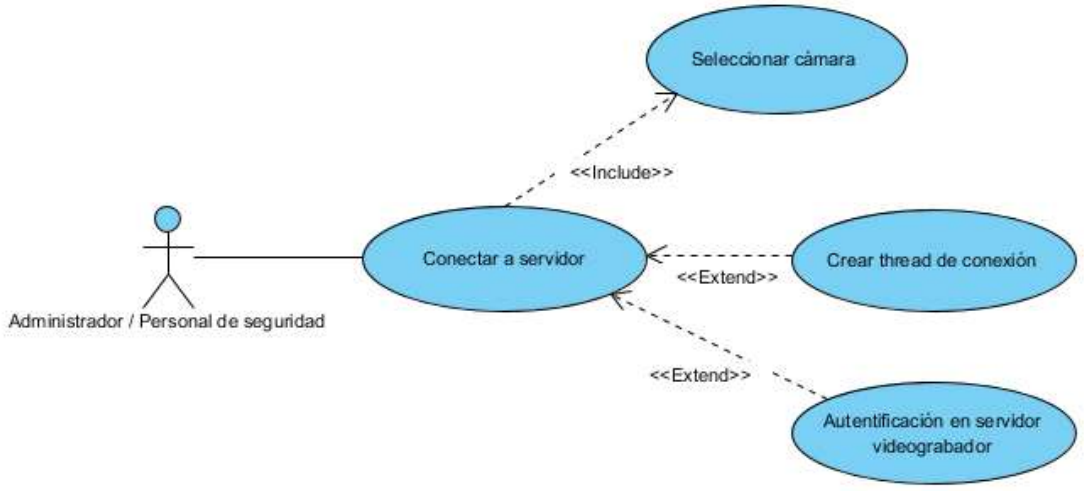

Ilustración 14: Diagrama de caso de uso de conectar a servidor

Conectar a servidor	
Descripción	Proceso mediante el cual el usuario se conecta al servidor videograbador.
Precondiciones	Usuario identificado en el sistema. Cámara añadida.
Parámetros	Dirección IP de servidor. Usuario y contraseña de servidor videograbador. Puerto.
Comportamiento	El usuario selecciona una cámara y se conecta al servidor. Intrínsecamente se realiza un login con el servidor videograbador. El proceso crea un manejador de servidor.
Postcondiciones	Cámara conectada.

Tabla 4: Definición de caso de uso de conectar a servidor

- Mostrar video en directo.

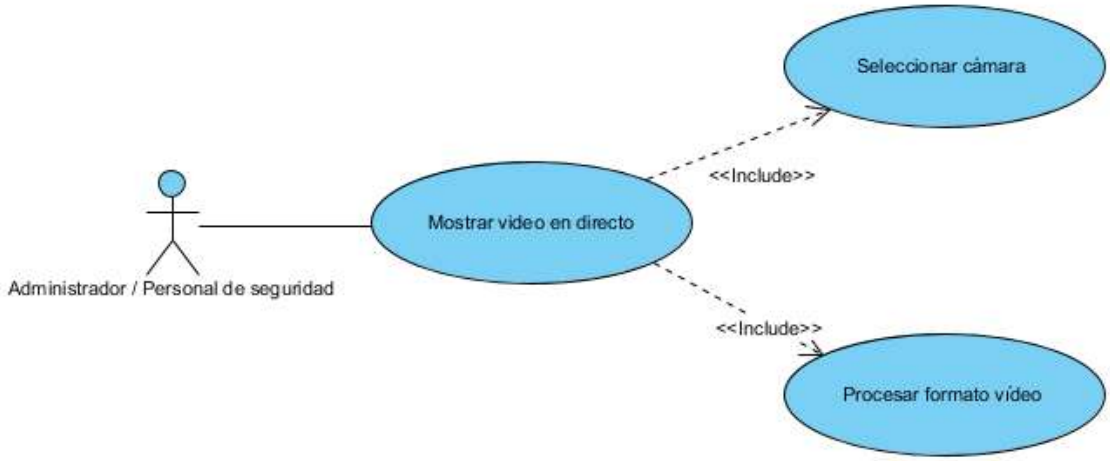

Ilustración 15: Diagrama de caso de uso de mostrar video en directo

Mostrar video en directo	
Descripción	Proceso mediante el cual el usuario muestra un streamming de vídeo de una cámara seleccionada.
Precondiciones	Usuario identificado en el sistema. Cámara añadida.
Parámetros	Manejador del servidor. Canal. Formato de video.
Comportamiento	El sistema muestra en una ventana el vídeo en directo de una cámara previamente seleccionada por el usuario.
Postcondiciones	Vídeo en directo mostrándose.

Tabla 5: Definición de caso de uso de mostrar video en directo

8.3.3 Requerimientos no funcionales

Existe una serie de requisitos no funcionales que debe cumplir el proyecto:

- La tasa de imágenes por segundo de video debe ir acorde a la tasa máxima de frames permitida por el videograbador (sin contemplar el mosaico).
- El sistema incluirá un procedimiento de autorización de usuarios, con el cual deben identificarse mediante un nombre de usuario y contraseña. Solo usuarios autorizados de esta forma pueden acceder al histórico de video.
- Los permisos de acceso al sistema solo pueden ser modificados por el administrador.
- El acceso a las imágenes de video debe ser estrictamente confidencial y no grabar o reproducir a terceros.
- El sistema debe poseer interfaces gráficas bien formadas.
- Debe priorizarse el uso de Software Libre siempre que sea posible.

9. Diseño

Como primer paso del diseño, se especifica la arquitectura del software planteada en base al análisis realizado previamente.

9.1 Arquitectura del software

Para el desarrollo del PFC se ha optado por una arquitectura de cuatro niveles. Se trata de una arquitectura de tipo cliente – servidor, con la particularidad de contar con una infraestructura propia de la red de seguridad de la ULPGC.

- Capa de usuario: capa donde el sistema interactúa con el usuario, recogiendo los datos o peticiones del usuario y mostrando por pantalla los resultados o interfaces.
- Capa de datos: capa donde los datos de la aplicación y del usuario son guardados y gestionados, además de controlar el acceso a los mismos.
- Capa de servidor: capa donde se realiza la comunicación y se llevan a cabos las peticiones a los servidores videograbador de las cámaras.
- Capa de red de cámaras: capa donde se realizan peticiones de video a una cámara en concreto.

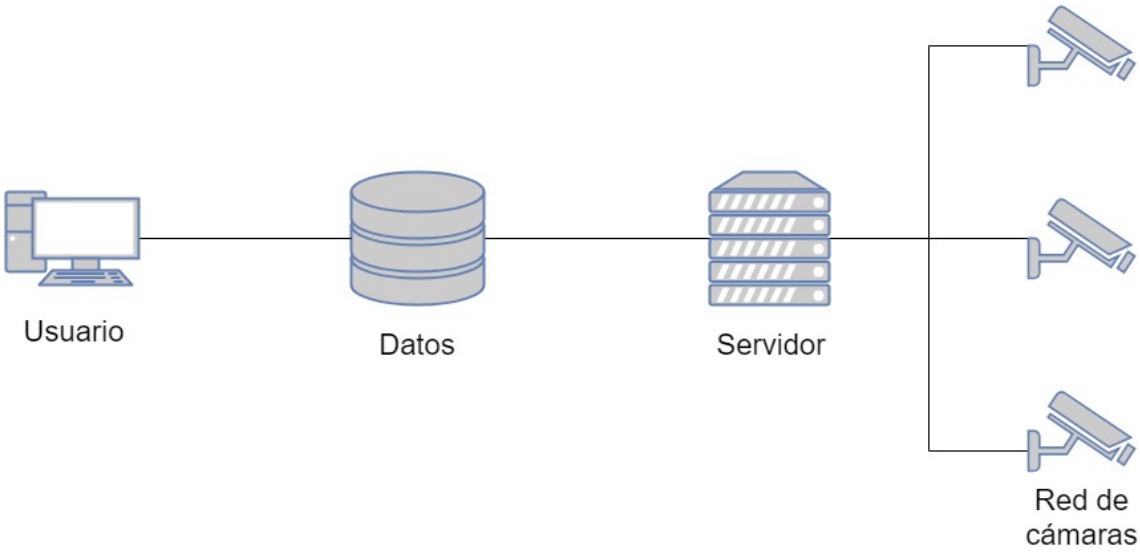

Ilustración 16: Esquema de arquitectura de cuatro capas

La diferencia más significativa con respecto a una arquitectura clásica de cliente – servidor, donde los datos son almacenados en un servidor propio, consiste en que los servidores nos proporcionan el acceso a la red de cámaras y no a una base de datos corriente. Los datos son almacenados de forma local por su correspondiente gestor de base de datos ligado a la aplicación.

9.2 Librería en C++ wxWidgets

Para la implementación del PFC se ha optado por utilizar las bibliotecas multiplataforma wxWidgets. La razón de esta elección consiste principalmente en un diseño continuista con la aplicación principal Capaware, que fue desarrollada utilizando un diseño también con wxWidgets. Se ha considerado importante elaborar el plugin, respetando las decisiones de diseño del sistema original.

WxWidgets es totalmente libre y son usadas mayoritariamente para el desarrollo de interfaces gráficas programadas en C++. Dichas librerías están publicadas bajo una licencia LGPL, similar a GPL, la cual permite desarrollar aplicaciones empresariales sin coste de licencias. Es también compatible con otros lenguajes de programación como Java, Javascript, Python, Ruby, entre otros.

9.2.1 Patrón de diseño MVC (Modelo Vista Controlador)

Siguiendo con la premisa de un modelo continuista con el diseño original de Capaware, uno de los patrones fundamentales para el desarrollo del proyecto (en concreto de las interfaces de usuario) será el bien conocido MVC o Modelo-Vista-Controlador. Este patrón divide una aplicación dada en tres partes interconectadas entre sí, segregándolas según su función:

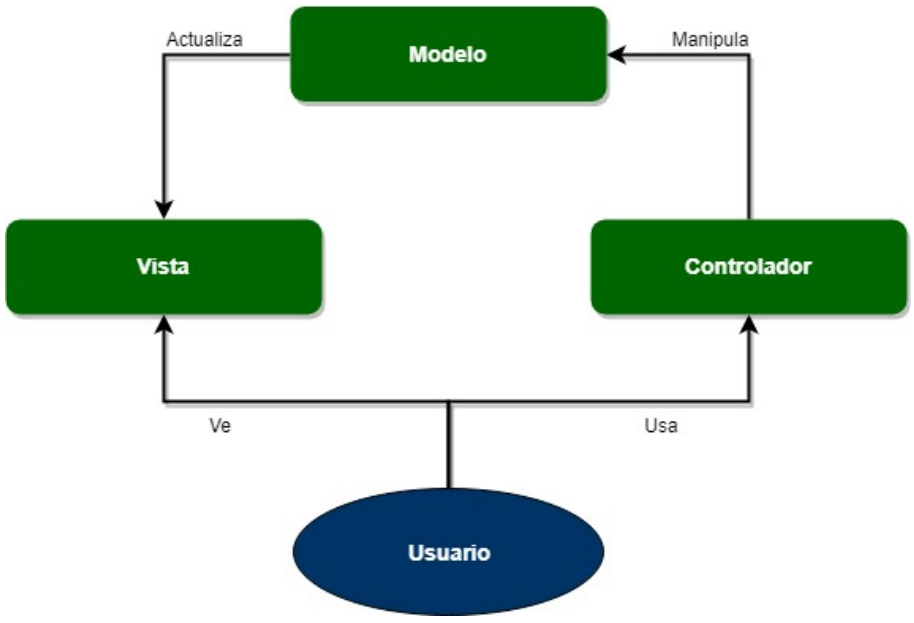

Ilustración 17: Patrón de diseño MVC

- Modelo: es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, implementando también todos los privilegios de acceso descritos por las especificaciones de la aplicación. Envía a la vista la información que sea solicitada por el usuario. Las peticiones de acceso o actualización de los datos llegan al modelo a través del controlador.

- Controlador: su función principal es responder a eventos (acciones del usuario) e invoca peticiones al modelo cuando se realiza una petición de información. También ejerce cambios en la vista, si el usuario solicita un cambio en la forma de distribución de la información procedente del modelo.
- Vista: presenta el modelo en un formato adecuado para la interacción del usuario. Debe ser lo más intuitiva y limpia posible.

9.2.2 Patrón de diseño Observer

Debido a la naturaleza "abstracta" de los servidores videograbadores de Capaware (al no ser un elemento 3D de la escena), es posible que sea necesario la introducción del patrón Observer. Este patrón de diseño define una dependencia del tipo uno a muchos entre objetos, de manera que cuando un objeto cambia su estado, notifica a los demás dicho cambio.

El Sujeto (objeto de datos) contiene atributos mediante los cuales una vista o el Observer, se suscriben a él pasando una referencia a sí mismo. El sujeto mantiene una lista de las referencias a Observers. Los Observers implementan métodos que permiten al Sujeto notificar a su lista de Observers los cambios que haya hecho en alguno de sus estados contemplados. Por tanto, si una vista (o un propio Observer) realiza un cambio de estado en el Sujeto, el Sujeto puede avisar al resto notificándolo en su lista de Observers.

Este patrón es común verlo en desarrollos de interfaces gráficas orientadas a objetos. También es común verlo asociado al patrón de diseño MVC descrito en el anterior apartado.

La dependencia del tipo uno a muchos existe entre las cámaras pertenecientes al mismo servidor videograbador. Si una cámara realiza la operación de conectar al servidor, se deberá actualizar el estado de conexión de todas las cámaras asociadas a ese servidor (si se llevó a cabo con éxito).

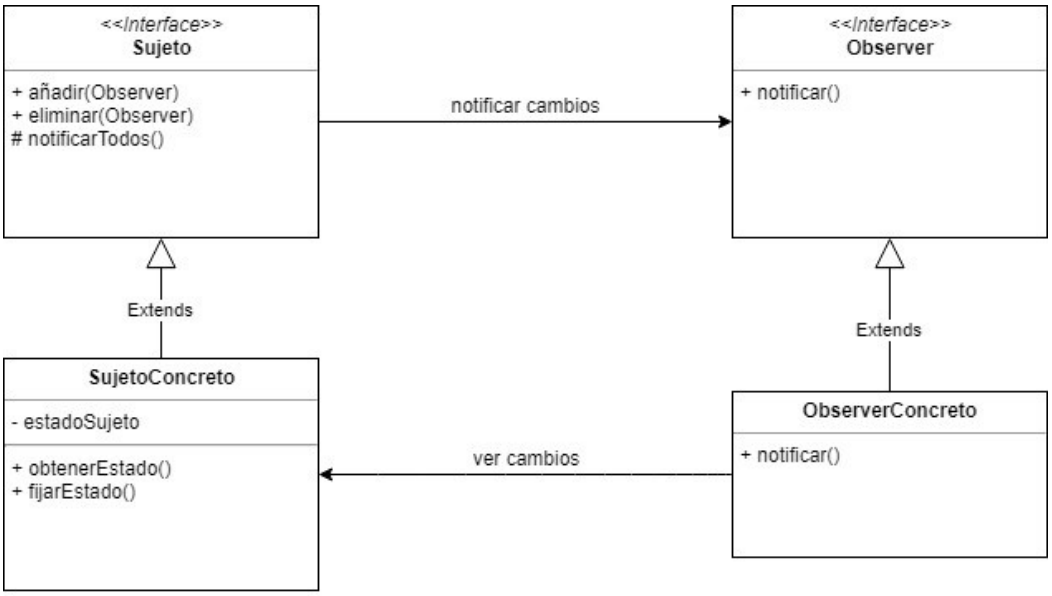

Ilustración 18: Diagrama UML de un patrón Observer genérico

9.2.3 Estructura de una extensión en Capaware con wxWidgets

La creación de extensiones en Capaware permite la introducción sencilla de nuevas funcionalidades en la aplicación mediante el uso de la interfaz de plugins y los módulos a través de la API de CPW.

Mediante una extensión se pueden definir nuevas entidades, insertarlas en el escenario, modificar su comportamiento en el escenario, entre otros. Aspectos deseados para el desarrollo del PFC.

Para llevar a cabo la implementación de un plugin, es necesario crear una serie de funciones y directrices para que el sistema sea capaz de reconocer la librería como una extensión de la aplicación. Dichas funciones a implementar son las siguientes:

- RegisterMenu: Instrucciones a realizar cuando el plugin se registre en la aplicación.
- InitPlugin: Instrucciones a realizar cuando se inicia el plugin.
- ExecPlugin: Operaciones a realizar cuando el usuario selecciona el ítem principal del menú Plugins.
- GetFunctions: Función adicional que introduce todas las funciones personalizadas por el desarrollador de la extensión.

Dentro de GetFunctions indicamos que funciones estamos introduciendo (rellenando una estructura de datos) rellenando parámetros que indican: nombre de la función, título, tipo de función, sobre qué menús enlazo las funciones (se permiten otros menús como Ver, clic secundario de ratón, etc.) y un identificador.

Una vez rellenada la información de GetFunctions, se realiza la implementación de todas las funciones indicándolas como funciones externas. Para más información sobre el desarrollo de extensiones en Capaware, puede consultarse el manual de referencia de la aplicación (Anexo 4).

9.3 Control del tiempo en modo histórico

Capaware cuenta con unos controles que permiten manipular la animación de un elemento 3D en la escena (ver imagen 19). En un plugin, se puede programar el comportamiento que tendrá el elemento en la escena, y, mediante el uso de los controles, manipular dicha animación. El objetivo en el PFC será intentar redefinir estas funciones para adecuarlas al uso de las imágenes de video. Con ello, evitaríamos crear interfaces redundantes en el plugin desarrollado.

Es condición indispensable, que al manipular o redefinir los controles de animación no se produzca un solapamiento de ellos con su funcionalidad original, perjudicándola o incluso inutilizándola. Este requisito es fundamental, o sino, se crearía un problema donde antes no existía, además de ir en contra de los principios de desarrollo de una extensión.

En la figura mencionada, se observa que se dispone de todos los elementos necesarios para realizar una petición al servidor videograbador de histórico. Se cuenta con un calendario para elegir fecha, velocidad de la reproducción (x1, x2, x4…) y los controles típicos de reproducción: play, stop, rewind, flashforward…

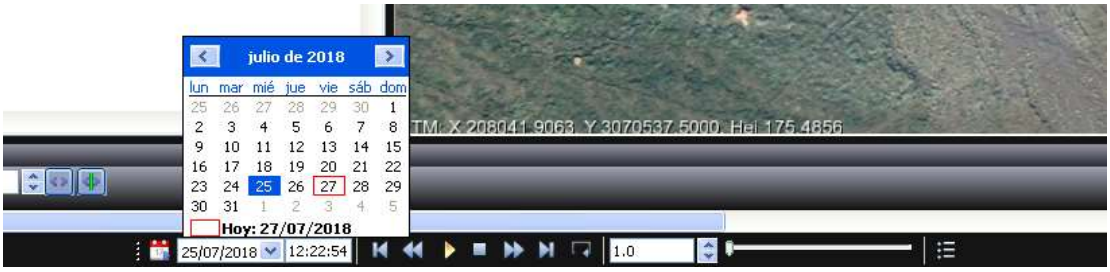

Ilustración 19: Controles de animación de elementos3D en Capaware

9.4 Procesado de imágenes de video

La conversión de YUV a RGB puede provocar una alteración del color como ya se describió en la sección de análisis del formato YUV. Para llevar a cabo un estudio posterior de la calidad de la imagen de video, es interesante planificar un diseño sencillo de algunas estrategias y comparar el resultado final. El flujo que puede verse en la ilustración 20 representa a las técnicas a implementar en este PFC.

En primer lugar, se llevará a cabo la función clamp, que consiste en transformar aquellos valores que sobrepasen los límites del formato RGB24 (8 bits por canal), es decir, valores inferiores a 0 o superiores a 255.

En segundo lugar, sin llevar a cabo el clamp, se realizará un normalizado de la imagen.

Por último, se realizará una segunda transformación del espacio de color a HSV, para llevar a cabo una disminución de la saturación del color.

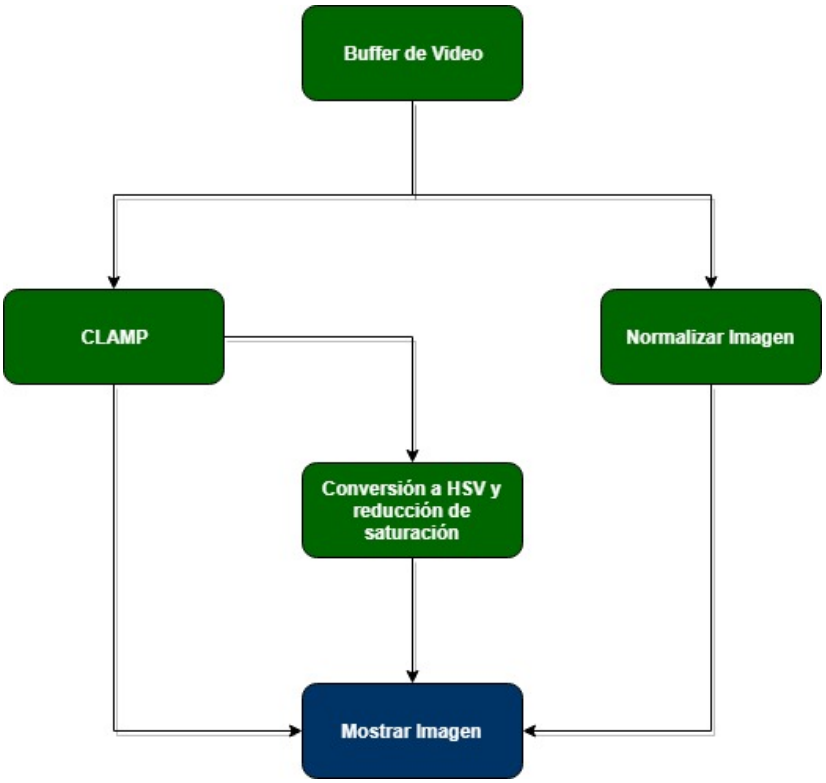

Ilustración 20: Flujo de procesado de imagen de video

9.5 Diseño de tablas adicionales en la base de datos

Capaware permite la instalación de una base de datos más completa mediante MySQL. La filosofía de un plugin se basa en respetar las decisiones de arquitectura o diseño originales y no realizar instalaciones adicionales que perjudiquen la experiencia de usuario. Por ello, se considera de suma importancia seguir un modelo continuista de la aplicación y llevar a cabo la persistencia de los datos del PFC, atacando la misma base de datos de Capaware en MySQL.

En una sesión de Capaware se almacena la información de muchos elementos de ésta en la base de datos (posición de la cámara, elementos de la escena, etc.). Por ello, es necesario crear una tabla adicional para los elementos 3D de cámaras, la cual almacene aquellos datos relevantes a este tipo (servidor, canal…). La ilustración 21 muestra la relación entre las tablas.

Ilustración 21: Relación entre la tabla Element3D y la tabla cameraobject

La tabla cameraobject viene configurada de la siguiente manera:

- Id: clave primaria de valor entero.
- Idcpw: string no nula que representa el identificador único del elemento en Capaware.
- Serchn: texto que contiene el nombre de la cámara (etiqueta), nombre del servidor asociado, dirección IP del servidor asociado y canal al que está conectado, todos ellos separados por el signo almohadilla.

En principio, la relación de herencia entre Element3d y cameraobject requeriría el identificador de la tabla element3d en la tabla cameraobject. Sin embargo, en este caso particular, ya se dispone de otra clave única para identificar al elemento en ambas tablas. Esta clave es idcpw, el cual es un identificador generado por la API CPW de Capaware.

La lista de servidores es otro de los aspectos que necesitan de persistencia. Para evitar que el usuario, cada vez que quiera añadir una nueva cámara, deba introducir de nuevo los datos de los servidores, será necesario crear una tabla adicional en la base de datos para servidor videograbador. La ilustración 22 muestra la tabla cameraserver.

Ilustración 22: Tabla cameraserver

La tabla cameraserver viene configurada de la siguiente manera:

- Id: clave primaria de valor entero.
- Idcpw: string no nula que representa el identificador único del elemento en Capaware.
- Ser: texto que contiene el nombre del servidor (etiqueta) y la dirección IP del servidor, ambos separados por el signo almohadilla.

Como puede observarse, se ha simplificado el diseño de las tablas englobando la información como texto plano, evitando con ello expandir demasiado el número de atributos y aumentar la complejidad de las relaciones entre tablas. Por último, comentar que el signo almohadilla que hace de "separador" de información, será proporcionado por una función del recurso iStorageDevice, dentro de la API de Capaware CPW.

9.6 Diseño de vistas

El desarrollo de una interfaz de usuario lo más limpia e intuitiva posible representa uno de los aspectos fundamentales de este PFC. Por ello, se ha considerado usar una herramienta de diseño de vistas basada en Mockups (maquetas), en concreto, Balsamiq Mockups [24]. Los Mockups permiten la demostración, evaluación y promoción de un diseño. Por tanto, se puede afirmar que un Mockup representa un primer prototipo de lo que se está diseñando.

Balsamiq ofrece tanto una aplicación de escritorio, como plugins para Google Drive u otros, que permiten la creación de diseños de vistas para todo tipo de aplicaciones, ya sean de escritorio, móviles, entre otros.

Con el objetivo de no inundar esta sección con una cascada de imágenes, se ha optado por mostrar sólo el diseño de la vista general, al desplegar el menú Plugins de Capaware. Se puede consultar el resto de ellos, en la sección de Anexos (en concreto, sección 14.2 de este documento).

- Diseño de Vista General

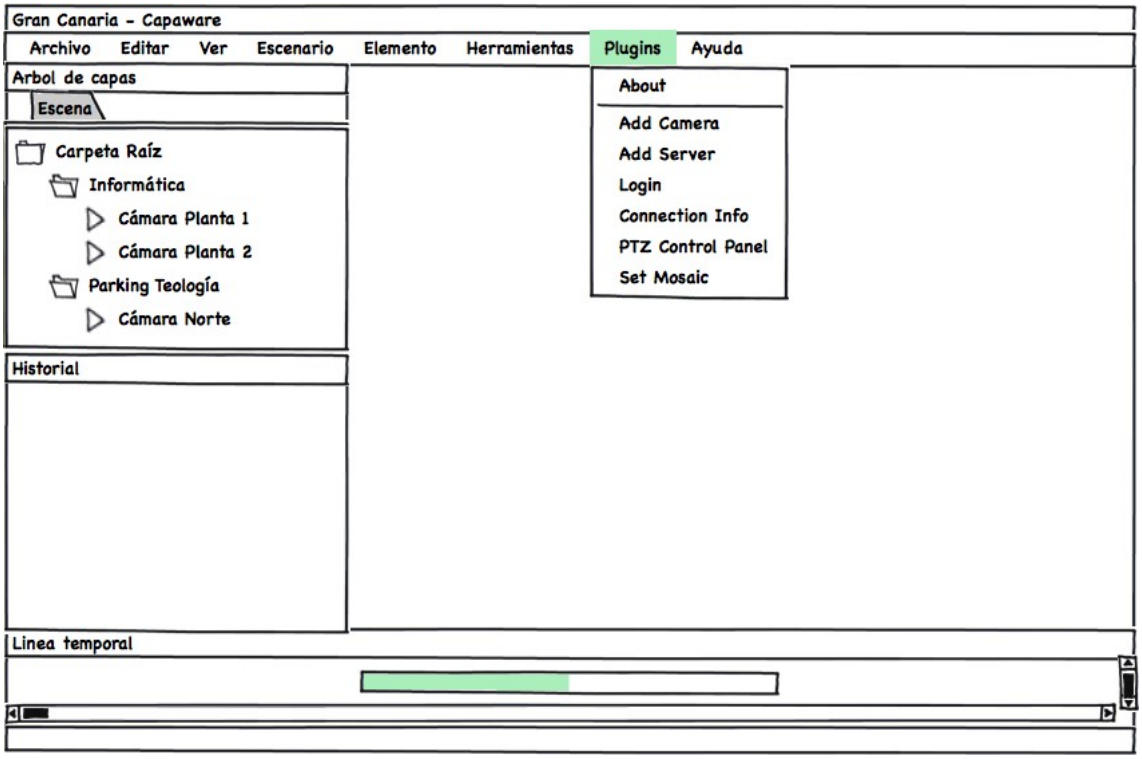

Ilustración 23: Mockup de vista general de Capaware. Menú Plugins

10. Desarrollo

En esta sección, se describe como se ha procedido con la implementación de la persistencia de los datos nuevos en Capaware, así como los módulos implementados para el PFC y la organización de los mismos en la carpeta del proyecto.

10.1 Tablas añadidas en la base de datos de Capaware

La instalación de Capaware requiere de una base de datos mySQL sobre la que se puede almacenar los distintos modelos y elementos añadidos en el escenario para modos de trabajo colaborativo. Tal y como se planteó en el diseño, se ha decidido optar finalmente por la solución continuista, atacando la misma base de datos de Capaware en mySQL y añadiendo en ella las tablas necesarias para el PFC. Si bien es cierto que se podría implementar otra base de datos, se respeta de esta forma la coherencia seguida en la decisión de diseño de implementar las interfaces de usuario con wxWidgets.

La tabla de cámaras es una extensión de la tabla de Element3D, la cual guarda la información asociada a una cámara: (dirección IP asociada, canal, etc.). Por ello, a la hora de decidir cuándo se debe realizar un *Save,* se debe hacer un *Save* también de Element3D. Con ello, guardamos los atributos asociados al elemento (posición en el mapa, modelo, etc.).

A pesar de ser un concepto abstracto dentro de Capaware, la tabla de servidor cuenta con un identificador unitario de Capaware. Cuando se habla de concepto abstracto, se quiere decir que no tiene modelo 3D asociado a el mismo, es decir, no se añade a la escena. Sin embargo, toda cámara va asociada a un único servidor, por lo que se ha decidido hacer así, en el caso de que se quisiera realizar una representación física del servidor en la escena.

Dentro de la tabla de usuarios propia de Capaware, se añadió una entrada de usuario/contraseña que coincidía con el par usuario/contraseña otorgado por el departamento de seguridad del campus para acceder al videograbador. Con esto evitamos que un posible usuario final, deba memorizar dos usuarios y contraseñas diferentes; uno para autentificarse en Capaware, y otro para acceder al videograbador.

10.2 Jerarquía de ficheros del proyecto

Dentro de la carpeta del proyecto, nos encontraremos tres carpetas:

- BasicSDK: carpeta que contiene los ficheros DLL con todas las funcionalidades proporcionadas por la API de Videograbador DX8100.
- Debug: carpeta donde se vuelcan los ficheros generados al realizar una compilación del proyecto, entre ellos, el fichero DLL generado con el plugin de Capaware, de nombre proyecto.dll.
- Proyecto: en esta carpeta se encuentran todos los ficheros fuente del proyecto realizado.

Siguiendo el patrón de diseño MVC, los ficheros se han organizado en diferentes carpetas que representan si son controlador o vista. Dentro de la carpeta proyecto se encuentran los siguientes directorios:

- Controllers: en esta carpeta irán todos los ficheros fuente de los controladores necesarios para aquellas características que hayan sido implementadas siguiendo el patrón de diseño MVC.
- Gui: representa las siglas de *graphical user interface*. Aquí irán todos los ficheros fuente de las vistas implementadas por el plugin siguiendo el patrón de diseño MVC.
- Extra: en esta carpeta se encuentran los ficheros fuente de algunos de los modelos necesarios para el PFC, así como otras funcionalidades implementadas, como puede ser el reporte de errores.

10.3 Módulos del proyecto

Todo el desarrollo del proyecto se distribuye en un total de veintidós módulos. En esta sección, se describe brevemente que función cumple cada uno de ellos, y a su vez, qué clases contiene. Cuando se habla de módulo, se refiere al par de ficheros con extensiones .h y .cpp de C++. La siguiente figura muestra una representación en forma de organigrama de la relación entre los distintos módulos:

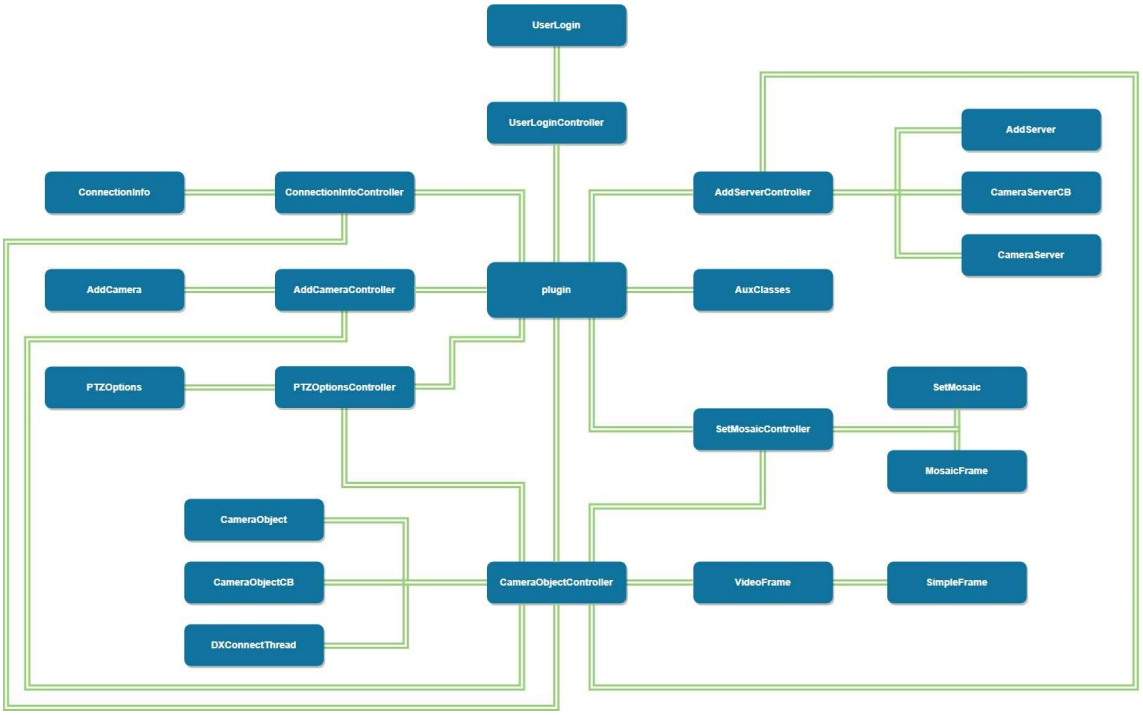

Ilustración 24: Representación de los distintos módulos del PFC

10.3.1 Manejo de cámaras

Una cámara es un elemento 3D del escenario. Los módulos y clases que gestionan el correcto funcionamiento de las cámaras en Capaware son las siguientes:

- CameraObject

  CameraObject representa el modelo de una cámara en Capaware. La clase CameraObject hereda de la clase Element3D de Capaware, con el fin de imitar el comportamiento de un elemento 3D normal del escenario, además de introducir nuevas propiedades propias de las cámaras.

- CameraObjectController

  CameraObjectController es el controlador de la clase CameraObject. Además de la clase con el mismo nombre del módulo, contiene la clase CameraObjectControllerReceiver, la cual gestiona eventos de ratón.

- CameraObjectCB

  CameraObjectCB gestiona las funcionalidades básicas de acceso a la base de datos de Capaware para almacenar o cargar la información de cámara.

Capaware tiene la capacidad de introducir elementos 3D en la escena y poder llevar a cabo las operaciones más comunes: traslación, rotación, animación, etc. Además, cuenta con su propia gestión de la persistencia en una base de datos ya implementada. Esto supone una gran ventaja a la hora de introducir el concepto de cámara en el PFC. Partiendo de esta base, se registra en la configuración de la aplicación Capaware mediante la clase PluginResources (descrita en el apartado 10.3.10), accediendo a la tabla de entidades de Capaware y añadiendo nuestra nueva clase CameraObject.

La implementación de esta clase se realiza mediante una herencia de la clase Element3D de Capaware (ver imagen 25). Gracias a ello, podemos despreocuparnos por la gestión de las acciones típicas de traslación, rotación, etc., además de propiedades como posición geográfica en la escena (coordenadas UTM), modelo 3D asociado al elemento, entre otros.

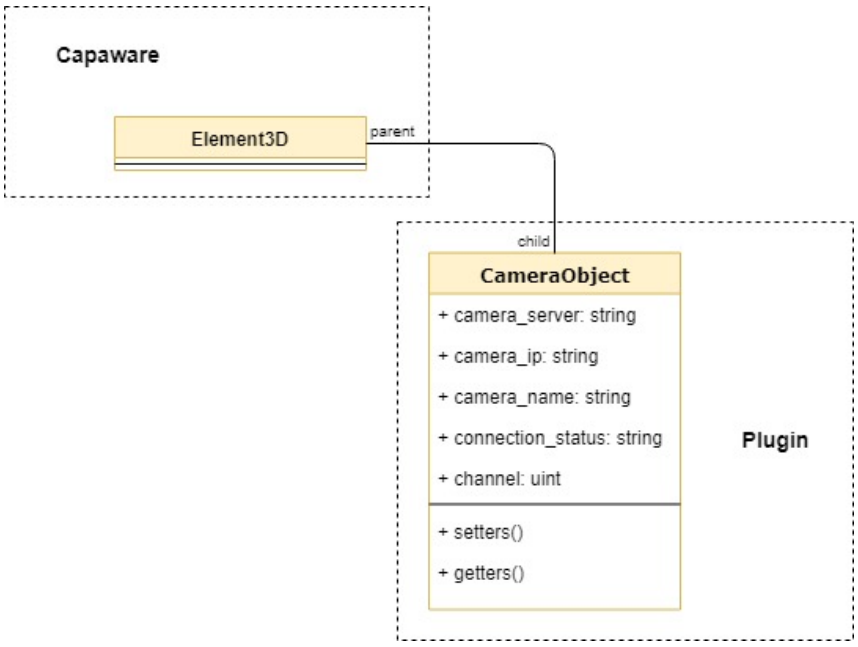

Ilustración 25: Diagrama de clase CameraObject, herencia de element3d

Sin embargo, es necesario introducir la nueva información a tratar. Es por ello, que se extienden las propiedades ya definidas por Capaware de un elemento3D, añadiendo: nombre del servidor al que está conectada la cámara, dirección IP del servidor, nombre de cámara, canal al que está conectado y, finalmente, estado de conexión. Al tratarse de un modelo de datos, las únicas funciones públicas implementadas son setters y getters (obtener y establecer información), además de otras funciones características del lenguaje (sobrecarga de operadores, etc).

Por otro lado, como se está definiendo un nuevo modelo de datos, el cual hereda de un modelo ya definido con persistencia en base de datos, es necesario llevar a cabo una implementación nueva de la persistencia de CameraObject. Exceptuando el estado de conexión, toda la información relacionada con la cámara, es guardada en la base de datos mediante la clase CameraObjectCB.

Como se está llevando a cabo una implementación siguiendo un modelo MVC (sin vista en este caso), es necesaria la implementación de un control, el cual lo lleva a cabo la clase CameraObjectController. Esta es una de las clases fundamentales del proyecto, ya que cualquier acción que involucre a una cámara, es gestionada por este control. Contiene instancias de muchas otras clases del proyecto e implementa la gran mayoría de peticiones al servidor videograbador DX8100 mediante la API de videograbador. Funcionalidades como mostrar video, acceder al histórico, conectar a servidor…

CameraObjectController se soporta en la clase CameraObjectControllerEventReceiver, la cual permite capturar eventos de ratón. Esta clase es necesaria para llevar a cabo la adición de una cámara a escena. En el apartado 10.3.2 hay más información de este proceso.

Otra de las funcionalidades que se llevan a cabo en el controlador CameraObjectController es la de realizar la conexión y desconexión con el servidor videograbador. La API de videograbador necesita además de la implementación de varias funciones *callback* adicionales para gestionar los posibles resultados de las peticiones de conexión al servidor. Estas funciones *callback* cuentan con ciertas particularidades en el lenguaje C++ (funciones estáticas, manejo de estados dentro de las estructuras de datos de la API y otros), las cuales han dificultado un poco la elaboración de esta característica. Dichas funciones son: la gestión del resultado de la petición de conexión al servidor (viene indicada por un estado en la estructura de datos necesaria para llevar a cabo el proceso) y la gestión ante la pérdida espontánea de la conexión durante la sesión.

En este punto, surge una pequeña discrepancia entre “filosofías de trabajo”. Por un lado, en Capaware trabajamos en un entorno 3D con elementos 3D en la escena. Por otro lado, la API de videograbador trabaja enfocada a servidores, el cual es un concepto que no es “físico” en Capaware.

Aunque existen varias posibles soluciones a este pequeño problema, en su momento se pensó en dos alternativas: o bien, se hace una imitación de la solución que proporciona la mayoría de aplicaciones de videovigilancia en el mercado, creando interfaces con la lista de servidores y estableciendo conexiones con cada uno de ellos; o se implementa un mecanismo que garantice que, al establecer la conexión al servidor con una cámara, todas las cámaras compartan el mismo estado. Finalmente, se decidió por esta última, introduciendo el patrón de diseño Observer, el cual permite resolver este problema.

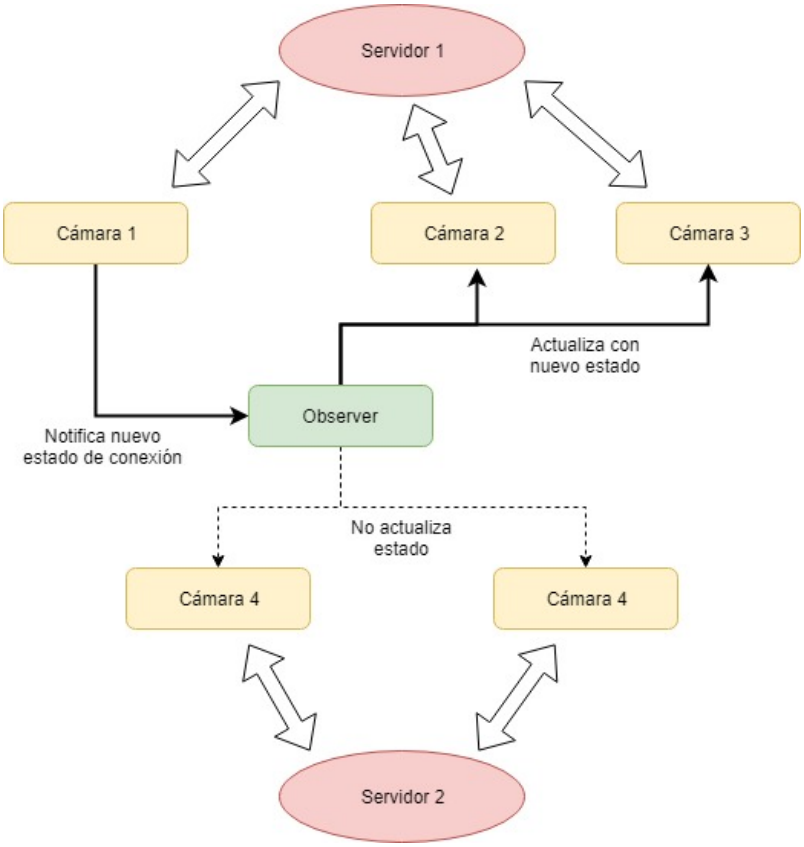

Ilustración 26: Representación del proceso de actualización de estado de conexión

En el proyecto, cuando el usuario establece conexión con un servidor videograbador a través de una cámara, si la conexión es satisfactoria, se realiza un *broadcast* con el nuevo estado de conectado, a todas las cámaras pertenecientes al mismo servidor. Una representación de este proceso se puede ver en la figura previa. La dirección IP es lo que se usa como clave unitaria para este proceso.

El mismo proceso sucede cuando se realiza tanto la desconexión manual con el servidor de una cámara, como con una pérdida en la conexión y se llame a una de las funciones *callback* implementadas para gestionar este evento.

El resultado de las peticiones de conexión puede verse de dos maneras: o accediendo a la información de conexión (apartado 10.3.5) o mediante la línea de comandos que se inicia con la ejecución de Capaware (esta opción se inició como ayuda en la depuración), como se muestra en la imagen a continuación:

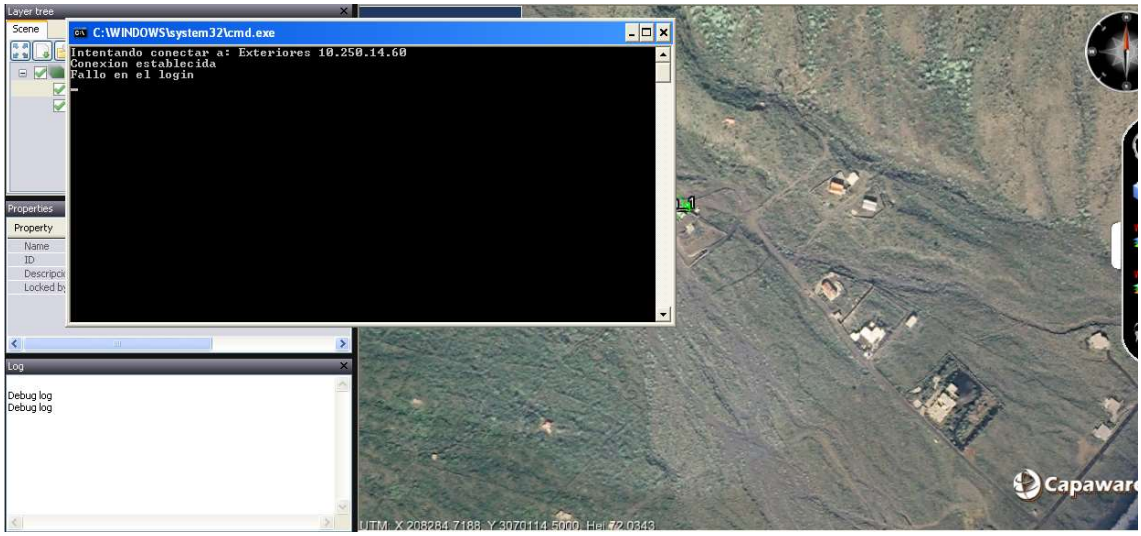

Ilustración 27: Resultado de una petición de conexión en el cmd iniciado por Capaware

10.3.2 Adición de cámaras a la escena

Añadir una cámara a la escena no puede hacerse con la simple adición de un elemento 3D, ya que se requiere información adicional de la cámara (información de conexión principalmente):

- AddCameraController

  AddCameraController es el controlador de la clase AddCamera. Contiene una única clase, cuyo nombre coincide con el nombre del módulo.

- AddCamera

  AddCamera es el módulo que proporciona la vista, con el fin de pedir al usuario a través de una GUI, toda información adicional necesaria mencionada previamente. Contiene las clases CameraScene, CameraOptions y CameraAdition, representando cada uno de los pasos necesarios para llevar a cabo la funcionalidad.

La adición de una cámara en la escena es un proceso suficientemente complejo para llevar a cabo una separación de los controles de esta funcionalidad con el controlador CameraObjectController. Si bien están estrechamente ligados, ya que CameraObjectController proporciona acceso al modelo de cámara, todo el control de las interfaces implicadas se lleva a cabo mediante AddCameraController.

El primer elemento a tener en cuenta a la hora de añadir una cámara en la escena, es la jerarquía de la escena (ver ilustración 28). Capaware tiene implementado un árbol jerárquico de la escena, el cual permite introducir “padres” en él mediante el concepto de carpeta contenedora, la cual no es más que una herramienta que permite agrupar elementos de un mismo tipo o alguna relación semántica dada por el usuario. Esta característica es sorprendentemente similar al aspecto de una jerarquía de cámaras en una aplicación de videovigilancia. Por ello, accediendo a los recursos de la API de Capaware mediante la clase PluginResources, obtenemos acceso al árbol de la escena (mediante un puntero a la posición más alta llamada *Top Layer* en Capaware).

Ilustración 28: Representación de la escena en la aplicación Capaware

AddCameraController lleva el control de la vista implementada en AddCamera. Dentro de AddCamera se encuentran las tres clases definidas arriba (en la descripción del módulo al comienzo de este apartado). Cada una de estas clases implementa una de las pestañas de la vista AddCamera. Esto se ha hecho así, debido a la naturaleza de wxNotebook de wxWidgets. Ayuda a simplificar la complejidad de este elemento particular.

Especial hincapié en dos listas que se muestran en dos pestañas concretas. En la pestaña de Buildings, implementada por la clase CameraScene, se muestra una lista con la escena. En esta lista, solo se muestran las carpetas contenedoras de donde puede "colgar" una cámara (ver imagen 29). Al seleccionarla en la lista, la cámara será añadida en ese punto del árbol de la escena.

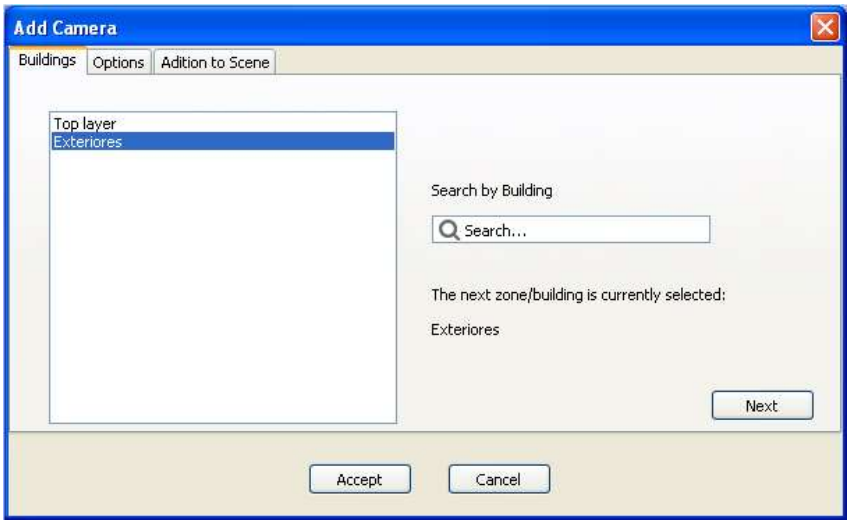

Ilustración 29: Pestaña buildings, lista con "padres" del árbol de escena

La segunda lista se observa en tercera pestaña, implementada por la clase CameraAdition (ver imagen 30). En esta lista, aparecerán los elementos 3d que son hijos del padre seleccionado en la primera lista en la pestaña CameraScene. Si, por ejemplo, se están añadiendo múltiples cámaras hijos directos de *Top Layer*, irán apareciendo en esta lista como hermanos de la nueva cámara que se añada.

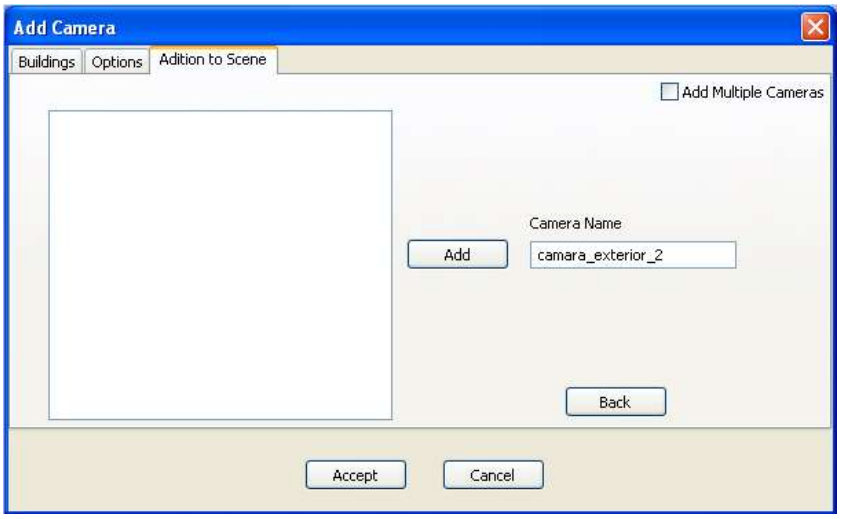

Ilustración 30: Pestaña adition to scene, lista con hijos del padre seleccionado

En la segunda pestaña (Options), implementada por CameraOptions, se introduce el número de canal al que está conectada la cámara, además de seleccionar el servidor de la lista de servidores videograbadores añadidas previamente (o incluso llevar a cabo una adición de servidor justo en ese momento). Tanto el proceso de selección de un servidor como el proceso de gestión de servidores se realiza en la misma vista. Más información sobre este proceso en el apartado 10.3.4.

Al pulsar el botón Add en la tercera pestaña (Adition to Scene), el plugin tomará el control de los eventos de ratón del usuario para registrar la posición donde el usuario haga clic en la escena (la posición donde quiere añadir la cámara en la escena). Es aquí donde AddCameraController se apoya en el controlador de CameraObjectController para llevarlo a cabo.

Como último paso, AddCameraController realizará el proceso de inicialización de los distintos atributos de element3d. así como de los atributos de CameraObject para llevar a cabo la adición en la escena. Una representación del árbol de la escena final siguiendo el ejemplo seguido al añadir la cámara de nombre *cámara_exterior_2* sería el siguiente:

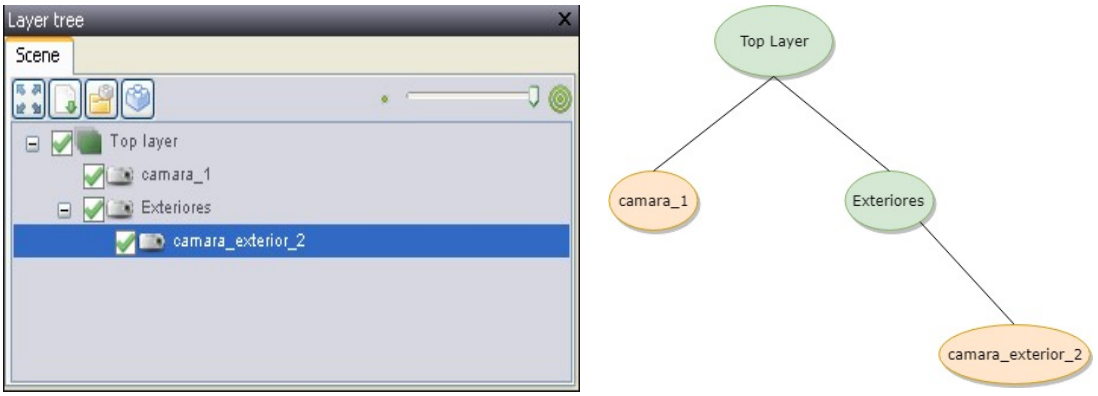

Ilustración 31: Escena final y representación del árbol

10.3.3 Autentificación de usuario

La autentificación de los usuarios en Capaware ya consta de modelo, ya que la aplicación ya proporciona una tabla de usuarios en su base de datos. Por ello, es recomendable hacer coincidir las credenciales de Capaware con las del servidor Videograbador.

- UserLoginController

  UserLoginController es el controlador de la clase UserLogin. Contiene una única clase, cuyo nombre coincide con el nombre del módulo.

- UserLogin

  UserLogin proporciona la vista para llevar a cabo la autentificación de usuarios en Capaware.

La clase UserLoginController implementa las funciones que consultan la tabla de usuarios en la base de datos de Capaware. Recogiendo las credenciales del usuario mediante el par Usuario – Contraseña a través de la vista implementada por UserLogin, busca coincidencias en la tabla correspondiente. Si encuentra al usuario, establecerá el usuario en la aplicación Capaware. Este usuario y contraseña es el que se usará para acceder al videograbador, por lo que las credenciales deben coincidir y no provocar una redundancia de datos o de interfaces que perjudiquen la experiencia de usuario.

10.3.4 Gestión de servidores de videograbación

Los servidores de videograbación representan un elemento abstracto en Capaware y requieren de un modelo, con el fin de almacenar la información en la BBDD de Capaware.

- AddServerController

  AddServerController es el controlador de la clase AddServer y CameraServer. Contiene una única clase, cuyo nombre coincide con el nombre del módulo.

- AddServer

  AddServer representa la vista con la que el usuario añade información de un servidor videograbador a almacenar en Capaware.

- CameraServer

  CameraServer contiene el modelo de un servidor videograbador. A efectos prácticos, un servidor videograbador consta de una dirección IP y una etiqueta.

- CameraServerCB

  CameraServerCB gestiona las funcionalidades básicas de acceso a la base de datos de Capaware para almacenar o cargar la información de servidor videograbador.

La necesidad de otorgar persistencia a la lista de servidores videograbadores que vayan registrando los usuarios, causó la creación de las clases CameraServer y CameraServerCB. CameraServer gestiona la información básica de los servidores: nombre del servidor y dirección IP. Cuenta además con las funciones getters y setters clásicas de un modelo de datos.

A pesar de que, como se ha dicho anteriormente, el concepto de servidor videograbador es abstracto dentro de Capaware (no representa a un elemento visible dentro de la escena), se ha decidido otorgarle un identificador de capaware TypeId. Con ello, otorgamos al servidor un

identificador unitario, el cual resulta más manejable que dos cadenas string o el string combinado con el par nombre de servidor y dirección IP. Si bien, la dirección IP representa ya una clave de por sí (no tiene sentido una misma dirección IP con dos sobrenombres), se ha optado por otorgarle otra clave en caso de no haber contemplado otras posibilidades.

La clase AddServer implementa una vista que soporta dos acciones: por un lado, permite la adición de servidores a la lista por parte del usuario, introduciendo el susodicho par nombre de servidor y dirección IP; y, por otro lado, permite la selección de un servidor de la lista (ver imagen 32).

Ilustración 32: Interfaz de AddServer

La principal razón de ello, consiste en simplificar un poco el número de vistas resultantes y evitar duplicidad. Ambas supuestas vistas mostrarían la misma lista, tendrían el mismo campo search de búsqueda, etc.

La clase AddServerController implementa el control de todas estas clases, permitiendo gestionar la lista (cambiar nombre o IP, eliminar un miembro de la lista, buscar en la lista…).

10.3.5 Información de conexión de las cámaras

Una simple ventana donde se muestra el estado de conexión y propiedades de las cámaras añadidas en la escena.

- ConnectionInfoController

  ConnectionInfoController es el controlador de la clase ConnectionInfo.

- ConnectionInfo

  ConnectionInfo es el módulo que muestra una vista con las cámaras de la escena y su información de conexión asociada (a qué servidor está conectado, canal y el estado de la conexión en ese momento).

Como con Capaware se tiene una vista del entorno virtual con una disposición de las cámaras, es necesario proporcionar alguna herramienta donde podamos ver los detalles asociados a las cámaras (obviando otros elementos3d de la escena como posibles edificios u otros insertados

en ella). La clase ConnectionInfo proporciona una vista, donde se muestra una lista con todas las cámaras de la escena. Al hacer clic con el ratón en alguna de esas cámaras, puede verse toda la información asociada al modelo de datos de CameraObject: nombre del servidor, dirección IP del servidor, canal al que está conectado y, por último, el estado de conexión (ver imagen 33).

ConnectionInfoController es la clase que lleva el controlador de la vista y accede a los getters del modelo de datos de CameraObject a través de su correspondiente controlador. Es importante señalar, que esta herramienta refleja el funcionamiento del patrón Observer implementado en el PFC. Como ya se describió en el apartado 10.3.1 Manejo de Cámaras, Si se ha establecido la conexión con una cámara concreta al servidor, todas las cámaras pertenecientes al mismo servidor deben reflejar el estado *Connected*.

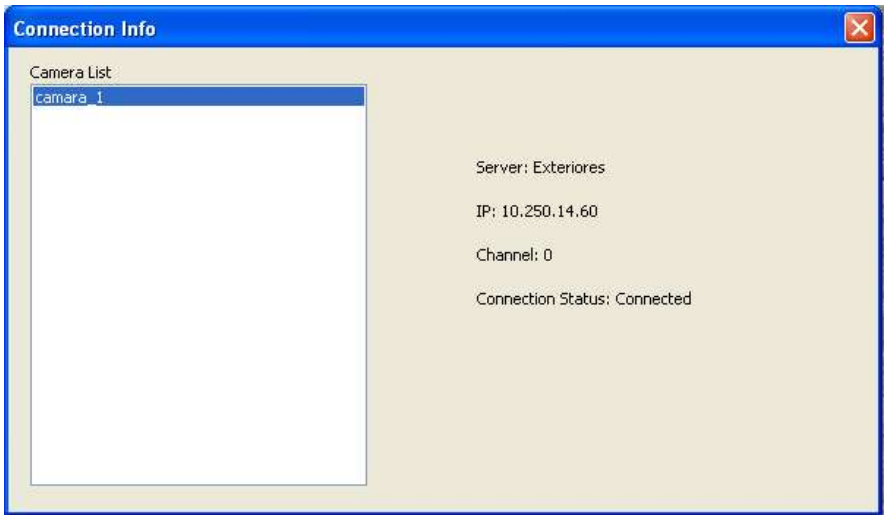

Ilustración 33: Interfaz de información de conexión de las cámaras

10.3.6 Control PTZ de cámara

Algunas cámaras de seguridad instaladas en el campus cuentan con controles PTZ. Mientras se realice una visualización de una cámara cualquiera, puede accederse a dichos controles desde el menú Plugins en Capaware.

- PTZOptionsController

  PTZOptionsController es el controlador de la clase PTZOptions.

- PTZOptions

  PTZOptions es el módulo que contiene la vista de los controles PTZ.

Las cámaras con tecnología PTZ cuentan con multitud de funcionalidades que las cámaras fijas tradicionales no otorgan. La API de videograbador proporciona acceso a muchas de ellas, por lo que resulta interesante proporcionar esta utilidad en el proyecto. En este PFC se ha optado por implementar las utilidades más comunes como son: el movimiento (Up, Left, Right, Down), hacer focus, hacer zoom, y manipular el iris de la cámara (ilustración 34).

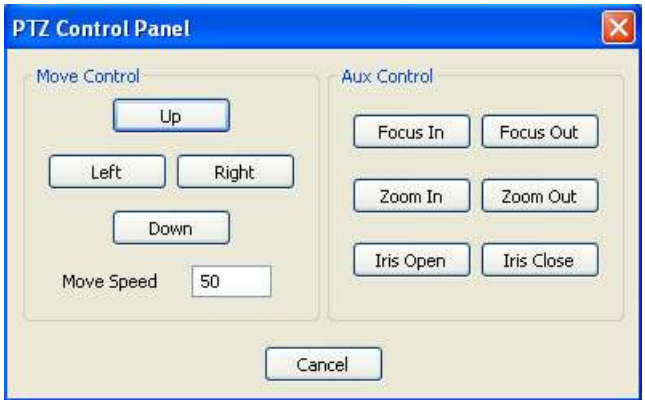

Ilustración 34: Vista con los controles PTZ implementados en el PFC

Para poder ejecutar estos comandos, es necesario la condición de haber seleccionado una cámara en la escena. Esta condición es consecuencia del cambio realizado en el diseño. En un principio, el acceso al control PTZ, se iba a poder hacer a través de la propia ventana de visualización de cámara. Sin embargo, en la implementación se decidió por ajustar el frame de video al tamaño de la ventana para minimizar el espacio que ocupa en pantalla, y por razones puramente visuales, se decidió removerlo de esta interfaz (no terminaba de convencer donde colocar el botón, su tamaño, etc.).

Por ello, se selecciona la cámara que se desea manipular en la escena, y se accede al panel de control a través del menú Plugins.

10.3.7 Visualizar vídeo

Los módulos de visualización engloban tanto el streamming de video en directo, como el modo de vídeo en histórico. Los módulos que se encargan de mostrar video son:

- VideoFrame

  VideoFrame es el controlador de la clase SimpleFrame.

- SimpleFrame

  SimpleFrame representa la vista de las imágenes de un único vídeo.

- MosaicFrame

  MosaicFrame representa la vista de un mosaico de imágenes de vídeo. Su controlador es la clase SetMosaicController (apartado 10.3.8).

Este apartado es uno de los más extensos de esta sección de la memoria, ya que engloba varios aspectos del desarrollo respecto a esta funcionalidad. En primer lugar, se hablará de la funcionalidad en sí, es decir, se describe en qué consiste mostrar video. En segundo lugar, se describirá como se lleva a cabo la gestión del tiempo (como establecer la fecha, controles de tiempo como play, stop, flashforward…). En tercer y último lugar, se presentará el pequeño estudio sobre la imagen resultante.

Para poder realizar una petición de video al servidor, es necesaria cumplir la precondición de haber establecido conexión con ella. La API del videograbador proporciona una estructura de datos donde se encuentra un búfer, el cual recoge el streamming de video.

En el caso de querer mostrar vídeo de una cámara en solitario (no en modo mosaico), estas funciones son gestionadas por la clase VideoFrame. Si se quiere mostrar un mosaico de cámaras, la clase SetMosaicController es la que lleva el control.

La API de videograbador exige crear unas funciones *callback*, tal y como sucedía con las funciones de conexión al servidor. Desde la función *callback*, donde nos llega el frame de video codificado en el formato YUV, la clase VideoFrame simplemente lee la información del búfer, aplica la transformación de YUV, vuelca la nueva imagen en un bitmap de wxWidgets y se muestra la vista implementada por SimpleFrame, mediante el evento OnPaint de la ventana (imagen 35).

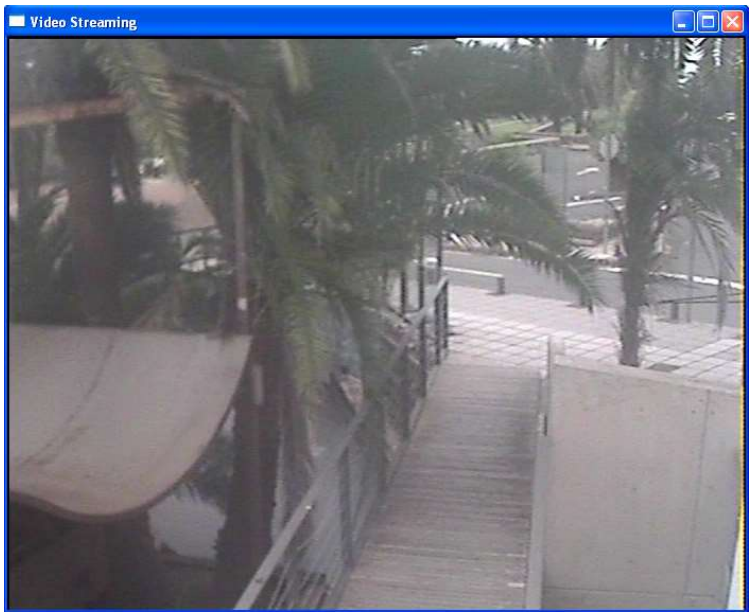

Ilustración 35: Vista de cámara única

El problema surge al intentar realizar la visualización de un mosaico de cámaras (imagen 36). Si intentamos realizar el pintado de las diferentes imágenes (no entre frames de la misma cámara, sino en frames de distintas cámaras) sobre la misma ventana, puede darse el caso de que al lanzar un refresh por el callback de una cámara, otra cámara interrumpa la operación realizando a su vez un onPaint.

En el caso específico del mosaico, cada cámara se pinta en un lugar diferente de la vista en MosaicFrame. Cada cámara tiene su propio frame, por lo que se evita el problema anterior. El búfer se copia en su correspondiente wxBitmap, ya que, según el tamaño del mosaico, se crea un vector de bitmaps de tamaño igual al tamaño de la malla. La alternativa a esto, era implementar un único bitmap muy grande, el cual englobe los diferentes frames de las diferentes cámaras.

Ilustración 36: Mosaico con las dos cámaras disponibles

Acceder a imágenes de video en modo histórico es un proceso similar al video en directo. CameraObjectController y VideoFrame implementan las peticiones y los *callback* de la API de videograbador. Es necesario aclarar el hecho de que, en el PFC, no se permite realizar un mosaico de históricos.

VideoFrame lleva el control de las funciones implementadas en el PFC para llevar a cabo el control de tiempo. En primera instancia, se intentó reutilizar los controles que proporciona Capaware para llevar a cabo animaciones de objetos en la escena. Finalmente, se descartó esta línea, ya que sin realizar nuevos añadidos en la interfaz de Capaware, con el objetivo de señalar de alguna forma por parte del usuario el modo que se quiere ejecutar, se solapaban los controles de cámara con los de animación con resultados no deseados.

Debido a la nueva situación, donde se podría utilizar parte de los controles (calendario y hora), pudiendo confundir al usuario con el resto de controles, se ha decidido por implementar un paso previo a la hora de invocar la petición de video en histórico. Antes de realizar dicha petición, se muestra un pequeño diálogo con un calendario y varios ComboBox, que permiten al usuario seleccionar una fecha y hora concreta del tiempo (ver Imagen 37).

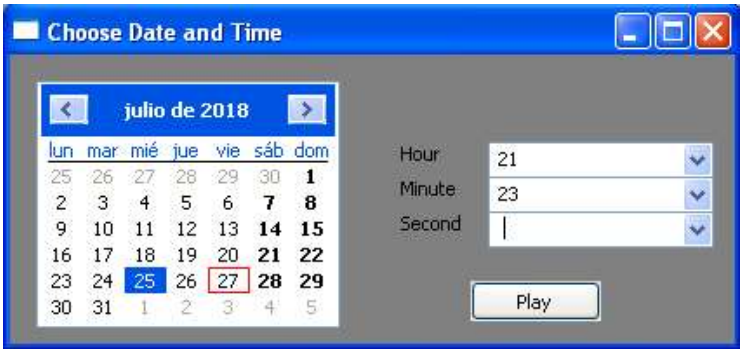

Ilustración 37: Ventana de selección de fecha y hora

Al hacer una petición de video en histórico, se lanza una ventana como la del caso anterior, con la diferencia de incorporar una pequeña botonera debajo de la imagen con los controles. Los controles implementados fueron: *previous*, *pause*, *stop* y *forward*. Los botones prevoius y forward funcionan también para controlar la velocidad de reproducción (implementados x1, x2 y x4). En la siguiente imagen puede verse un ejemplo de un video histórico con los controles:

Ilustración 38: Imagen de histórico con controles

Las imágenes de video resultantes de la transformación de formatos presentan mejor aspecto del esperado. La primera cámara, la cual estaba conectada al servidor videograbador disponible en el laboratorio, mostraba imágenes en blanco y negro. En principio, no se detectó mayores problemas con la transformación del formato YUV a RGB.

Sin embargo, esta situación cambió cuando se realizaron pruebas ya con el acceso a un servidor videograbador real en explotación. La calidad de la imagen (en color en este caso), puede verse

en la ilustración 39. Esta imagen es el resultado de aplicar la transformación del formato y, posteriormente, la función CLAMP, la cual simplemente poda los valores inferiores a 0 y mayores de 255 en cada canal.

Ilustración 39: Imagen resultante con CLAMP

A pesar de las premisas establecidas en el análisis, se puede decir que la imagen resultante presenta un aspecto bastante decente de lo esperado. Si bien el color luce algo extraño, debido a la distorsión de los rangos en algunos canales, es posible diferenciar la mayoría de los elementos presentados en la imagen.

Cabe recalcar, que algunas zonas de la imagen como son el cielo, o la parte izquierda y superior de la pantalla (representa el fondo del escenario), apenas puede distinguirse nada en ellas. Otros elementos como las plantas, que resulta difícil diferenciarlas entre sí, o el suelo del pequeño "mirador" de la imagen, donde no se distinguen las sombras del techo (piezas de madera con espacios entre ellas).

Aunque se pueda dar por válida la imagen obtenida en la anterior imagen, es interesante analizar si se puede mejorar la calidad de ella mediante una normalización de los valores RGB. En primer lugar, es necesario ver que valores de cada canal se están obteniendo en el *frame* que se va a procesar. A continuación, se puede ver dichos valores en la tabla 6:

	Mínimos	Máximos
Valor de canal R	-14	272
Valor de canal G	-13	276
Valor de canal B	-21	326

Tabla 6: Valores de canales RGB para normalización

Tal y como puede observarse en la tabla 6, los valores mínimos y máximos de los tres canales superan el rango de valores válido por el formato RGB24. En particular, podemos ver que el canal azul es el que posee el mayor valor negativo, así como el valor máximo más alto. Siendo uno de los canales algo dispar en comparación a los otros, puede provocar una distorsión del color, como puede verse en la imagen 40.

Ilustración 40: Imagen resultante normalizada

Se observa que, en efecto, el color ha sufrido un cambio considerable. Los canales rojo y verde están ahora mucho más saturados que el canal azul, provocando una tonalidad “amarillenta” en la imagen, invalidando completamente el resultado.

Si bien la imagen resultante es totalmente inválida en este aspecto, al menos se observa que los elementos del fondo de la escena son mucho más distinguibles que en la imagen 21. Se puede apreciar mejor el cielo, y diferenciarlo de la montaña y algunos edificios que se encuentran al fondo de la imagen. Las plantas situadas por detrás de las pistas, se diferencian algo mejor entre sí.

Como último paso, queda por ver el resultado de aplicar una transformación al espacio de color HSV y reducir la saturación, con el objetivo de hacer más diferenciables los distintos elementos de la imagen. En la ilustración 41, se observa el resultado de aplicar una reducción de la saturación (S) del 20%.

Ilustración 41: Imagen resultante con saturación al 80%

En esta imagen, se consigue diferenciar mejor los elementos presentes en la misma, sin producir la distorsión del color que se obtenía mediante la normalización de los canales. Elementos como el cielo, el cual se distingue significativamente mejor de la montaña, a pesar del brillo (la imagen fue tomada en el mes de Julio sobre el mediodía). Otros como la vegetación por detrás de las pistas, la cual es ligeramente más clara y diferenciable entre sí; y en el suelo del pequeño mirador, se puede llegar a ver las sombras del techo de madera.

En resumen, se puede afirmar que el mejor resultado se obtiene aplicando una pequeña reducción en la saturación (sacrificando algo la pureza del color). Sin embargo, realizar múltiples transformaciones entre formatos y espacios de color podría llegar a afectar al rendimiento del muestreo de video. En este PFC, no se ha detectado problemas con ello, teniendo en cuenta que el tamaño de la muestra de la prueba es muy reducido.

10.3.8 Configurar mosaico de cámaras

La configuración de un mosaico se realiza mediante una vista con una lista de cámaras, donde se van seleccionando aquellas que se quieran incluir en el mosaico.

- SetMosaicController

  SetMosaicController es el controlador de la clase SetMosaic.

- SetMosaic

  SetMosaic es el módulo que representa la vista de las opciones de configuración de un mosaico. Además de la clase con mismo nombre, contiene las clases MosaicCameraList y MosaicGridSize, que implementan las pestañas de la vista en SetMosaic.

Configurar un mosaico requiere de realizar un proceso de selección de cámaras, además de selección de tamaño de mosaico. La clase SetMosaicController realiza el control de las listas localizadas en MosaicCameraList. En la primera de ellas, se muestran todas las cámaras disponibles en la escena, similar a lo que ocurría en el proceso de adición de cámara a escena. La segunda lista, irá mostrando aquellas cámaras que seleccionemos en la primera y pulsemos en el botón Add (imagen 42).

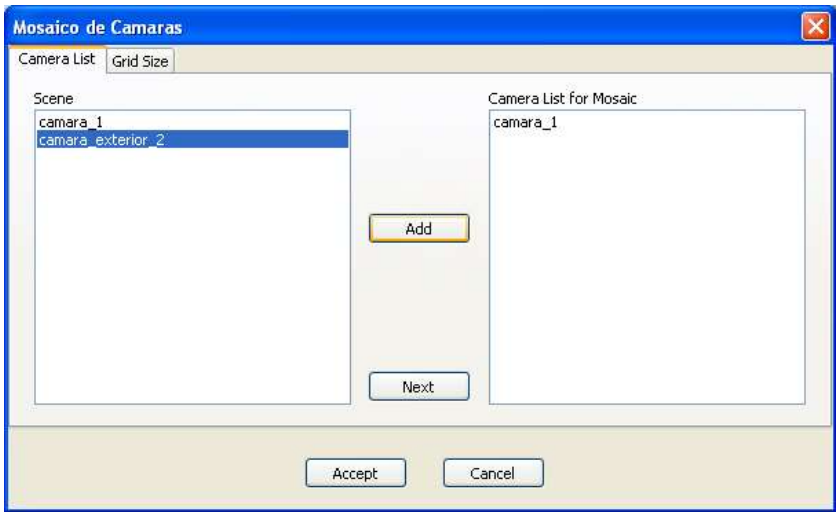

Ilustración 42: Configurar mosaico de cámaras, pestaña lista de cámaras

Realizar un mosaico de cámaras conectadas a diferentes servidores es, en principio, posible. Sin embargo, al solo disponer de acceso a un único servidor, no se ha podido realizar depuraciones en este escenario. La estructura de datos de la API permite saber de qué servidor y de qué canal procede la información contenida en el búfer.

En la pestaña *Grid Size*, el usuario puede elegir el tamaño de la malla a visualizar. Existen tres opciones implementadas: cuatro, seis y nueve. Si se seleccionan más cámaras de las que se pueden visualizar, saldrá un aviso advirtiendo que sólo se mostrarán las primeras de la lista (por ejemplo, si se seleccionan cinco cámaras, pero se elige una malla de tamaño cuatro, solo se verán las cuatro primeras cámaras de la lista *Camera List for Mosaic*.

La clase SetMosaicController implementa las funciones necesarias para iniciar la petición de imágenes de las cámaras al servidor. Los detalles de visualización al respecto, se encuentran en el apartado anterior.

10.3.9 Crear hilo de conexión y clases auxiliares

A diferencia del resto de módulos del proyecto, la conexión se gestiona mediante la inclusión del fichero Windows.h y haciendo peticiones al servicio operativo mediante su interfaz.

- DXConnectThread

  DXConnectThread contiene un controlador que proporciona hilos de conexión para establecer conexiones con el servidor videograbador.

- AuxClasses

  AuxClasses es un módulo que contiene la clase AuxMessage, la cual muestra pequeñas ventanas de avisos o errores a la hora de realizar acciones no permitidas por el plugin.

DXConnectThread maneja internamente las peticiones al SO Windows usando tipos de datos del propio Windows. Esto es necesario ya que la API de videograbador maneja un parámetro del tipo HSERVER, que es un handler de Windows.

En AuxClasses se llevó a cabo la implementación del calendario para elegir la fecha concreta del modo histórico. En principio, solo iba a consistir en un pequeño panel que se soportaba en VideoFrame.

10.3.10 Plugin.cpp

El fichero plugin.cpp es el módulo principal del PFC. Contiene algunas clases como PluginResources, la cual proporciona acceso a recursos de la aplicación Capaware (acceso a árbol de escena, entidades, base de datos, etc.), o PluginInfoPanel, el cual solo muestra un mensaje sobre el proyecto.

Incorpora también los métodos necesarios para la implementación de un plugin en Capaware, que son: RegisterMenu, EntitiesToRegister, InitPlugin, ExecPlugin y GetFunctions. En el manual de Capaware puede encontrarse una pequeña guía de plugins, además de un ejemplo sencillo.

Dentro de plugin.cpp, vienen definidas todas las funciones adicionales, es decir, todas las nuevas opciones de interacción con el usuario. Por ejemplo, los procesos de visualizar y ocultar todas las cámaras, están definidas en este módulo, pero es CameraObjectController quien lleva a cabo este proceso. En este caso, se hace un recorrido del árbol de la escena, y siempre que se encuentre una cámara, pondrá la propiedad de *visualize* a su valor correspondiente (*true* si se quieren ver o *false* si se quieren ocultar).

11. Pruebas

Con el fin de llevar a cabo una verificación adecuada del sistema, se han realizado pruebas de caja negra a cada funcionalidad implementada en el PFC. Aquellas funcionalidades que dependen del servidor videograbador se hicieron en dos momentos del tiempo diferentes.

Al comienzo del PFC se contaba con un servidor videograbador en el laboratorio de sistemas 1.2 y una cámara de vídeo de imágenes en blanco y negro. Gracias a esto, se pudo llevar a cabo una primera fase de pruebas en las que se realizaban conexiones al servidor y se accedía a las imágenes de video.

Con el tiempo, este servidor ya no se encontraba en el laboratorio por necesidades de la institución. Debido a esto, se realizó una petición al responsable de videovigilancia del campus de la ULPGC, con el objetivo de poder realizar una prueba final del PFC con un sistema de videovigilancia real.

El personal de seguridad proporcionó el acceso a un servidor videograbador DX8100 concreto con acceso a dos cámaras exteriores. Además, se otorgó un par usuario/contraseña para poder realizar la autentificación y acceder a funcionalidades restringidas como las imágenes de video en histórico. Mientras tanto, el personal de sistemas del edificio de Informática configuró uno de los accesos del laboratorio de sistemas 1.2 para conectar el equipo a la red de seguridad del campus.

A pesar del resultado satisfactorio por parte de las pruebas con este servidor, algunas funcionalidades (principalmente los controles PTZ de cámara) no pudieron llevarse a cabo. Las pruebas respecto a la funcionalidad de mosaico no pueden considerarse robustas, ya que sólo se dispuso de acceso a dos cámaras.

Respecto a la base de datos, al tratarse de una base de datos local en Capaware mediante MySQL, se llevó a cabo la instalación en el equipo del laboratorio y se realizaron sencillas pruebas con las tablas adicionales.

12. Conclusiones y trabajo futuro

12.1 Conclusiones

En líneas generales, considero que el proyecto presenta una línea muy interesante en cuanto a la integración del sistema de videovigilancia por CCTV en un entorno virtual 3D. La posibilidad real de utilizar una representación de la zona controlada facilita a un usuario final una tarea tan sencilla como localizar donde se encuentra cada cámara.

Valorar el resultado final del plugin puede llegar a ser complejo debido a los diferentes elementos a tener en cuenta. A pesar de ello, podemos centrarnos en tres aspectos fundamentales: rendimiento de la aplicación con el plugin, accesibilidad del producto final y la calidad de la imagen final.

Con respecto al rendimiento de la aplicación, se puede afirmar que el plugin funciona de una manera fluida con la muestra disponible. A pesar de que en este proyecto se han manejado resoluciones algo pobres y tasas de imágenes por segundo algo bajas, la aplicación no presenta los clásicos “parones” o ralentizaciones debidas a la combinación de un entorno 3D con la reproducción de video. En un principio sentía preocupación por esos posibles cuelgues o ralentizaciones del equipo al combinar ambas tecnologías.

La accesibilidad del producto final es un aspecto muy importante, ya que la idea del PFC era intentar llevar a cabo una integración del sistema de videovigilancia por CCTV en un entorno 3D y poder observar si las barreras de entrada en las labores de videovigilancia se podían evitar. En principio, disponer de un entorno virtualizado georreferenciado de la zona donde hay localizadas cámaras, ayuda visualmente a ser capaz de crear una imagen o mapa mental de donde están localizadas las cámaras. Con el sistema clásico, el etiquetado (que sigue teniendo un peso muy importante en la jerarquía de la escena) era la única herramienta que nos permitía localizar una cámara.

Si bien en algunos aspectos, el proyecto mejora la experiencia, también es cierto que surgen otros inconvenientes, como es el aprendizaje que conlleva tanto Capaware, como el plugin desarrollado. Insertar una cámara en la escena puede parecer un proceso sencillo para un alumno de Ingeniería Informática o personas familiarizadas con entornos 3D. Sin embargo, la posibilidad de insertar elementos 3D mediante *drag&drop* en la escena, simplificaría mucho el proceso. Llevar a cabo tal implementación en Capaware implicaría realizar modificaciones no solo en los módulos internos de Capaware (quebrantando uno de los principios básicos de una extensión de programa), sino modificaciones en el diseño original de la aplicación, por lo que se hace inviable esta propuesta.

Por otro lado, se ha intentado evitar la inundación de vistas o interfaces que puedan empobrecer la experiencia de usuario. Cada vista contiene los mínimos elementos necesarios para llevar a cabo su función. Siempre que se ha podido, se ha intentado aprovechar todos los recursos que nos aporta la API de Capaware CPW. Recursos como el árbol de la escena o acceder a la persistencia del sistema, han sido indispensables para el desarrollo del proyecto.

Respecto a la calidad de la imagen final, a pesar de las advertencias en el análisis respecto a las transformaciones del formato YUV, he podido observar que la calidad de la imagen es bastante

buena pese a la baja resolución de imagen. Aun así, estoy contento de haber podido dedicar algo de tiempo a implementar varios métodos sencillos con el objetivo de intentar mejorar en parte la calidad de la imagen, o hacer reconocibles algunos elementos de la misma que no podían verse con claridad.

Como conclusión final, me siento satisfecho por el desarrollo del PFC, el cual me ha permitido aprender cómo integrar librerías, combinar diferentes APIs, adaptarme a un programa ya creado y utilizar sus recursos (en ocasiones más complejo que partir de cero), entre otros. En principio, me habría gustado poder acceder a muchas más cámaras, en especial cámaras singulares que disponen de tecnología de visión nocturna o similares, con el objetivo de hacer un estudio mucho más profundo en la gestión de un gran número de elementos y visualizaciones. Sin embargo, las fuertes leyes de protección de datos en la videovigilancia restringen en gran medida la posibilidad de acceder a imágenes de video sensible, como pueden ser zonas de mucho tránsito de personas.

12.2. Trabajo futuro

12.2.1 Compatibilidad con otros servidores videograbadores

En el desarrollo del proyecto solo se ha trabajado con el servidor videograbador DX8100, debido a que se disponía de uno en el laboratorio de sistemas. Sin embargo, el campus de la ULPGC cuenta con varios sistemas diferentes (todos de la serie DX) como puede ser el DX4500/4600.

Por otro lado, también se contempla ir adquiriendo en el futuro servidores videograbadores que sean compatibles con una resolución de alta definición. Por ello, es necesario en el futuro combinar las APIs de las distintas versiones, además de contemplar posibles servidores que no pertenezcan a la serie DX (diferente arquitectura, APIs para otros lenguajes de programación como Python…).

Los servidores de alta definición manejan resoluciones grandes, por lo que el rendimiento de la aplicación alcanzará una importancia fundamental. En el caso de Capaware, el cual es una aplicación algo pesada en sí, soportar el motor 3D junto a resoluciones de imágenes de 1080p, con una tasa de imágenes por segundo de 60 fps, es probable que ralentice el equipo significativamente. Estudiar la posible migración de la aplicación de gestión a otros globos virtuales que sean más ligeros, puede ser una opción ardua pero interesante.

12.2.2 Migración a un globo virtual de uso más extendido

A pesar de que Capaware es una aplicación más que capaz de integrar la labor de videovigilancia, sigue siendo una aplicación algo pesada y que requiere de aprendizaje por la parte de los usuarios. En este proyecto se han realizado pruebas con cámaras con una resolución algo paupérrima y tasas de imágenes por segundo muy bajas para los estándares actuales. Resoluciones de alta defincón de 1080p con una tasa de frames de 60 fps, puede ocasionar problemas de rendimiento con Capaware.

Llevar a cabo una migración del proyecto a otros globos actuales más ligeros, puede ayudar a evitar el problema de rendimiento. A pesar de que los equipos de videovigilancia suelen disponer de procesadores y gráficas bastante potentes (debido al manejo continuo de múltiples imágenes de video), combinar entornos 3D con dichas imágenes es algo que debe ser estudiado

en detalle. Con respecto al aprendizaje, otros globos virtuales mucho más conocidos popularmente como puede ser Google Earth, facilitarían esta etapa.

Sin embargo, debe tenerse en cuenta las implicaciones negativas que puede tener llevar a cabo la migración a otro entorno. Si bien algunas funcionalidades pueden beneficiarse de ello, otras podrían recibir un impacto negativo.

12.2.3 Adaptación de las funcionalidades al nuevo entorno

Si se llevase a cabo la migración a otro globo virtual o se lleva a cabo alguna otra solución similar como se describe en el apartado anterior, es posible que algunas de las funcionalidades implementadas por el PFC puedan modificarse y adaptarlas a las posibilidades que ofrezcan las APIs ofertadas por el nuevo entorno.

Si, por ejemplo, nos centrásemos en la adición de cámaras a escena, se podría plantear un escenario donde se implemente una colección de elementos 3D favoritos o de distintos tipos de cámara, y con ello, añadirlas mediante un drag & drop sencillo. Con una funcionalidad por el estilo, facilitaríamos una de las características fundamentales del proyecto. Evidentemente, esto no exime al usuario de rellenar algún tipo de formulario con la información de la cámara (a qué servidor pertenece, canal, etiqueta…).

Por otro lado, poder operar en el nuevo entorno con nuevos modos de visualización, permitiría expandir las funcionalidades actualmente desarrolladas. Poder cambiar la visualización de una ortofoto, por la de un mapa, nos ofrece la posibilidad de integrar otros sistemas de videovigilancia.

12.2.4 Integración de otros sistemas de vigilancia

La labor de videovigilancia no solo consiste en utilizar un CCTV. Otros sistemas como el control de acceso a los edificios del campus, el sistema de detección de incendios, entre otros también son utilizados por el personal de seguridad.

La idea de integrar el sistema de detección de incendios a la gestión de cámaras resulta bastante interesante. Si, por ejemplo, se presenta un escenario donde se ocasiona un incendio en algún lugar del campus, el sistema de gestión podría recoger también la señal de la alarma de incendios y presentar al personal de seguridad imágenes de video de la zona afectada.

Esta idea permitiría al personal de seguridad ver en directo la situación de la zona del incendio. Funciones vitales como la evacuación, posible intervención mediante extintores u otras acciones pueden ser inspeccionadas con una inmediatez que de otra forma no se lograría. Otra de las características del sistema, permitiría realizar grabaciones de esas imágenes en los equipos locales (además de los servidores) y así facilitarlas a las autoridades competentes.

12.2.5 Implementación de algoritmo de agrupación

Uno de los principales inconvenientes de realizar pruebas con muestras pequeñas, suele ocasionar que algunos problemas no se detecten. En el caso de este proyecto, solo se ha tenido acceso a un servidor y una cámara en su primera fase, y a un servidor y dos cámaras en la segunda fase (tal y como se describió en la sección 11 Pruebas).

En un ensayo real, un edificio como puede ser el de informática, tendrá varias cámaras por planta en los distintos módulos que compone el edificio. Dependiendo del nivel de detalle máximo al que podamos acercarnos con la cámara, esta masificación de elementos 3D en la escena puede ocasionar dificultades para seleccionar una cámara en concreto, teniendo que

recurrir forzosamente a la jerarquía de cámaras. Por no mencionar que el resultado no será beneficioso, quedando la escena caótica y nada vistosa para el usuario.

Implementar un algoritmo que realice agrupaciones de cámaras, ocultando la mayoría de ellas y dejando visibles algunas que "representen" al conjunto según el nivel de detalle de la escena cargado o el valor zfar de la cámara, podría solventar alguno de estos problemas.

Esto a su vez, genera otra serie de problemas, ya que el algoritmo debe tomar decisiones según los criterios que se definan. Estos criterios bien pueden ser elegidos por el usuario mediante alguna interfaz extra implementada, o simplemente ser programado en el código en base a un análisis y entrevistas con el personal de seguridad.

12.2.6 Combinación con sistemas de reconocimiento facial

En el apartado 13.3 se habla de integrar los otros sistemas de los que dispone el personal de seguridad para la labor de videovigilancia. Si se llegase a integrar el sistema de acceso, por ejemplo, mandando un aviso al sistema de gestión de varios intentos de acceso fallido en algún lugar del campus, permitiendo al usuario visualizar la cámara que está orientada a ese punto, podría extenderse este sistema combinándose con un sistema automático de reconocimiento facial.

La respuesta a este tipo de situaciones nunca es inmediata, por lo que la aplicación de reconocimiento no tiene que dar una respuesta inmediata. Mientras el personal comprueba la veracidad de la situación de esta persona, la aplicación de reconocimiento podría trabajar en paralelo tratando de identificar a dicha persona y comprobar si tiene algún vínculo con la organización (personal, estudiante que tenga autorización o no, etc.).

Además, no solo puede ceñirse al ámbito del control de acceso. El sistema de reconocimiento podrá trabajar con frames concretos mientras constantemente se visualizan imágenes de video, o al menos, cuando se muestre vídeo de zonas con tasas de incidencias superiores a la normalidad.

13. Bibliografía

[1] Álvarez, R. (2018). El reconocimiento facial se sigue expandiendo en China. Fuente: https://www.xataka.com/privacidad/reconocimiento-facial-se-sigue-expandiendo-china-metro-beijing-shanghai-incorporan-sistemas-rastreo-biometrico

[2] Capaware rc2. http://www.capaware.org/

[3] Licencia de software libre GPL bajo GNU. https://www.gnu.org/licenses/licenses.en.html

[4] Pelco Security Cameras and Surveillance Systems. https://www.pelco.com/

[5] VideoXpert Profesional de Pelco. https://www.pelco.com/vxpro

[6] Open Geospatial Consortium. http://www.opengeospatial.org/

[7] Instituto Tecnológico de Canarias. http://web.itccanarias.org/

[8] Open Scene Graph. http://www.openscenegraph.org/

[9] wxWidgets. Cross-Platform GUI Library. https://www.wxwidgets.org/

[10] Santana, J.M. (2017). New methodologies and techniques on the development of virtual globes for mobile devices and the web.

[11] Google Earth. https://www.google.com/intl/es/earth/

[12] Google. https://www.google.es/

[13] NASA World Wind. https://worldwind.arc.nasa.gov/

[14] NASA. https://www.nasa.gov/

[15] United States Geological Survey. https://www.usgs.gov/

[16] Cesium - Next Dimension 3D Geospatial Mapping and Analysis. https://cesium.com/

[17] Marble. https://marble.kde.org/

[18] OpenStreetMap. https://www.openstreetmap.org/

[19] Blue Marble Image Repository. https://visibleearth.nasa.gov/view_cat.php?categoryID=1484

[20] Glob3 Mobile. http://glob3mobile.com/

[21] Unified Modeling Language. https://www.uml.org/

[22] Entorno de desarrollo Microsoft Virtual Studio Express. https://visualstudio.microsoft.com/es/vs/express/

[23] Visual Paradigm. https://www.visual-paradigm.com/

[24] Balsamiq Wireframes. https://balsamiq.com/wireframes/

Bibliografía adicional consultada durante el desarrollo:

- http://www.cplusplus.com/reference/
- https://es.cppreference.com/w/
- https://www.wxwidgets.org/docs/
- https://docs.microsoft.com/en-us/windows/desktop/winprog/windows-data-types
- https://stackoverflow.com/
- Manual de Referencia de Capaware rc2
- Material del curso de Capaware impartido en la ULPGC

14. Anexos

El apartado de anexos muestra aquellos listados de esquemas o diseños, los cuales ocupen demasiado espacio en otras secciones, además de información adicional de la instalación del PFC, así como una guía de uso de cada una de las funcionalidades del Plugin dentro de Capaware.

Al tratarse de una extensión de una aplicación ya existente, se añadirá una pequeña guía de instalación de Capaware mediante el uso de los archivos binarios.

14.1 Diagramas de casos de uso

Se dispone ahora a mostrar los diagramas de casos de uso. Aquellos casos que comparten rol o actor, no se pondrán por duplicado, sino que se mostrarán en un solo diagrama. En caso de existir alguna diferencia, se indicará en dicho diagrama.

14.1.1 Iniciar sesión

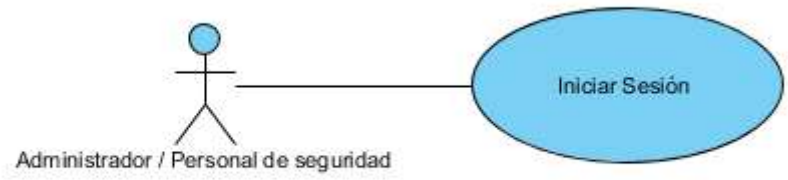

Ilustración 43: Diagrama de caso de uso de iniciar sesión

Iniciar sesión	
Descripción	Proceso mediante el cual un usuario autorizado del sistema introduce sus credenciales y ser reconocido por el sistema.
Precondiciones	El usuario debe poseer una cuenta autorizada para validar la autentificación.
Parámetros	Credenciales del usuario (usuario y contraseña).
Comportamiento	Si usuario y contraseña son válidos, se autentifica al usuario en el sistema.
Postcondiciones	Usuario identificado por el sistema.

Tabla 7: Definición de caso de uso de iniciar sesión

14.1.2 Añadir servidor

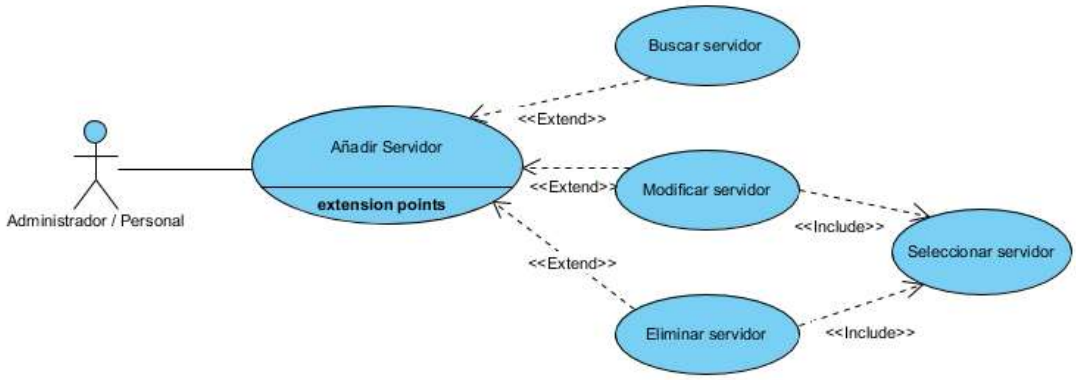

Ilustración 44: Diagrama de caso de uso de añadir servidor

Añadir servidor	
Descripción	Proceso mediante el cual un usuario puede añadir un nuevo servidor videograbador en el sistema.
Precondiciones	Usuario identificado en el sistema.
Parámetros	Nombre (o etiqueta) que identifica al videograbador y dirección IP.
Comportamiento	Si el par nombre/IP no coincide con ninguno previamente añadido, se añade en el sistema.
Postcondiciones	Servidor añadido.

Tabla 8: Definición de caso de uso de añadir servidor

14.1.3 Añadir carpeta contenedora

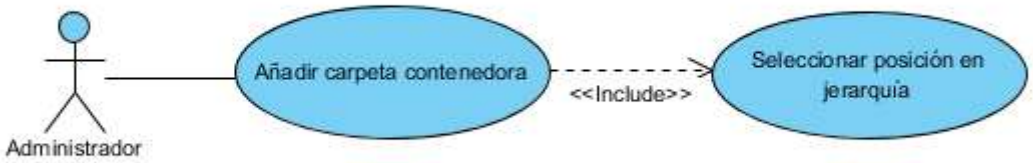

Ilustración 45: Diagrama de caso de uso de añadir carpeta contenedora

Añadir carpeta contenedora	
Descripción	Proceso mediante el cual el usuario crea una carpeta en el árbol de la escena y, con ello, establecer una jerarquía.
Precondiciones	Haber elegido previamente otra carpeta contenedora o la raíz del árbol que hará de padre de la misma.
Parámetros	Nombre (o etiqueta).
Comportamiento	El usuario elegirá una posición en el árbol de la escena, y procederá a añadir la carpeta, insertándose como hijo de la posición seleccionada.
Postcondiciones	Carpeta contenedora añadida.

Tabla 9: Definición de caso de uso de añadir carpeta contenedora

14.1.4 Mostrar y ocultar todas las cámaras

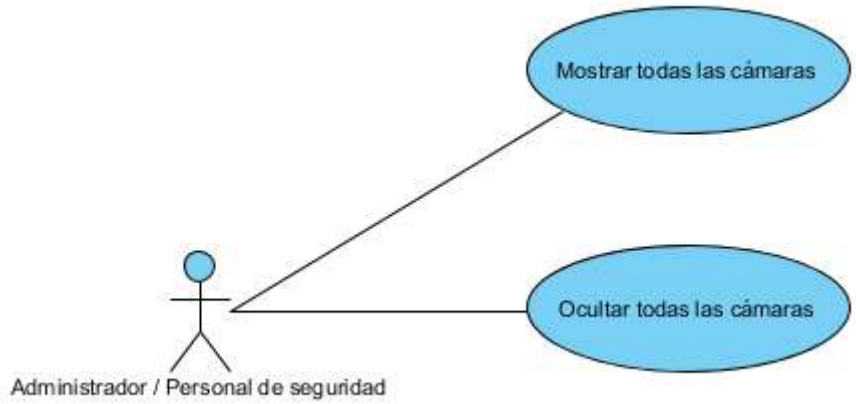

Ilustración 46: Diagrama de caso de uso de mostrar y ocultar todas las cámaras

Mostrar todas las cámaras	
Descripción	Proceso mediante el cual el usuario muestra todas las cámaras que estén presentes en la escena.
Precondiciones	
Parámetros	Escena (o lista de elementos de la escena).
Comportamiento	Todas las cámaras que se encontraban ocultas, vuelven a mostrarse en la escena. Si ya se encontraban en ese estado, el proceso no hace nada.
Postcondiciones	Cámaras visibles.

Tabla 10: Definición de caso de uso de mostrar todas las cámaras

Ocultar todas las cámaras	
Descripción	Proceso mediante el cual el usuario oculta todas las cámaras que estén presentes en la escena.
Precondiciones	
Parámetros	Escena (o lista de elementos de la escena).
Comportamiento	Todas las cámaras visibles se ocultan en la escena. Si ya se encontraban en ese estado, el proceso no hace nada.
Postcondiciones	Cámaras ocultas.

Tabla 11: Definición de caso de uso de ocultar todas las cámaras

14.1.5 Desconectar de servidor

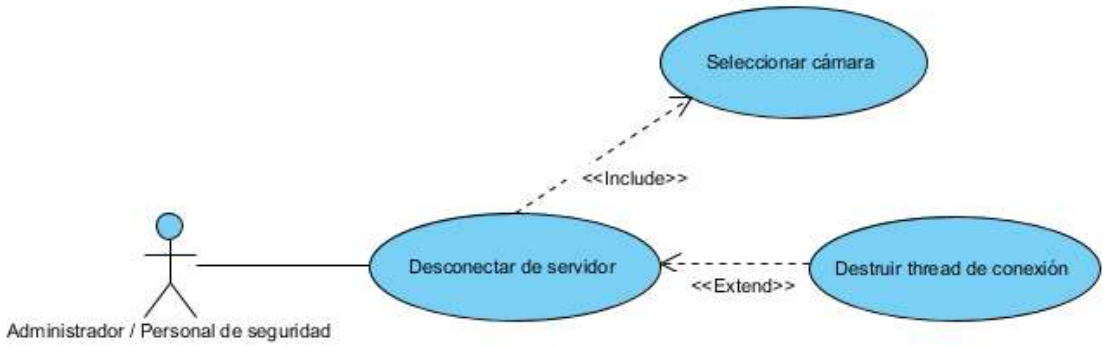

Ilustración 47: Diagrama de caso de uso de desconectar de servidor

Desconectar de servidor	
Descripción	Proceso mediante el cual el usuario se desconecta del servidor videograbador.
Precondiciones	Usuario identificado en el sistema. Cámara añadida.
Parámetros	Manejador de servidor.
Comportamiento	El usuario selecciona una cámara y se desconecta del servidor.
Postcondiciones	Cámara desconectada.

Tabla 12: Definición de caso de uso de desconectar de servidor

14.1.6 Mostrar información de conexiones

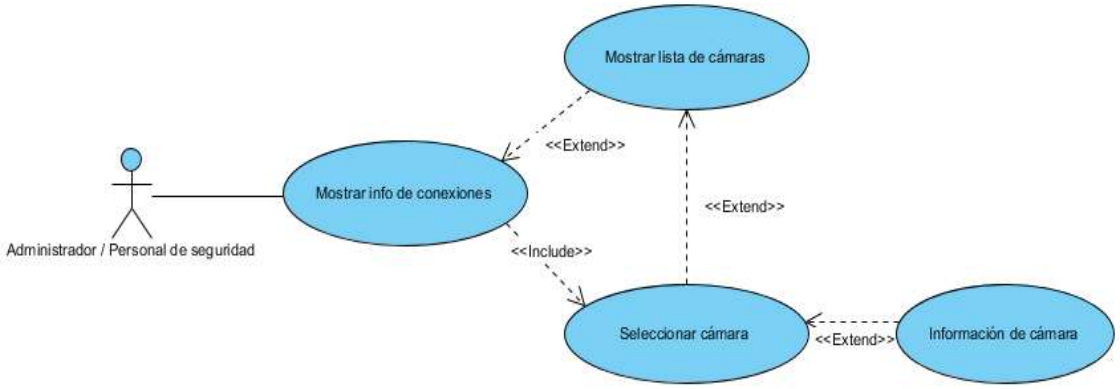

Ilustración 48: Diagrama de caso de uso de mostrar información de conexiones

Mostrar información de conexiones	
Descripción	Proceso mediante el cual el usuario puede acceder a una lista de cámaras y ver el estado de conexión de cada una de ellas, incluyendo a qué servidor están conectadas.
Precondiciones	Usuario identificado en el sistema.
Parámetros	Escena (o lista de elementos de la escena).
Comportamiento	Se muestra una lista de cámaras y, al seleccionar en una de ellas, se muestra información sobre el estado de conexión, servidor IP al que está conectada y canal seleccionado
Postcondiciones	

Tabla 13: Definición de caso de uso de mostrar información de conexiones

14.1.7 Mostrar video en histórico

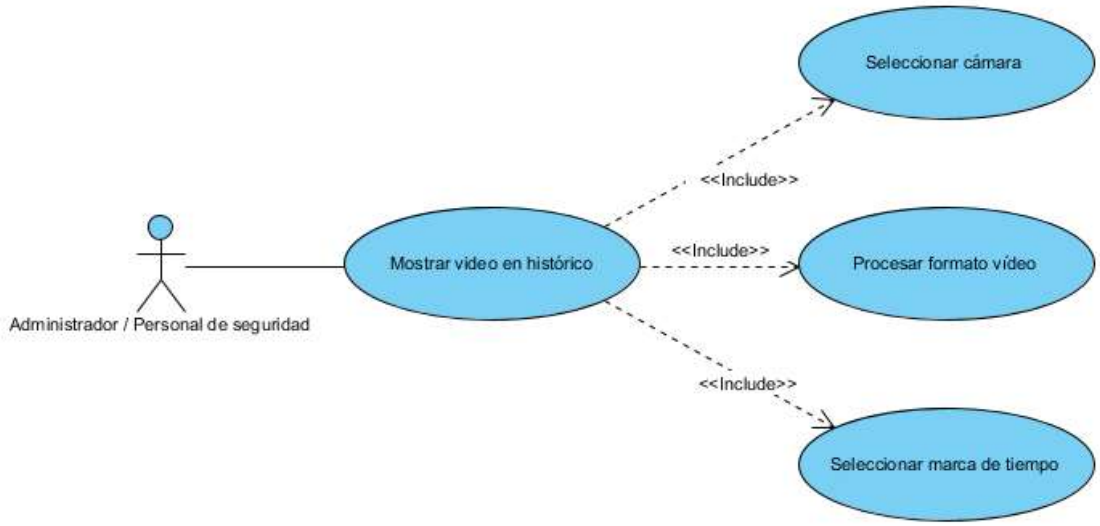

Ilustración 49: Diagrama de caso de uso de mostrar video en histórico

Mostrar video en histórico	
Descripción	Proceso mediante el cual el usuario accede al histórico de una cámara seleccionada previamente en un momento concreto del tiempo.
Precondiciones	Usuario identificado en el sistema. Cámara añadida.
Parámetros	Manejador del servidor. Canal. Formato de video. Marca de tiempo.
Comportamiento	El sistema muestra en una ventana el histórico de una cámara previamente seleccionada por el usuario en un momento concreto del tiempo.
Postcondiciones	Video en histórico mostrándose.

Tabla 14: Definición de caso de uso de mostrar video en histórico

14.1.8 Mosaico de cámaras

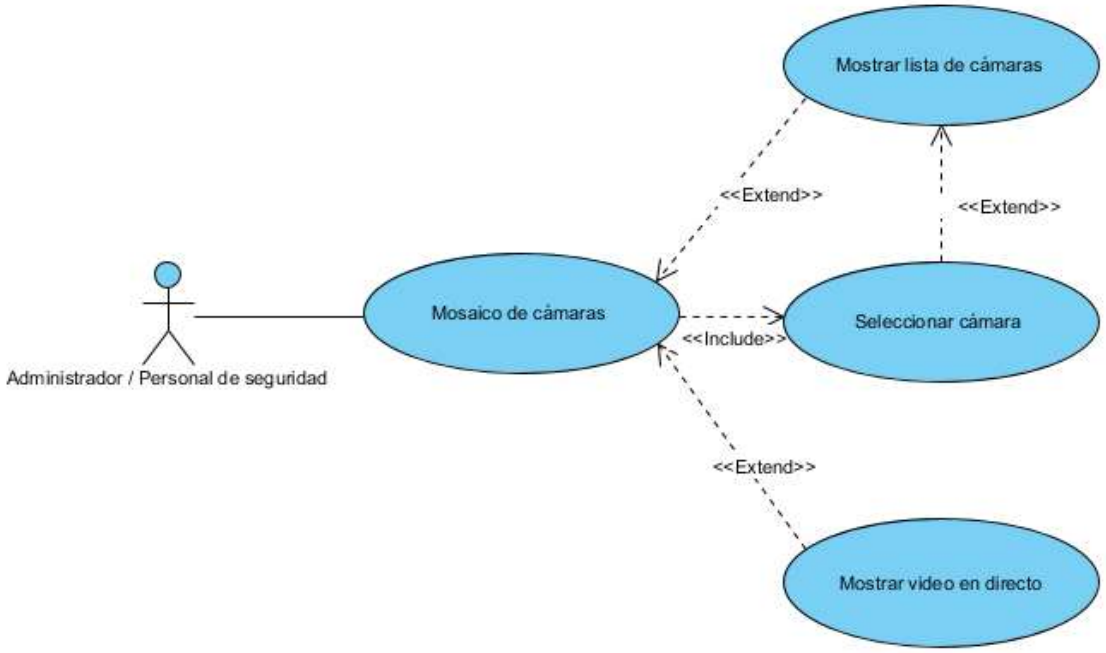

Ilustración 50: Diagrama de caso de uso de mosaico de cámaras

Mosaico de cámaras	
Descripción	Proceso mediante el cual el usuario muestra un mosaico de cámaras seleccionadas.
Precondiciones	Usuario identificado en el sistema. Cámara añadida.
Parámetros	Manejador del servidor. Canales. Formato de video. Distribución del mosaico.
Comportamiento	El sistema muestra un formulario con la lista de cámaras. El usuario elegirá que cámaras y en que distribución se mostrarán.
Postcondiciones	Mosaico de cámaras mostrándose.

Tabla 15: Definición de caso de uso de mosaico de cámaras

14.2 Mockups de Vistas

A continuación, se muestran todos los Mockups de las vistas correspondientes a la sección 9.6 Diseño de Vistas.

14.2.1 Menú Ver

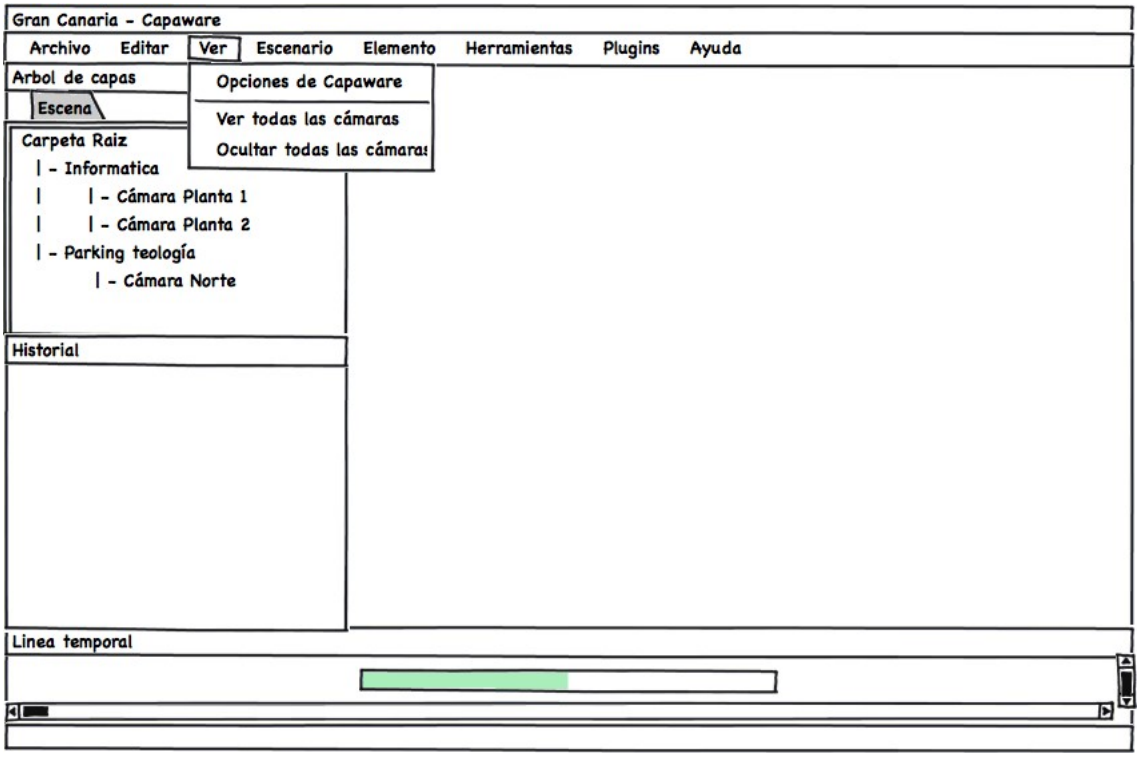

Ilustración 51: Mockup de vista general de Capaware. Menú Ver.

14.2.2 Clic Secundario Cámara

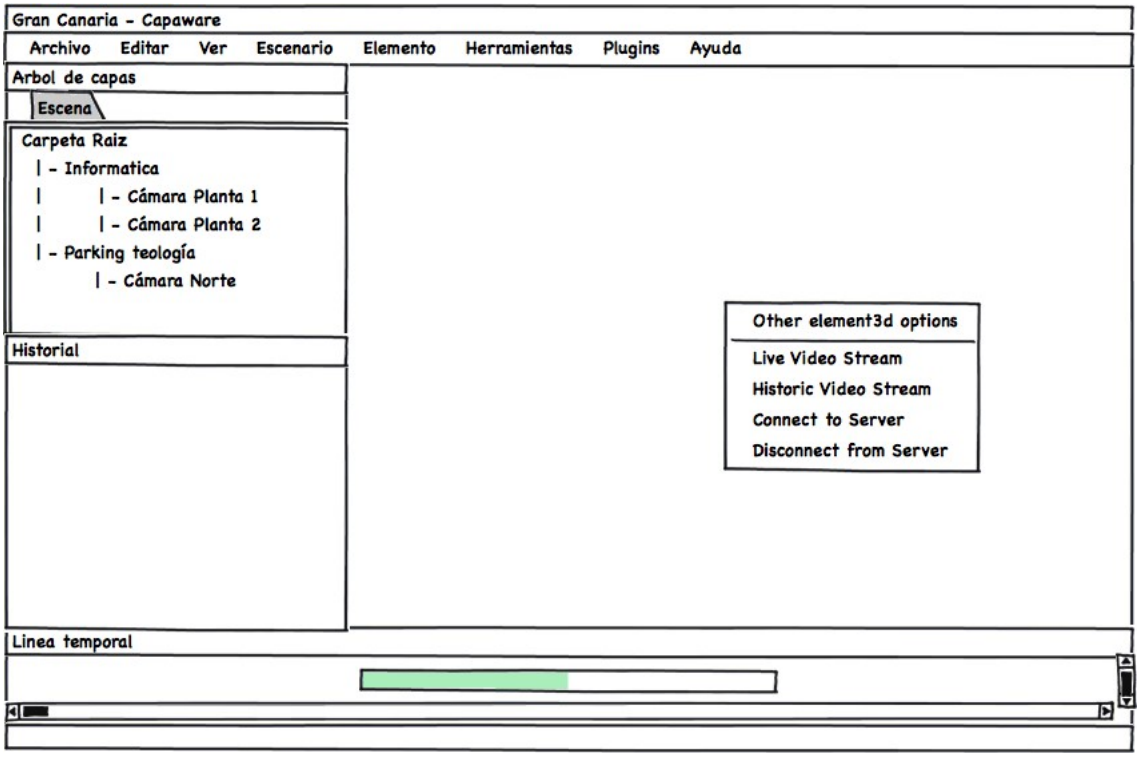

Ilustración 52: Mockup de vista general de Capaware. Menú Ver.

14.2.3 Añadir Cámara - Selección de área/edificio

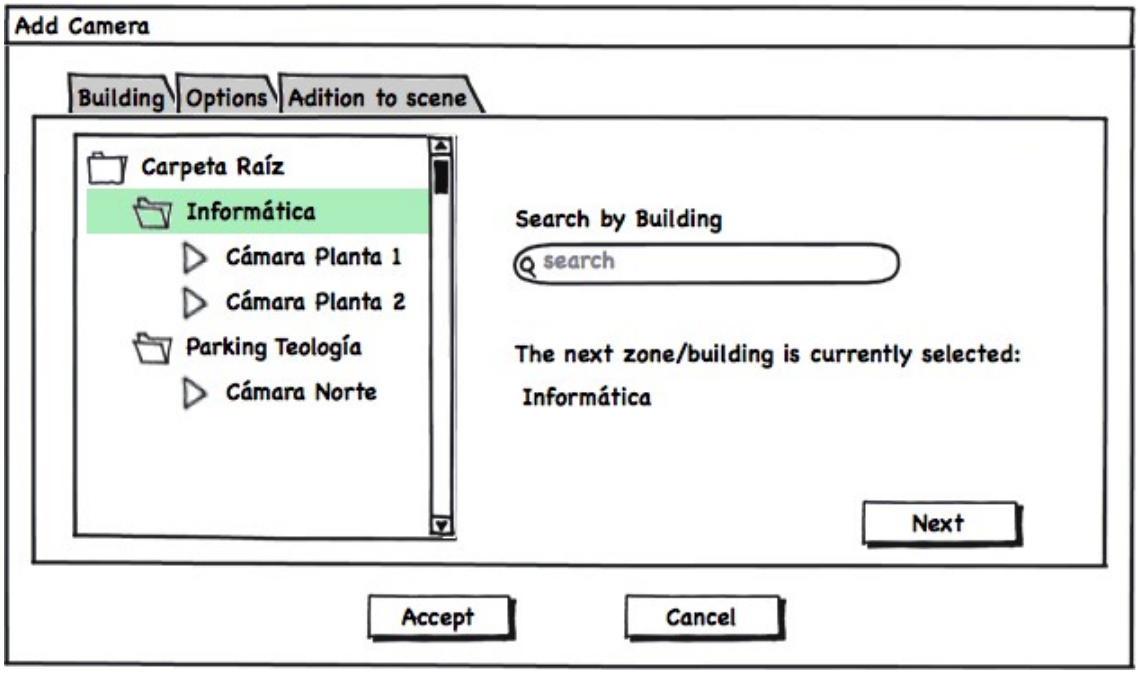

Ilustración 53: Mockup de añadir cámara a la escena, pestaña Building.

14.2.4 Añadir Cámara – Opciones

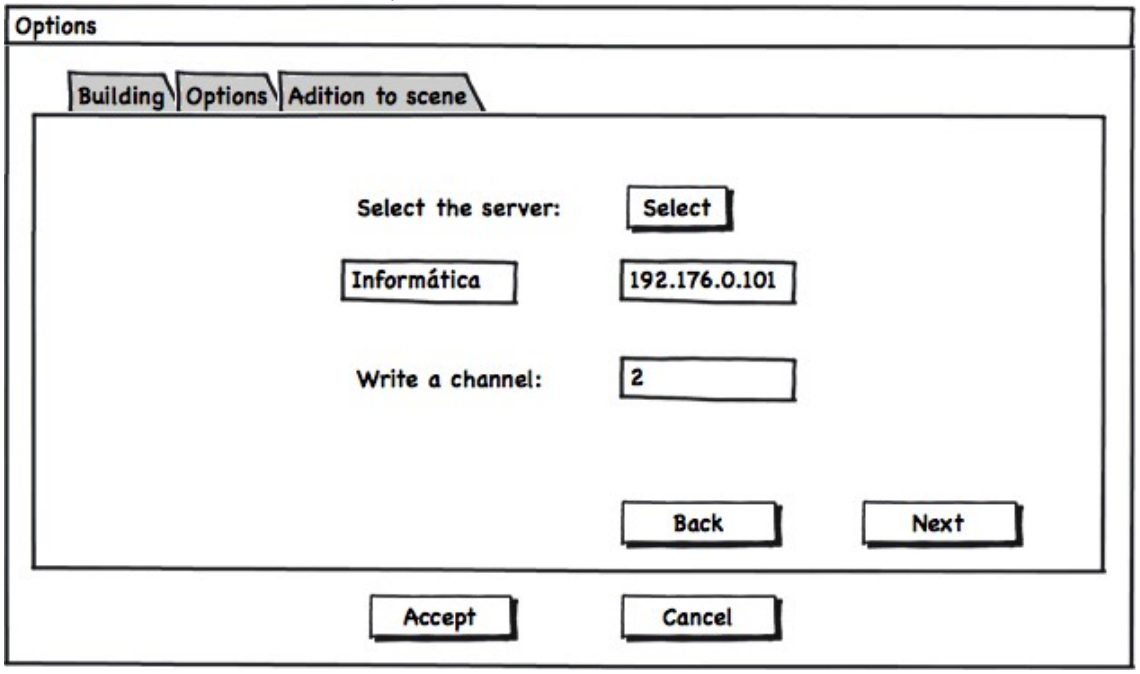

Ilustración 54: Mockup de añadir cámara a la escena, pestaña Options.

14.2.5 Añadir Cámara - Adición a escena

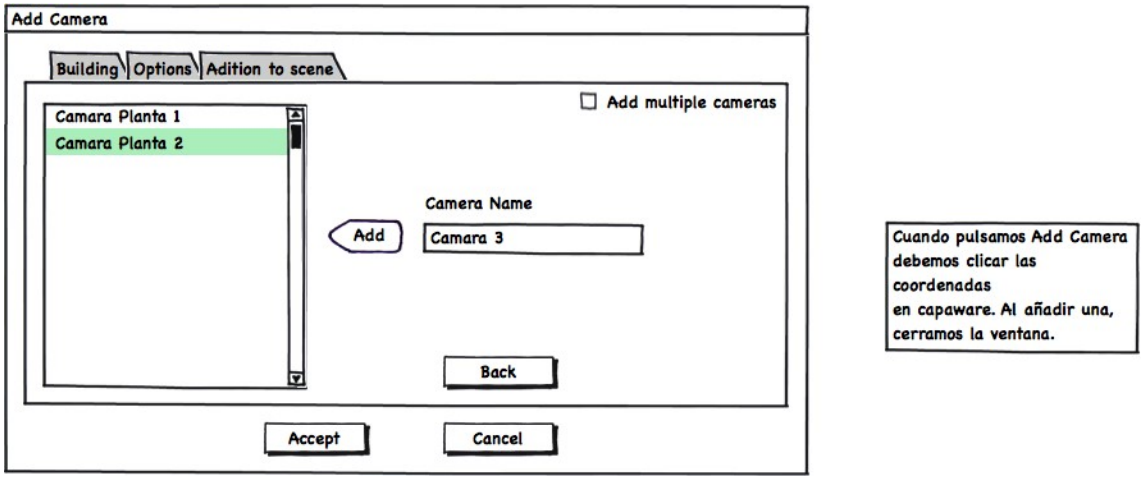

Ilustración 55: Mockup de añadir cámara a la escena, pestaña Adition to scene.

14.2.6 Visualización

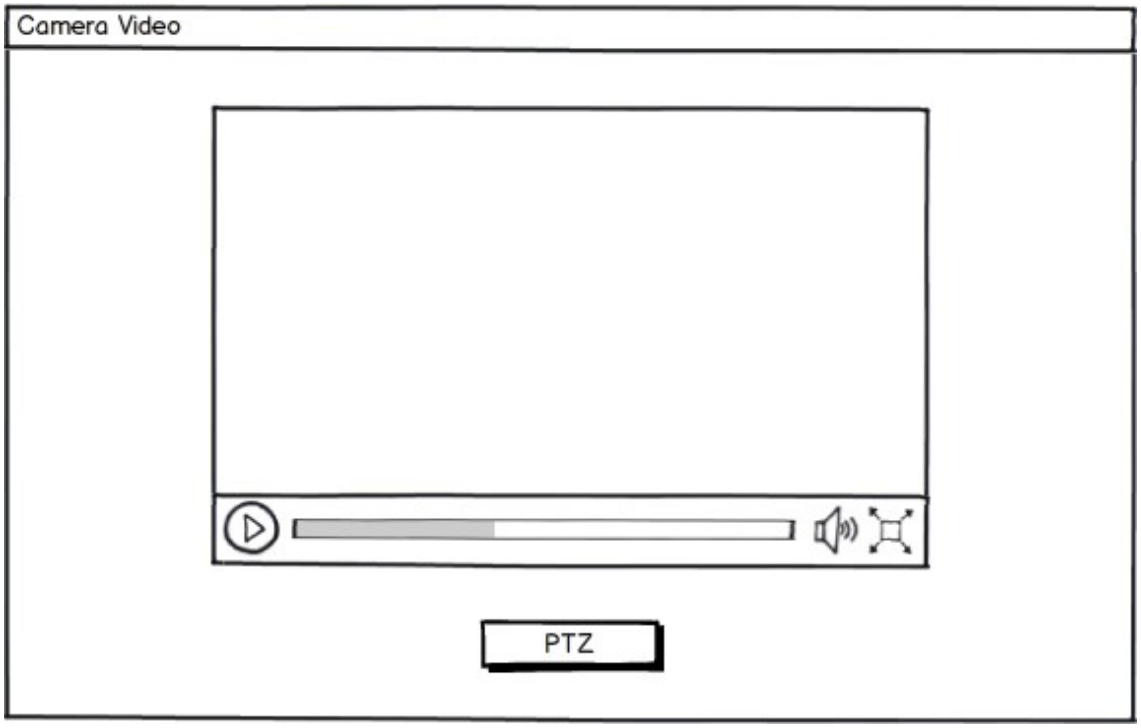

Ilustración 56: Mockup de visualizar video.

14.2.7 Mosaico

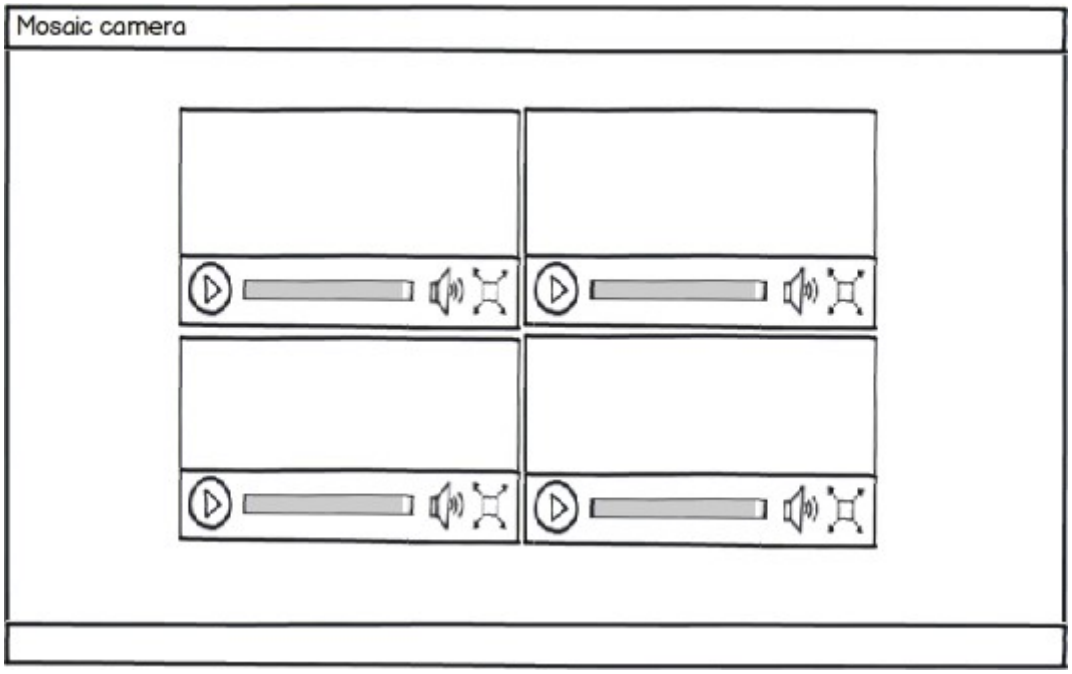

Ilustración 57: Mockup de visualizar video en mosaico de cámaras.

14.2.8 Control PTZ

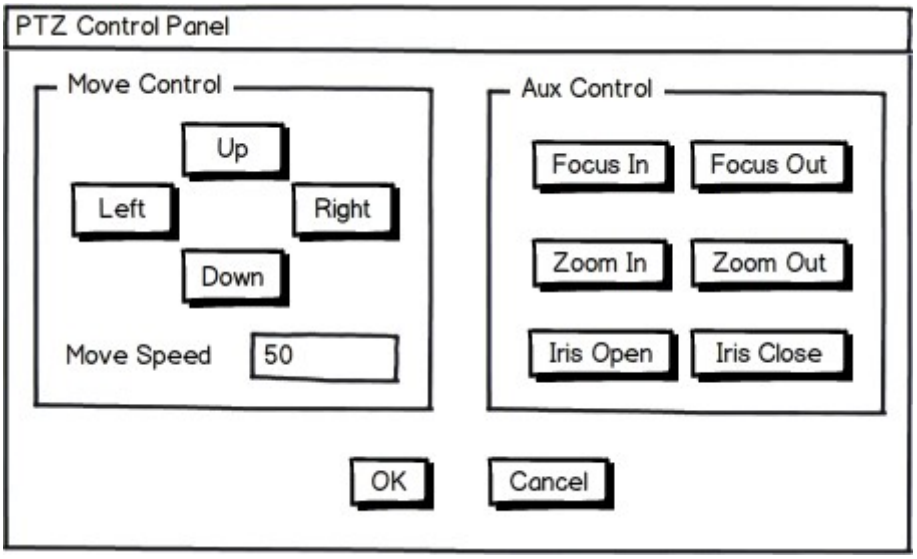

Ilustración 58: Mockup de controles PTZ.

14.2.9 Añadir servidor / Selección de servidor

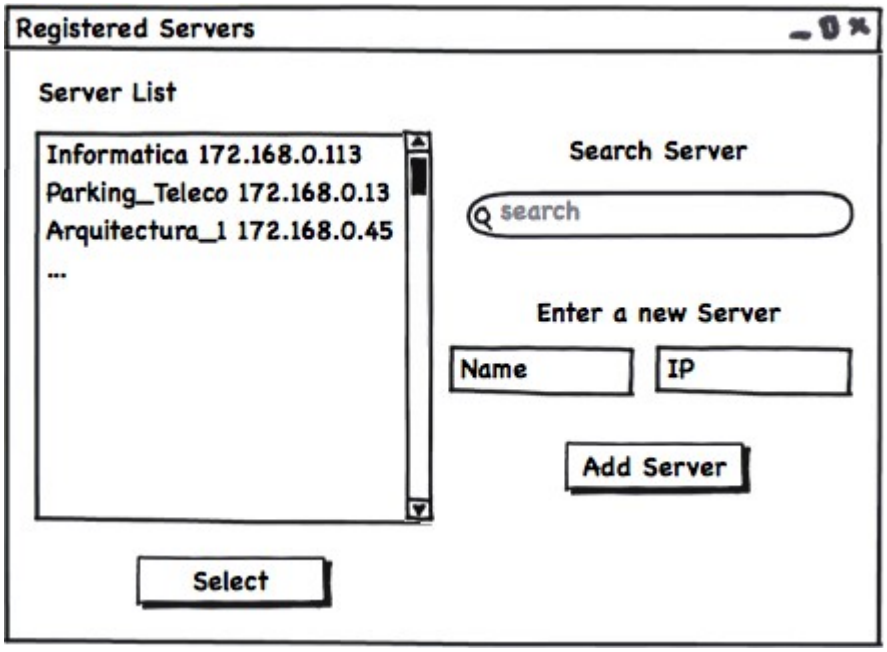

Ilustración 59: Mockup añadir servidor o seleccionar servidor.

14.2.10 Estado de conexión

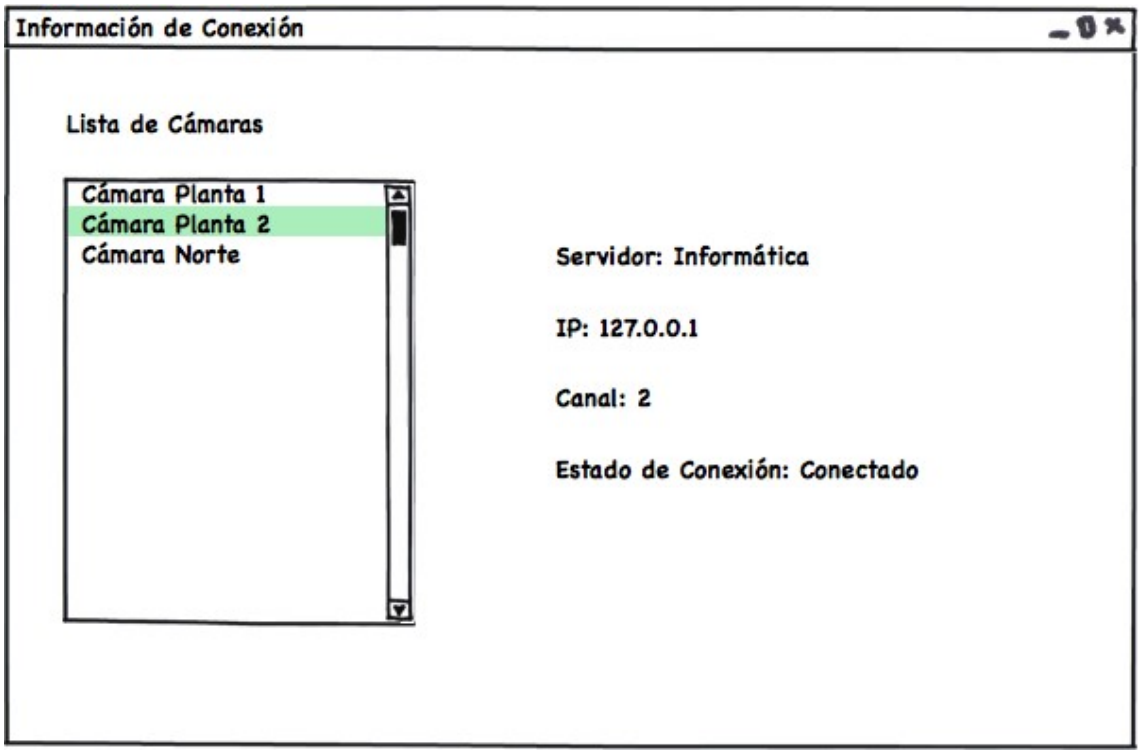

Ilustración 60: Mockup de estado de conexión.

14.2.11 Crear mosaico de cámaras - Lista de cámaras

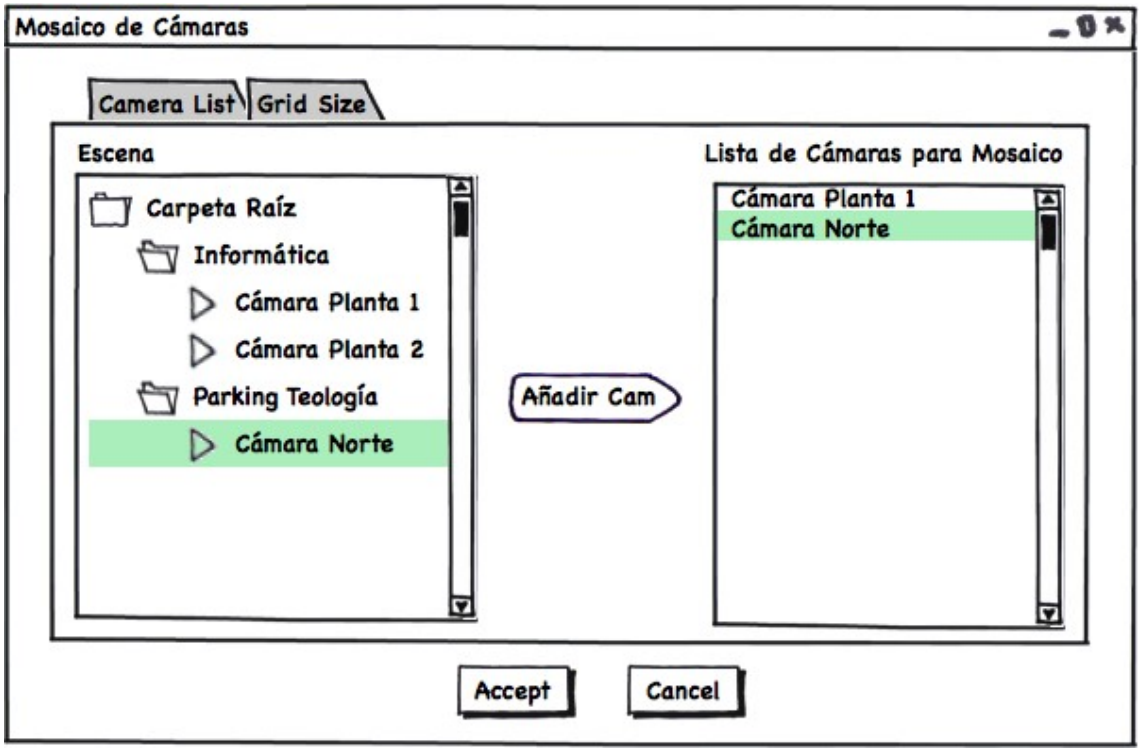

Ilustración 61: Mockup de crear mosaico, lista de cámaras a seleccionar.

14.2.12 Crear mosaico de cámaras - Tamaño de la malla

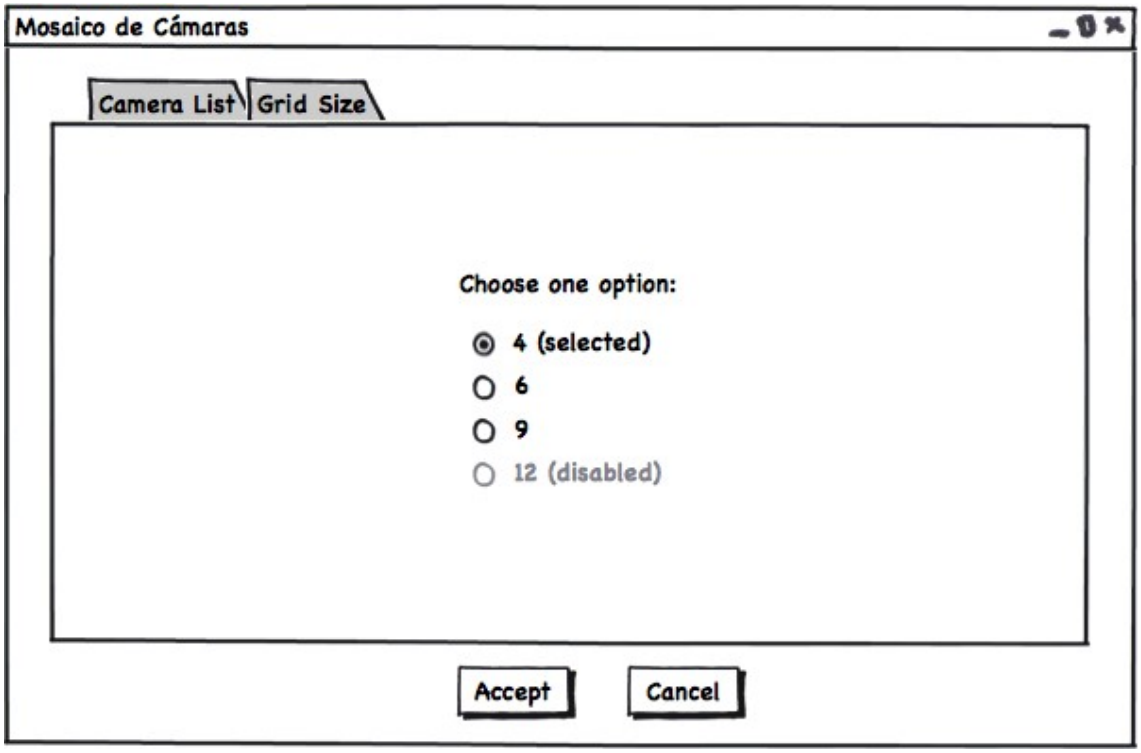

Ilustración 62: Mockup de crear mosaico, lista de cámaras a seleccionar.

14.2.13 Login de usuario

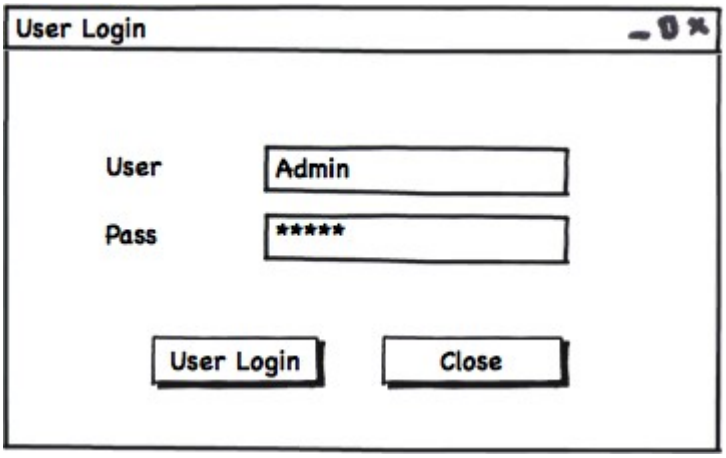

Ilustración 63: Mockup de autentificación de usuario.

14.3 Instalación

La instalación consta de tres pasos: instalación de la aplicación Capaware, instalación de la base de datos MySQL e instalación y configuración del plugin. A continuación, se describe brevemente como llevar a cabo los dos primeros pasos, además de detallar como hacer el tercero.

14.3.1 Paso 1: Instalación de Capaware

La instalación de Capaware puede llevarse a cabo de dos maneras: mediante el instalador de la aplicación o realizando una compilación de los archivos binarios del proyecto y generando el ejecutable. El instalador es muy sencillo de usar ya que se trata de una instalación típica de los programas de Windows, siendo sin duda la manera más sencilla de llevar a cabo la instalación de Capaware.

La otra opción consiste en descargar los archivos binarios de Capaware, los cuales son cuatro archivos comprimidos en formato ZIP llamados Capaware_code_apps_rc2, Capaware_code_src_rc2, Capaware_dep_extern_rc2 y Capaware_struct_bin_rc2.

En primer lugar, descomprimimos los ficheros fuente (Capaware_code_src_rc2) en la carpeta donde se desee volcar Capaware. Posteriormente, descomprimimos en la misma carpeta los ficheros fuente de las aplicaciones (Capaware_code_aps_rc2). En tercer lugar, descomprimimos en la misma carpeta la estructura bin (Capaware_struct_bin_rc2). Finalmente repetimos el proceso con las dependencias externas (Capaware_dep_extern_rc2).

Antes de la compilación, es necesario añadir algunas rutas a las variables de entorno del sistema. Dentro de la variable de entorno PATH, añadimos la ruta: *directorio_instalación*\extern\bin, siendo *directorio_instalación* la carpeta donde se han descomprimido los ficheros anteriormente descritos. Por último, creamos una nueva variable de entorno de nombre PROJ_LIB, siendo su valor: *directorio_instalación*\bin\configuration\proj4.7

Una vez completados estos pasos, procedemos a realizar la compilación de Capaware. Se recomienda usar alguna versión de Microsoft Virtual Studio Express. Abrimos el fichero con formato .sln y compilamos la solución en el modo que se desee (Release o Debug). Una cosa

importante para la compilación es que, en las propiedades del proyecto, opciones de Depuración y la variable Directorio de trabajo, habrá que ponerse la ruta: *directorio_instalación*\bin

14.3.2 Paso 2: Instalación de BBDD MySQL

En el manual de referencia de Capaware, se describe como realizar la instalación de una base de datos para gestionar la persistencia en la aplicación. En concreto, el apartado 7.1.1 (página 15) describe como llevar a cabo este proceso.

14.3.3 Paso 3: Instalación y configuración del plugin

Antes de instalar el plugin en Capaware, es necesario realizar un paso previo de configuración del plugin en Capaware. Para que Capaware sea capaz de reconocer el plugin, nos bastará con añadir una línea en el fichero pluginsfile.dat dentro del directorio *directorio_instalacion*/bin/. En este fichero, indicaremos la ruta completa del plugin.

Se recomienda situar el plugin dentro de una carpeta creada por el usuario, en el directorio *directorio_instalación*/bin/data/. Creamos, por ejemplo, la carpeta pfc y copiamos el contenido de la carpeta Instalador plugin proporcionada en la entrega del PFC. Con el fichero de extensión .dll es suficiente, aunque se recomienda copiar el resto de archivos.

Siguiendo el ejemplo, se nos quedaría en el fichero pluginsfile.dat de la siguiente manera:

directorio_instalacion/bin/data/pfc/plugin.dll,

siendo *directorio_instalación* la ruta completa donde se instaló Capaware.

A continuación, copiaremos los ficheros de la API en el directorio de instalación de Capaware, donde esté situado el .exe del programa. Estos ficheros serán proporcionados en la entrega del PFC.

Para finalizar, faltará crear dos tablas adicionales en la base de datos de Capaware. Abrimos una consola de línea de comandos cmd y nos situamos en el directorio de instalación de Capaware. Escribimos el comando:

sqlite3.exe capaware.db3

Ahora creamos las dos tablas:

CREATE TABLE cameraserver(id INTEGER PRIMARY KEY AUTOINCREMENT, idcpw VARCHAR(255) NOT NULL, ser TEXT)

CREATE TABLE cameraobject(id INTEGER PRIMARY KEY AUTOINCREMENT,idcpw VARCHAR(255) NOT NULL, serchn TEXT)

Con esto, escribimos ".exit" para salir de SQLite y queda finalizada la instalación del plugin.

14.4 Manual básico de uso

En esta sección se describe cómo utilizar las funcionalidades añadidas en este PFC.

14.4.1 Añadir cámara a la escena

Para añadir una cámara a la escena, pulsaremos en el menú Plugins y elegimos la opción *Add Camera* (ver ilustración 64).

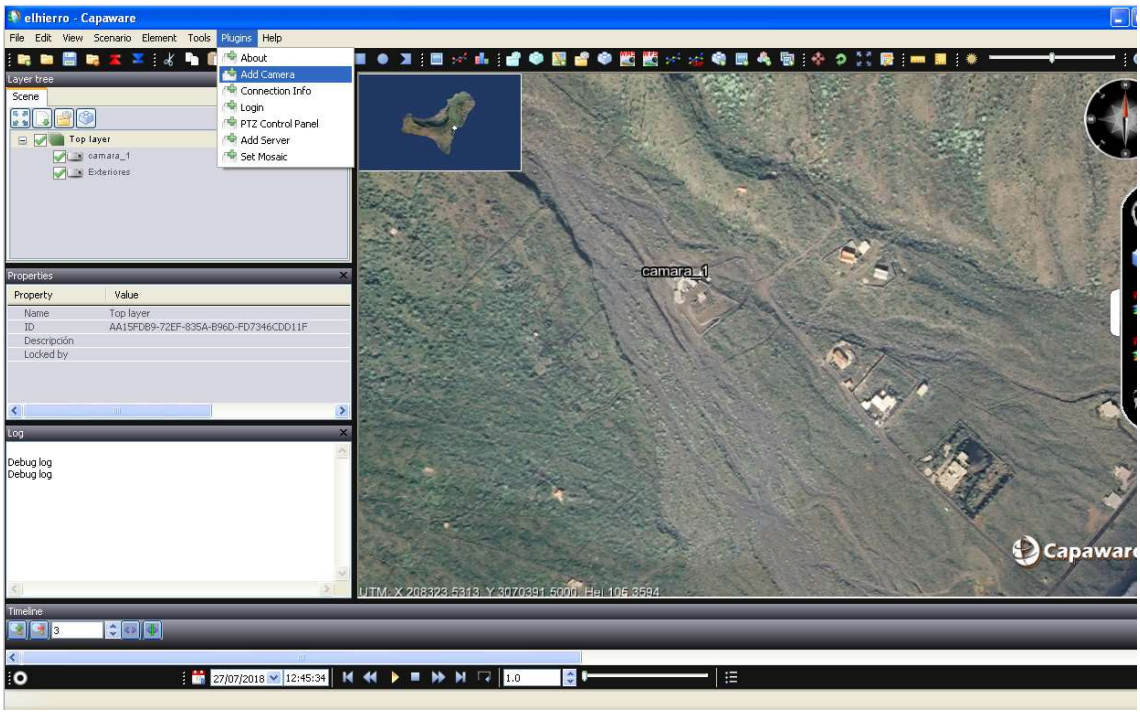

Ilustración 64: Menú Plugins – Add Camera.

Una vez hecho clic, se nos presenta una ventana con diferentes pestañas, donde rellenaremos la información necesaria para llevar a cabo la adición. La primera pestaña que se nos muestra es la de *Buildings,* la cual nos permite seleccionar en qué posición de la jerarquía de la escena queremos añadir la cámara (ver ilustración 65).

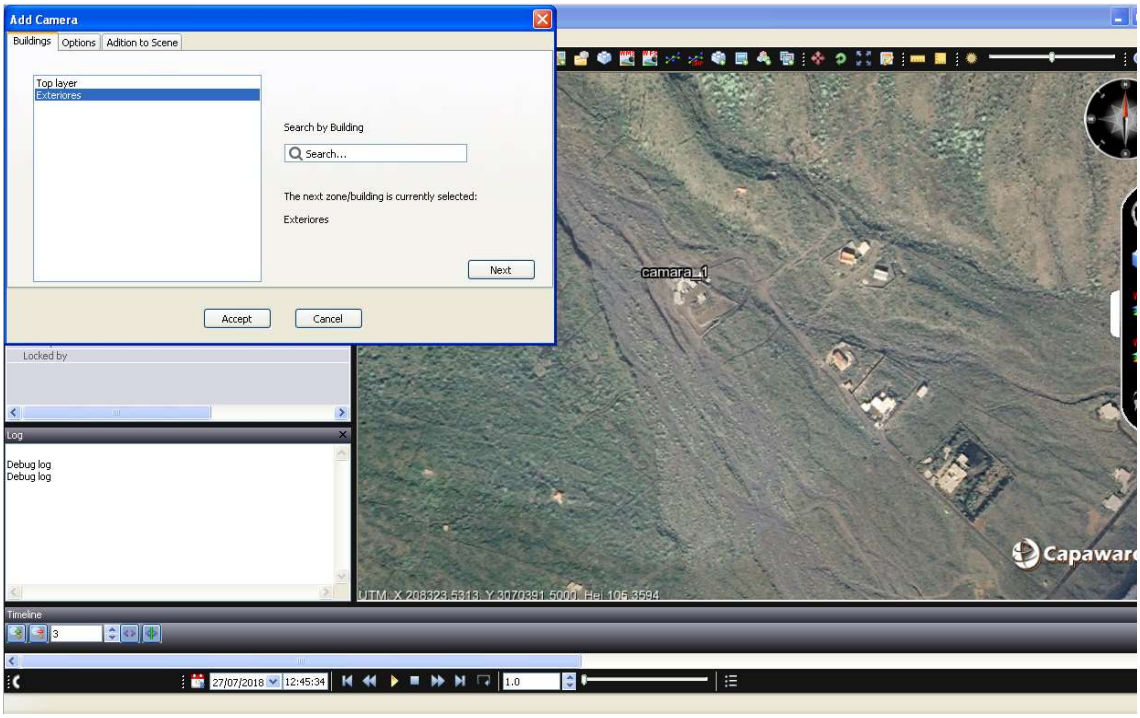

Ilustración 65: Add Camera - Buildings.

En la siguiente pestaña, rellenaremos la información asociada a la cámara. Primero pulsamos en *Select*, donde se nos abrirá la lista de servidores (la misma vista que añadir servidor) y elegimos el servidor asociado a la cámara. Una vez seleccionado el servidor, veremos el nombre del servidor y su dirección IP en la vista *Options*. A continuación, rellenamos el último campo poniendo en que canal está conectada la cámara (ver Ilustración 66).

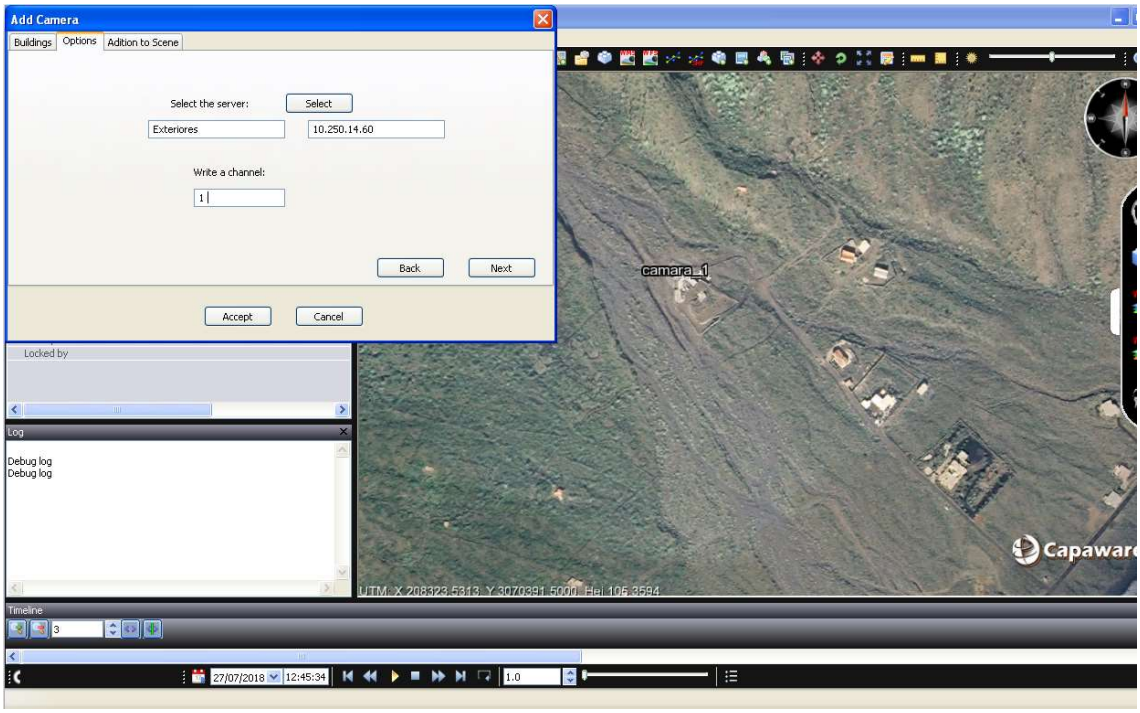

Ilustración 66: Add Camera - Options.

Al darle a *Next*, llegaremos a la pestaña *Addition to Scene*. En ella, escribimos el nombre de la cámara y seleccionamos *Add*. En caso de querer añadir múltiples cámaras, podemos seleccionar la casilla *Add Multiple Cameras*. Dicha casilla se encuentra en la parte superior derecha de esta vista. La lista de la parte izquierda irá mostrando que cámaras se encuentran actualmente en la carpeta contenedora o nivel del árbol seleccionado en la primera pestaña (ver Ilustración 67).

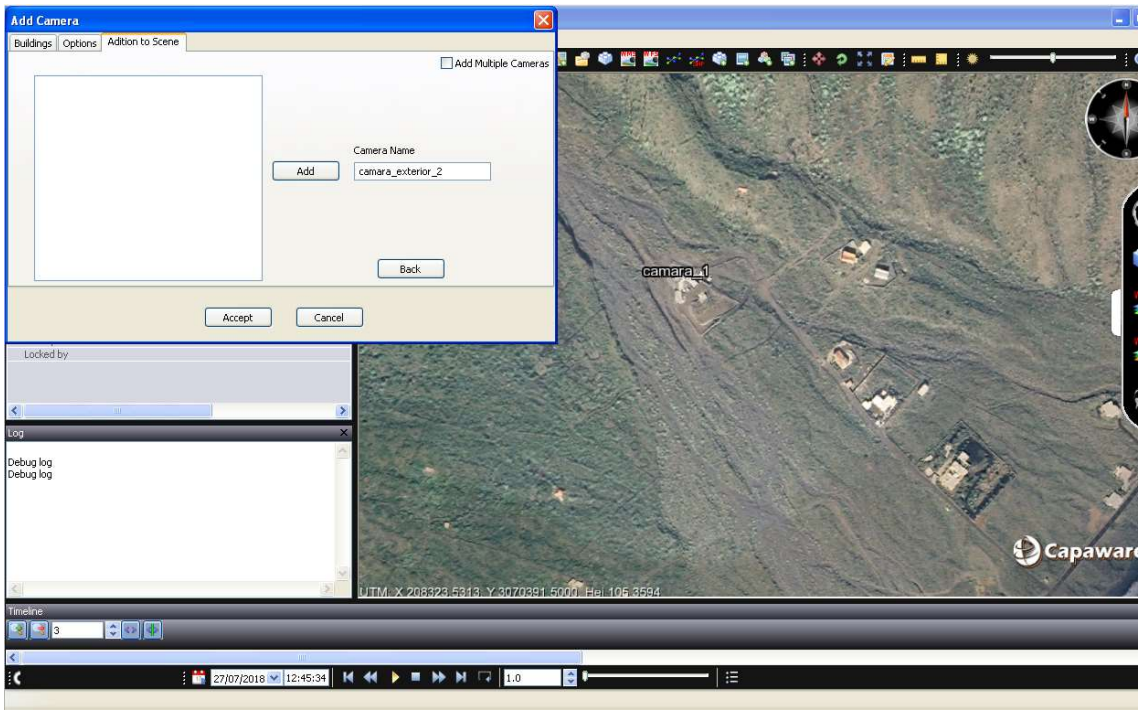

Ilustración 67: Add Camera – Addition to Scene.

Al pulsar el botón *Add*, se nos abrirá una pequeña ventana donde se nos indica pulsar en una posición de la escena donde queramos añadir la cámara (ver Ilustración 68).

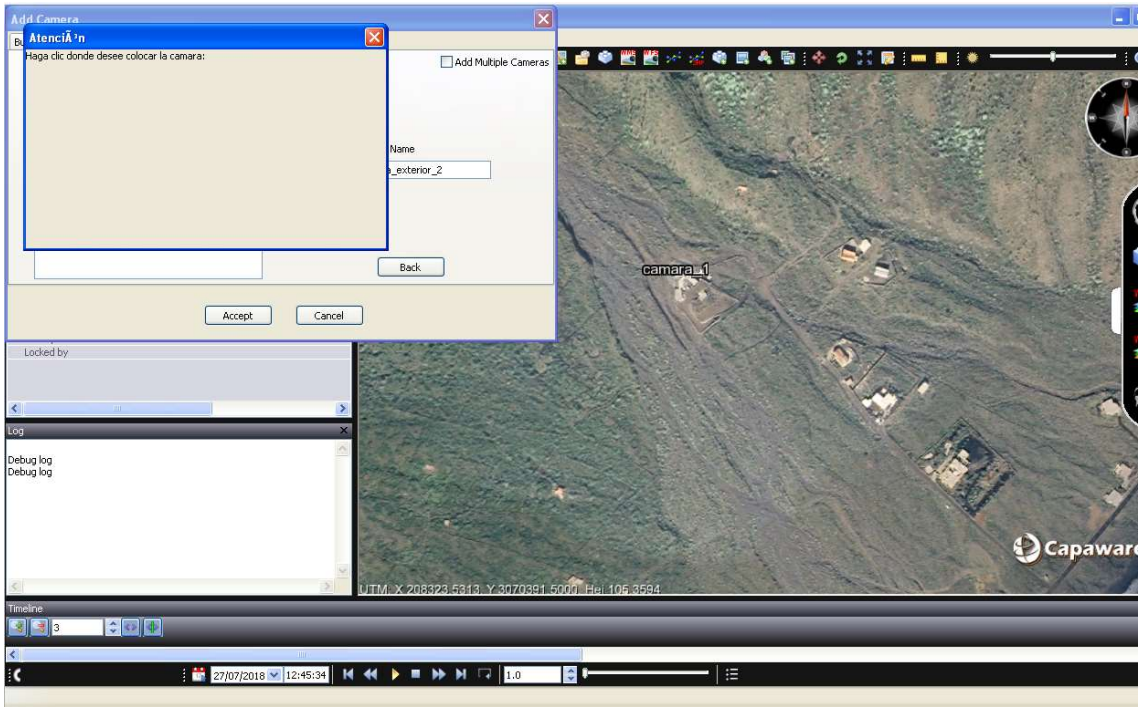

Ilustración 68: Add Camera – Seleccionar posición de cámara.

Pulsamos sobre la posición que queramos con el botón izquierdo del ratón y veremos cómo se añade la cámara a la escena (ver Ilustración 69).

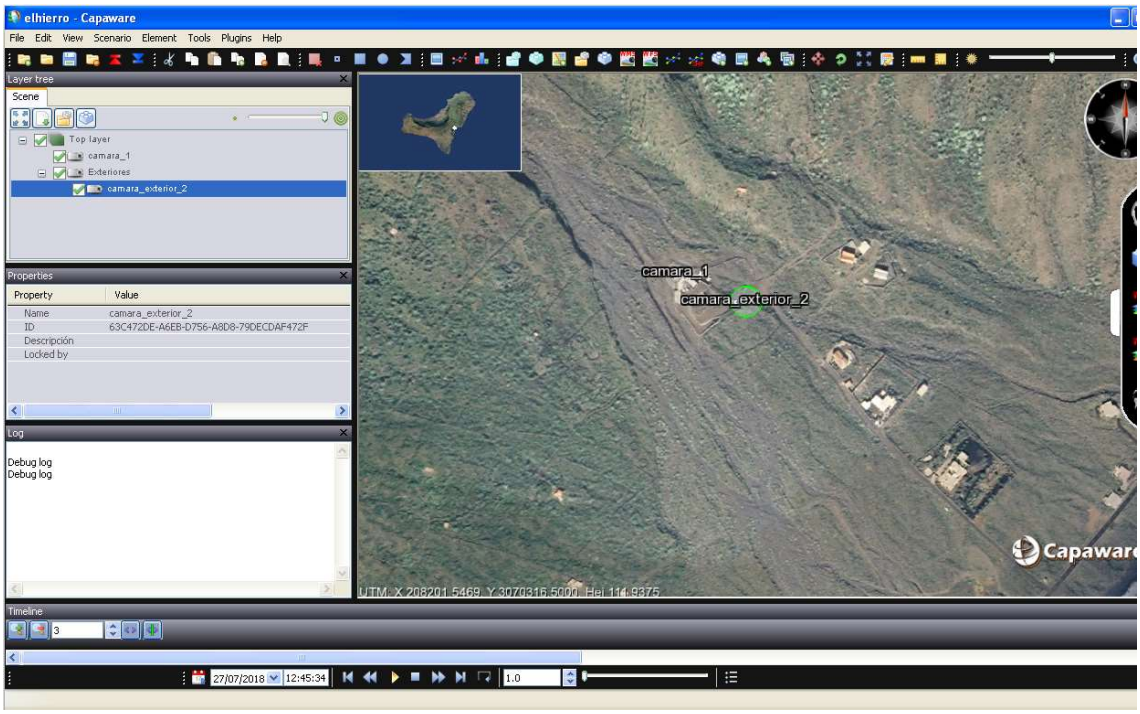

Ilustración 69: Add Camera – Cámara añadida.

14.4.2 Añadir servidor

Para añadir un servidor a la lista de servidores de Capaware, iremos dentro del menú *Plugins* y seleccionamos la opción *Add Server* (ver Ilustración 70).

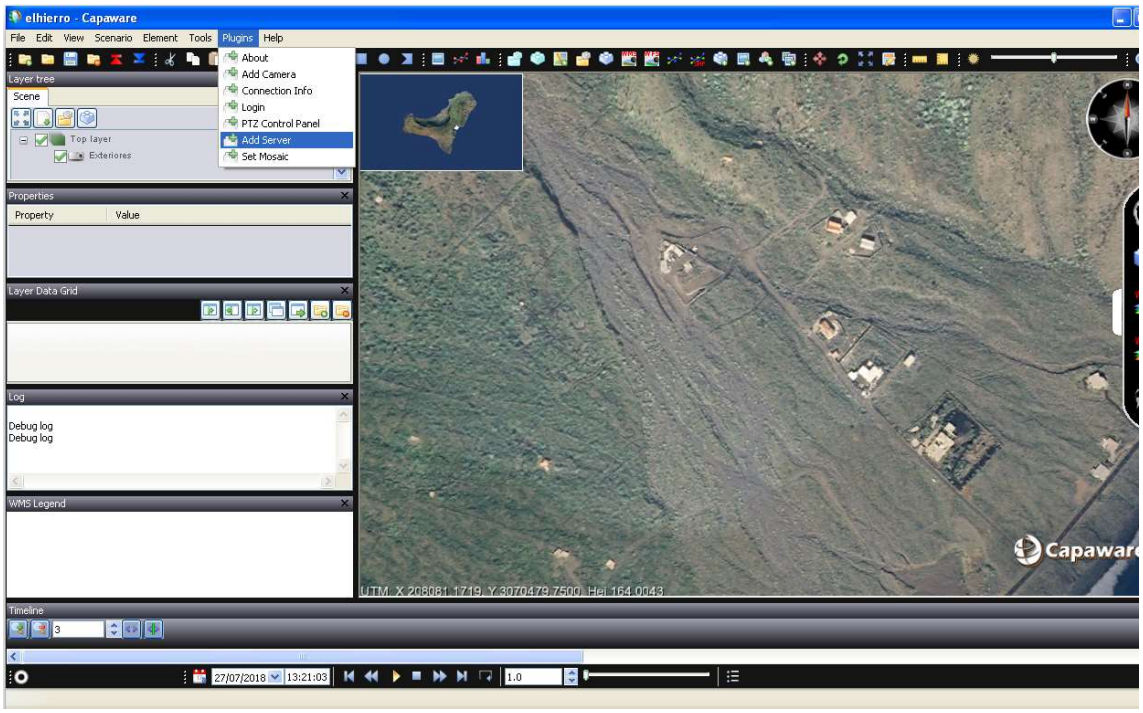

Ilustración 70: Menú Plugins – Add Server.

Se nos presenta una ventana donde tendremos varios elementos: en la parte izquierda veremos una lista de servidores ya añadidos. Disponemos además de un campo *Search*, útil para cuando queramos añadir una cámara a la escena, ya que esta ventana es la misma que se nos ofrece al elegir la opción *Select*, dentro de la pestaña *Options* de *Add Camera*. Por último, veremos dos campos que podemos rellenar con la información necesaria de un servidor. En el primero escribimos el nombre del servidor (etiqueta asociada al servidor) y en el segundo añadimos la dirección IP (ver Ilustración 71).

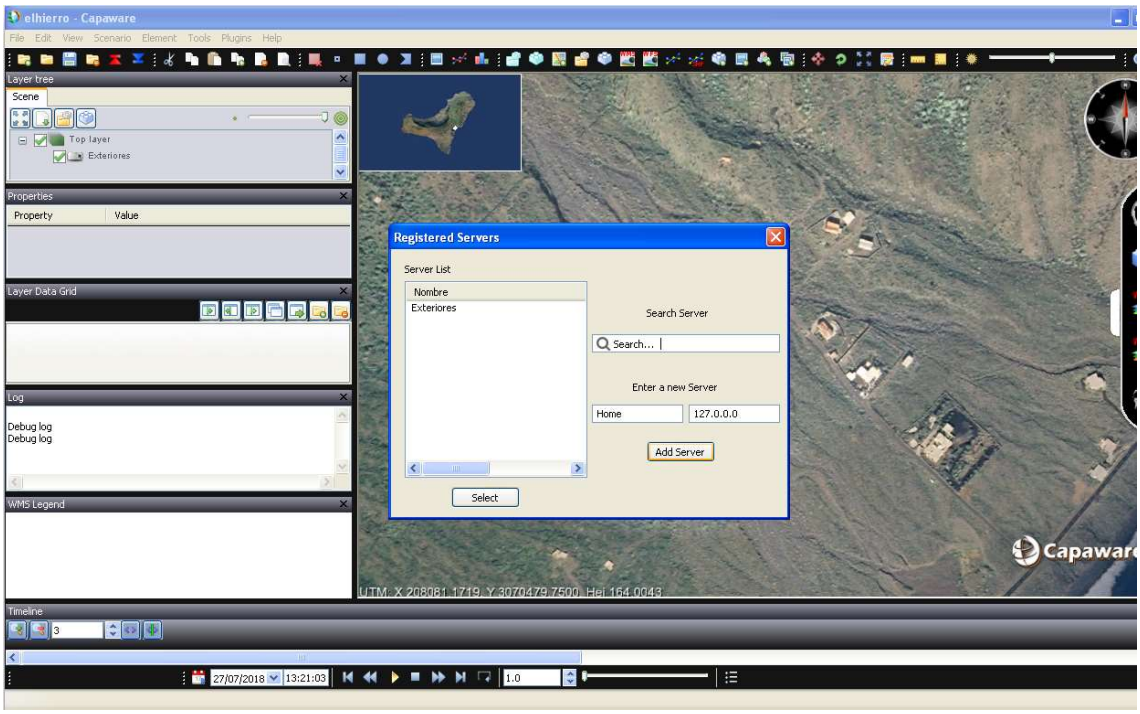

Ilustración 71: Add Server.

Una vez rellenado los campos mencionados, pulsamos en *Add Server*. Si la información es correcta se nos indicará que se añadió el servidor de manera satisfactoria como se enseña en la imagen posterior. En caso de introducir información redundante (etiqueta o dirección IP ya registrada) se mostrará un mensaje de error y no se añadirá el servidor.

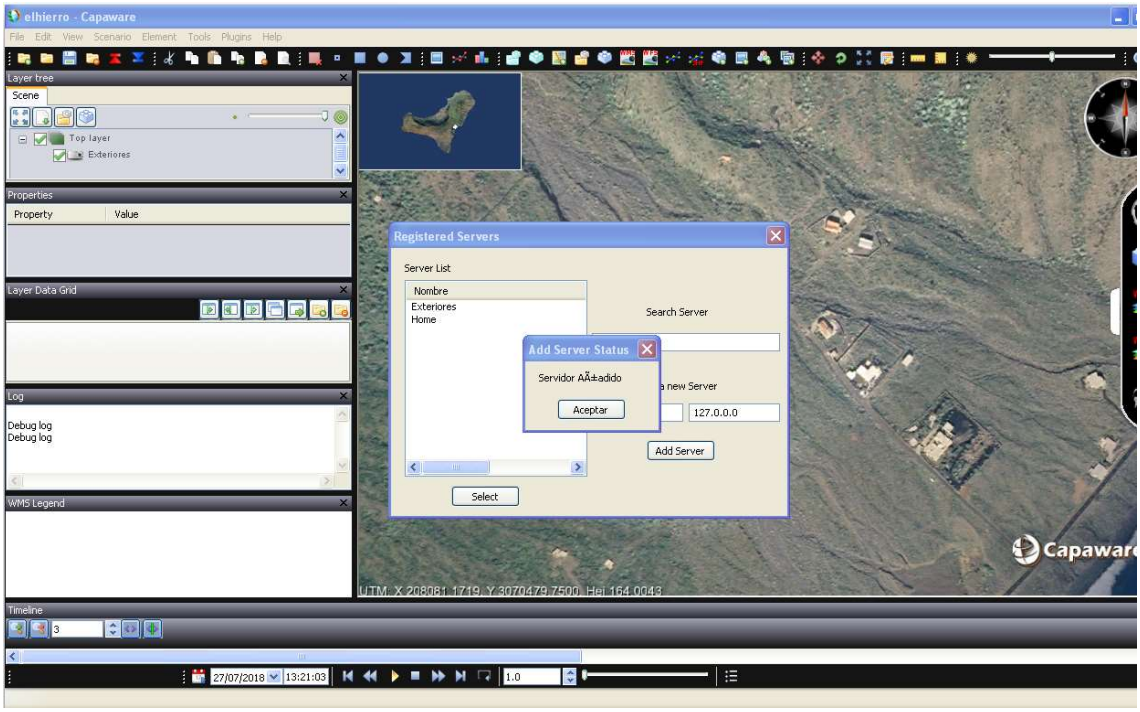

Ilustración 72: Add Server – Servidor registrado.

14.4.3 Conectar a servidor

La conexión a un servidor requiere de haber añadido al menos una cámara a la escena en Capaware perteneciente a dicho servidor. Seleccionando la cámara, hacemos clic derecho con el ratón y elegimos dentro del submenú *Plugins*, la opción Connect to Server (ver Ilustración 73).

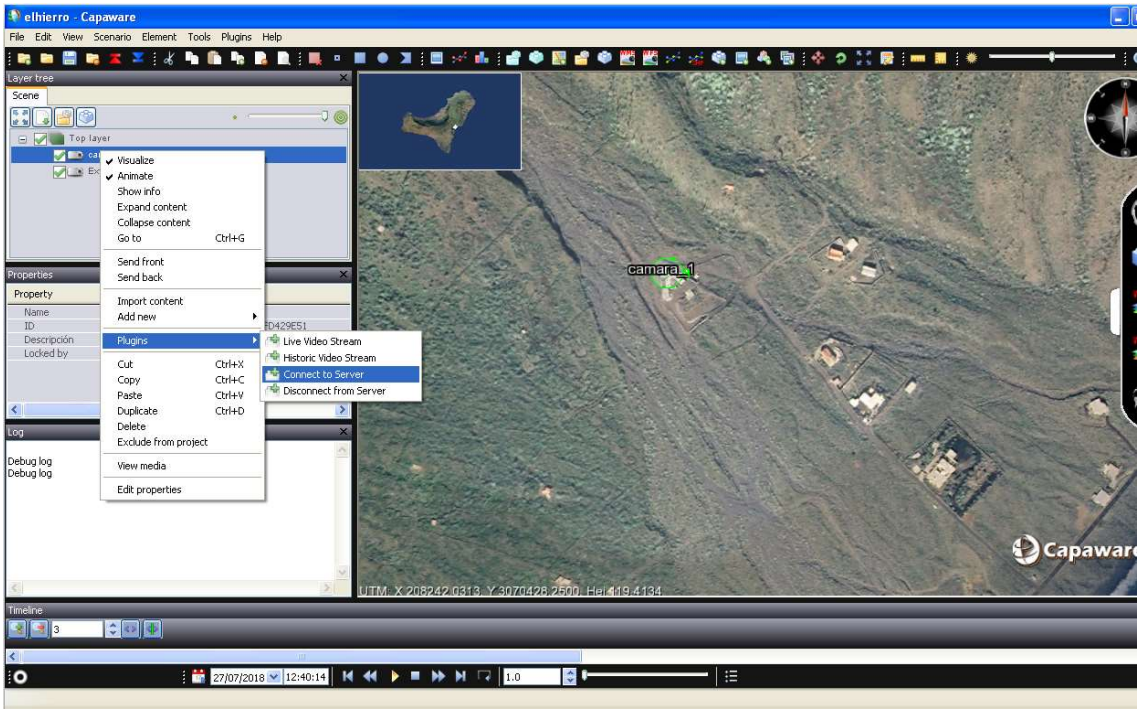

Ilustración 73: Submenú Plugins de Element3D – Connect to Server.

En principio no se verá ningún mensaje si la conexión ha sido satisfactoria. Existen dos formas de observar realmente si la conexión se ha establecido correctamente: la primera de ellas es a través de la interfaz de Información de conexión (apartado 15.4.5 de este anexo). La segunda es a través de la línea de comandos que se abre con la ejecución de Capaware. Aunque esta opción fue utilizada en principio con el objetivo de ayudar en fases de pruebas y depuración, se ha decidido dejar en la versión final (ver Ilustración 74).

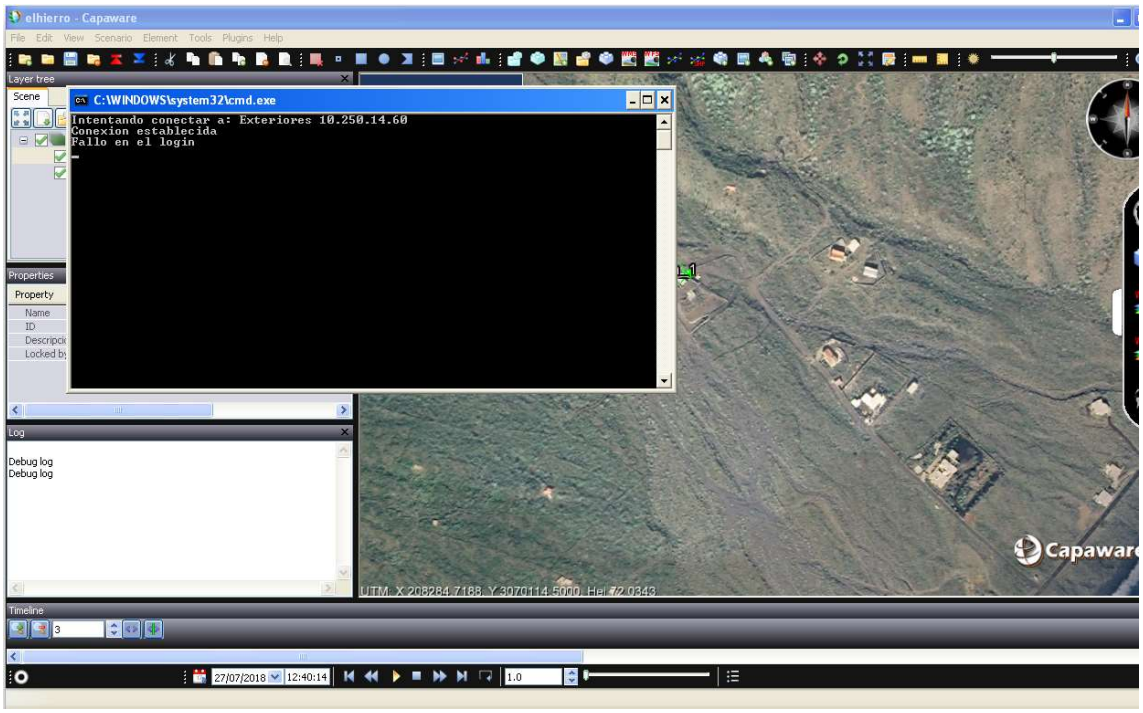

Ilustración 74: Conectando a servidor, línea de comandos.

14.4.4 Desconectar de servidor

La desconexión de un servidor funciona de manera similar a establecer conexión con un servidor descrita en la sección previa. Es necesario también tener una cámara añadida a la escena. Seleccionamos una cámara perteneciente al servidor que queramos desconectar y hacemos clic derecho con el ratón. Dentro del submenú *Plugins*, elegimos la opción *Disconnect from Server* (ver Ilustración 75).

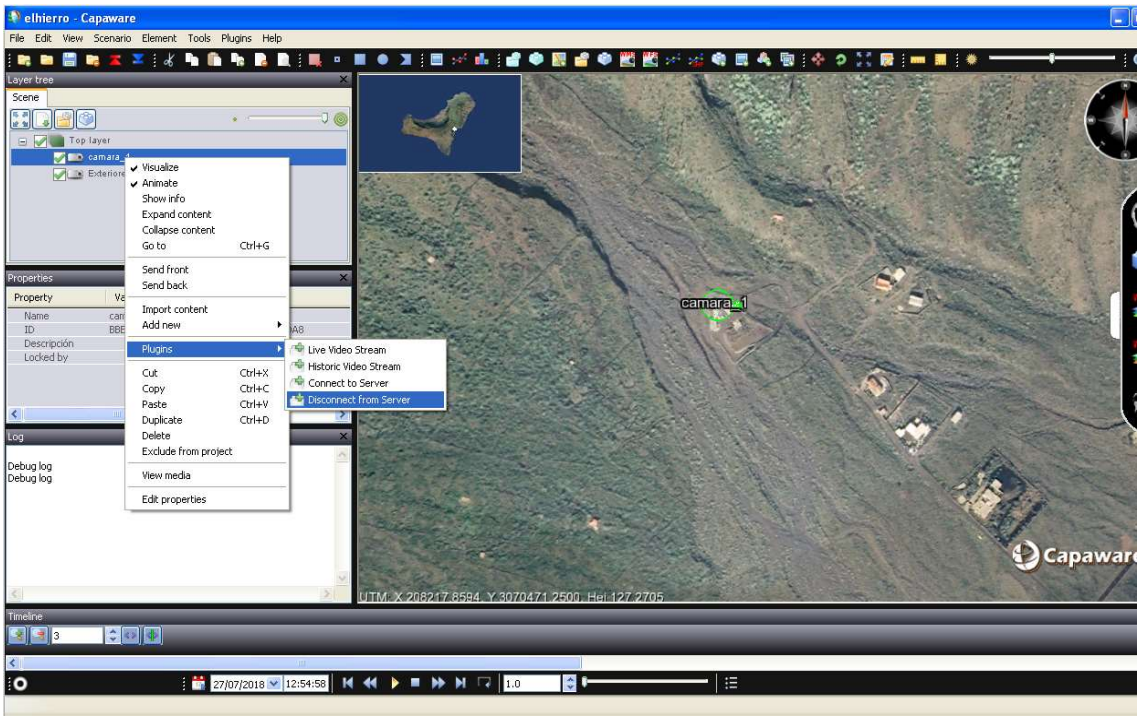

Ilustración 75: Submenú Plugins de Element3D – Disconnect from Server.

Como ocurría en la sección previa, no aparecerá ningún mensaje informando de que la desconexión fue satisfactoria. Si queremos asegurarnos de que la desconexión se ha producido de manera correcta, usaremos los mismos métodos descritos anteriormente: mediante el uso de la interfaz información de conexión, o bien mediante la línea de comandos abierta con la ejecución de Capaware.

14.4.5 Ver información de conexión

Si se quisiera ver el estado de conexión de las cámaras presentes en la escena, podemos verlas mediante la vista *Connection Info*. Para ello, nos dirigimos al menú Plugins y seleccionamos dicha opción (ver Ilustración 76).

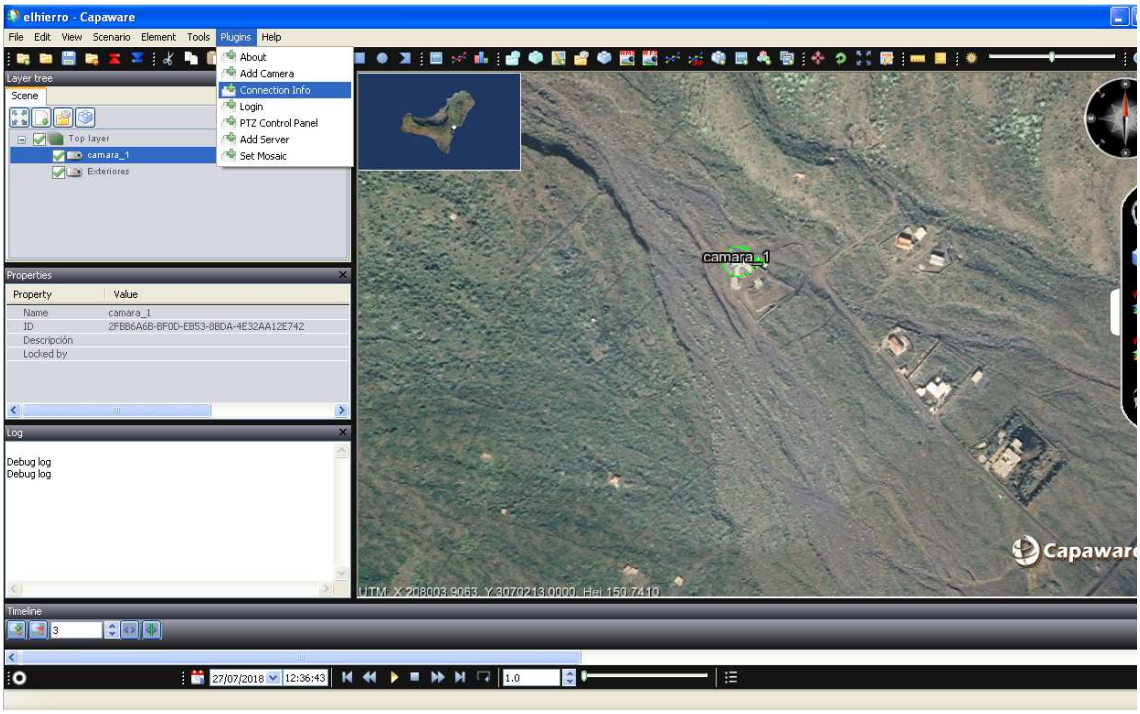

Ilustración 76: Menú Plugins – Connection Info.

Se nos abrirá una vista donde tendremos una lista de todas las cámaras presentes en la escena (sin jerarquía) en la parte izquierda. Si seleccionamos una cámara, se nos rellenará la información asociada a esa cámara en la parte derecha. Dicha información es: servidor al que está conectado, dirección IP, canal y estado de conexión (ver Ilustración 77).

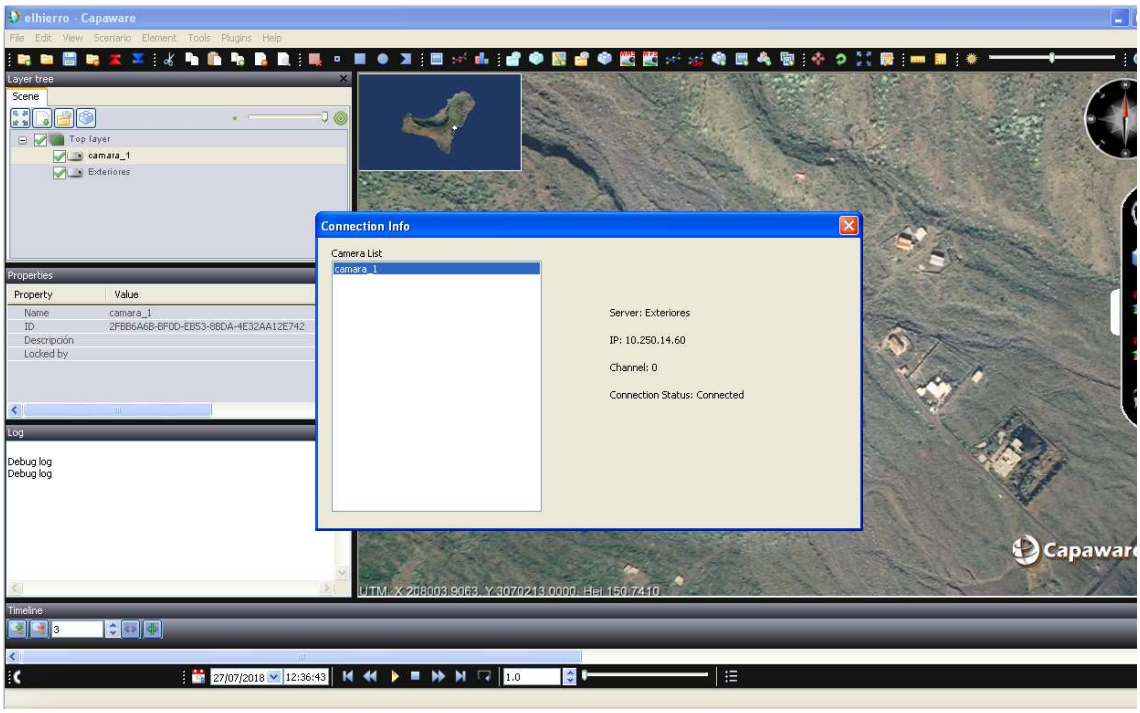

Ilustración 77: Información de conexión de cámaras

14.4.6 Autentificación de usuario

El proceso de autentificación del usuario en el sistema es muy sencillo. Pulsamos en el menú *Plugins* y elegimos la opción *Login*.

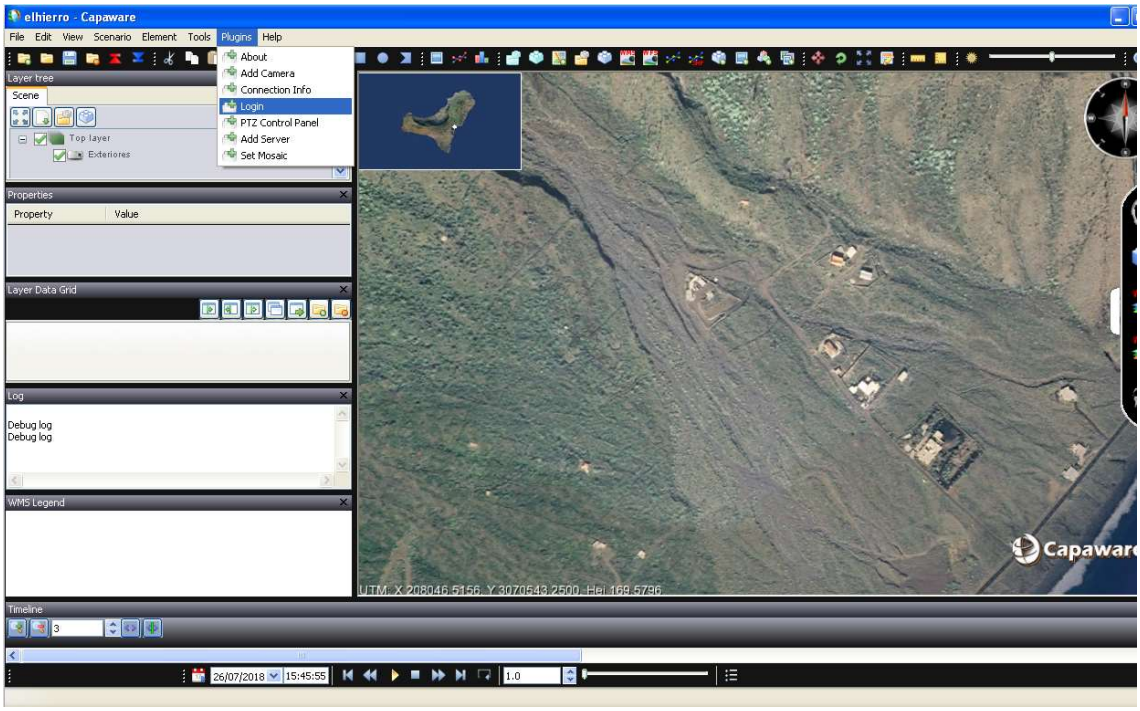

Ilustración 78: Menú Plugins – Login.

Se nos abrirá un formulario típico de autentificación basado en el par Usuario – Contraseña tradicional. Simplemente rellenamos los datos y le damos al botón *OK*.

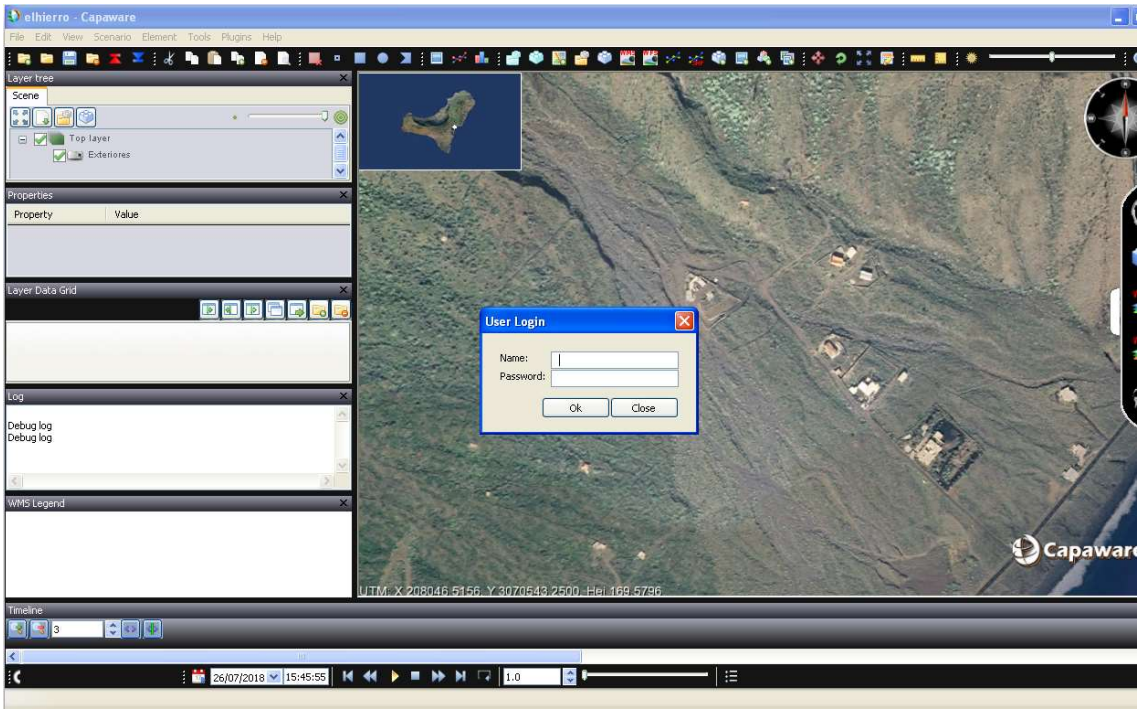

Ilustración 79: Login de usuario.

14.4.7 Ver video en directo

Para poder ver imagen de video en streamming, es necesario tener una cámara añadida en la escena y además haber establecido conexión con el servidor videograbador. Ambos procedimientos están explicados en apartados de este mismo anexo. Una vez cumplido los requisitos, seleccionamos la cámara de donde queremos obtener video y hacemos clic secundario con el ratón. Dentro del submenú Plugins, elegimos la opción *Live Video Stream* (ver Ilustración 80).

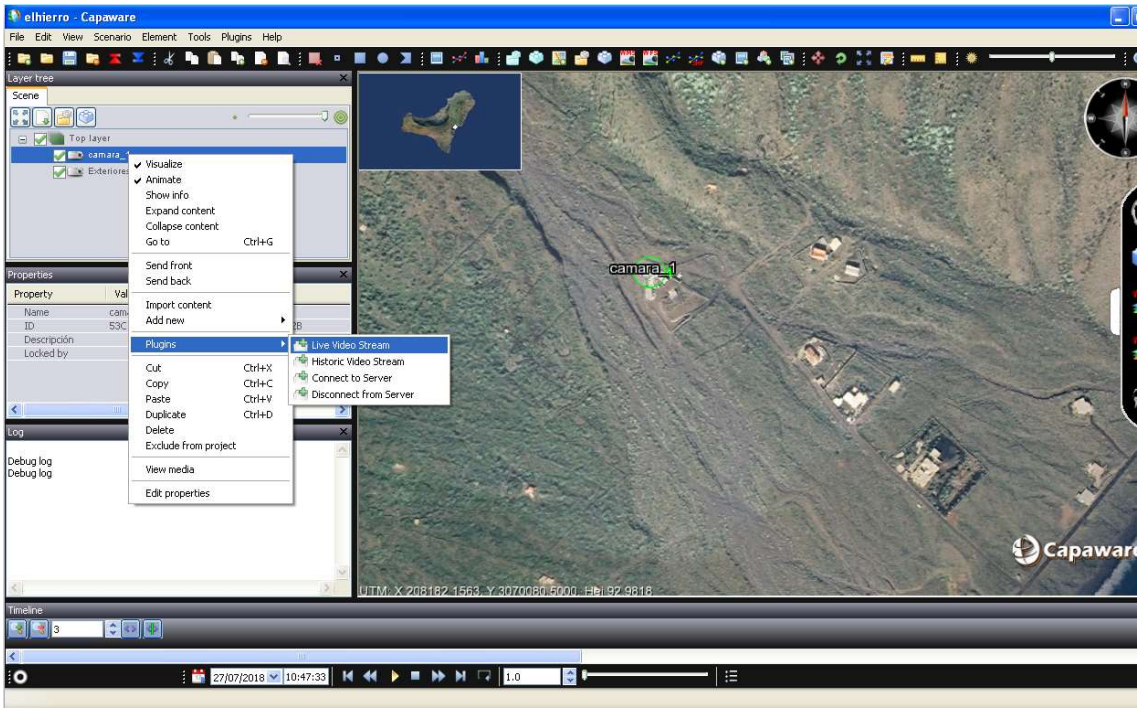

Ilustración 80: Submenú Plugins de Element3D – Live Video Stream.

Se nos abrirá una ventana presentando las imágenes de video de la cámara seleccionada, como se enseña a continuación.

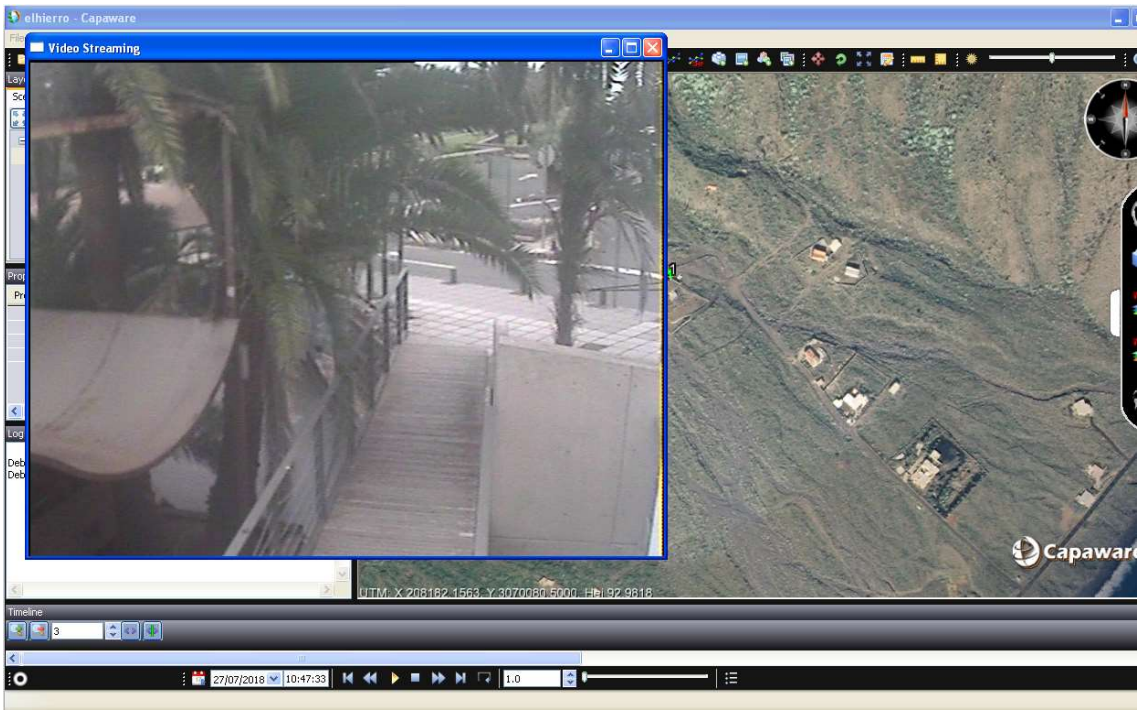

Ilustración 81: Imagenes de Video en streamming.

14.4.8 Controles PTZ

Los controles PTZ son funciones asociadas a cámaras, por lo que es necesario tener añadida alguna cámara en la escena. Con la cámara seleccionada, nos dirigimos al menú *Plugins* y elegimos la opción *PTZ Control Panel* (ver Ilustración 82).

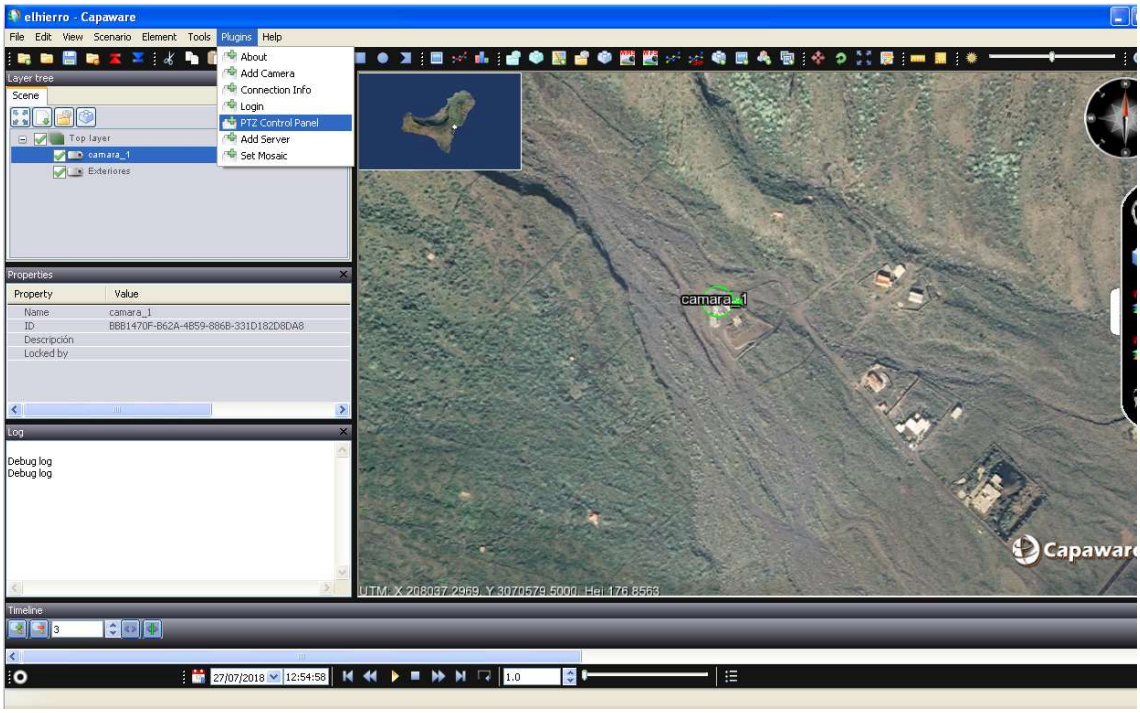

Ilustración 82: Menú Plugins – Panel de controles PTZ.

Se nos abrirá una ventana con una serie de controles implementadas en el proyecto. Dichos controles contemplan el movimiento de la cámara, hacer zoom y otros (ver Ilustración 83).

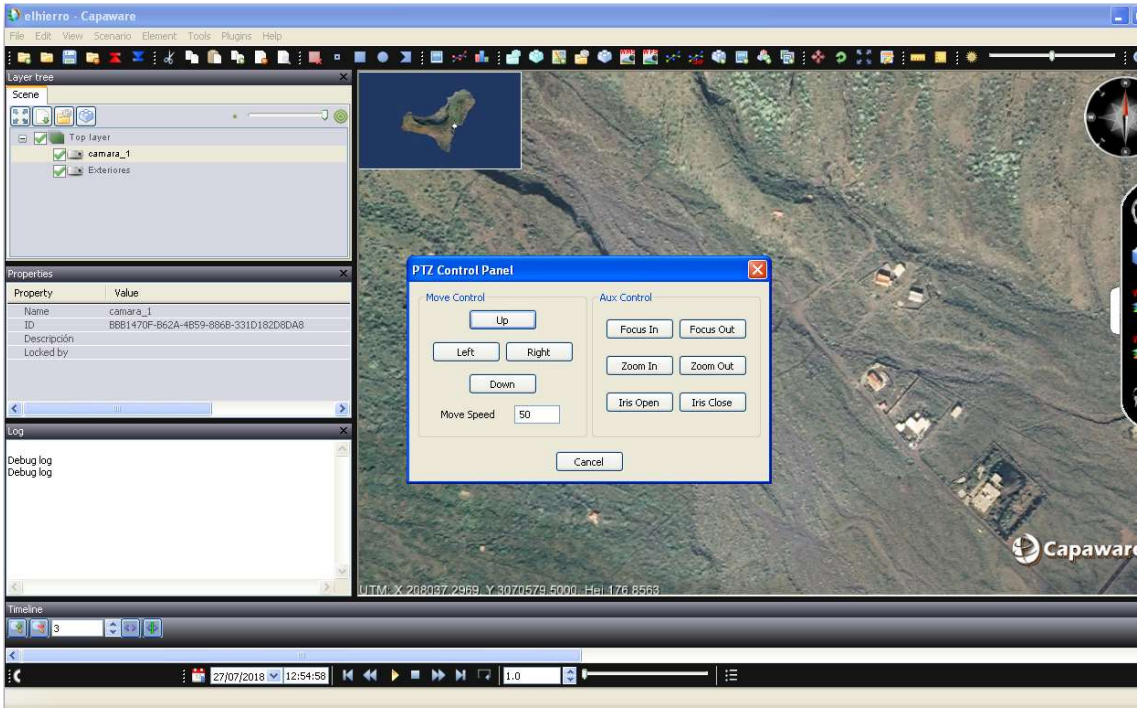

Ilustración 83: Panel de controles PTZ.

14.4.9 Mosaico de cámaras

El mosaico de cámaras requiere de una configuración previa (además de contar con al menos una cámara añadida a la escena). Nos iremos al menú *Plugins* y elegimos la opción *Set Mosaic* (ver Ilustración 84).

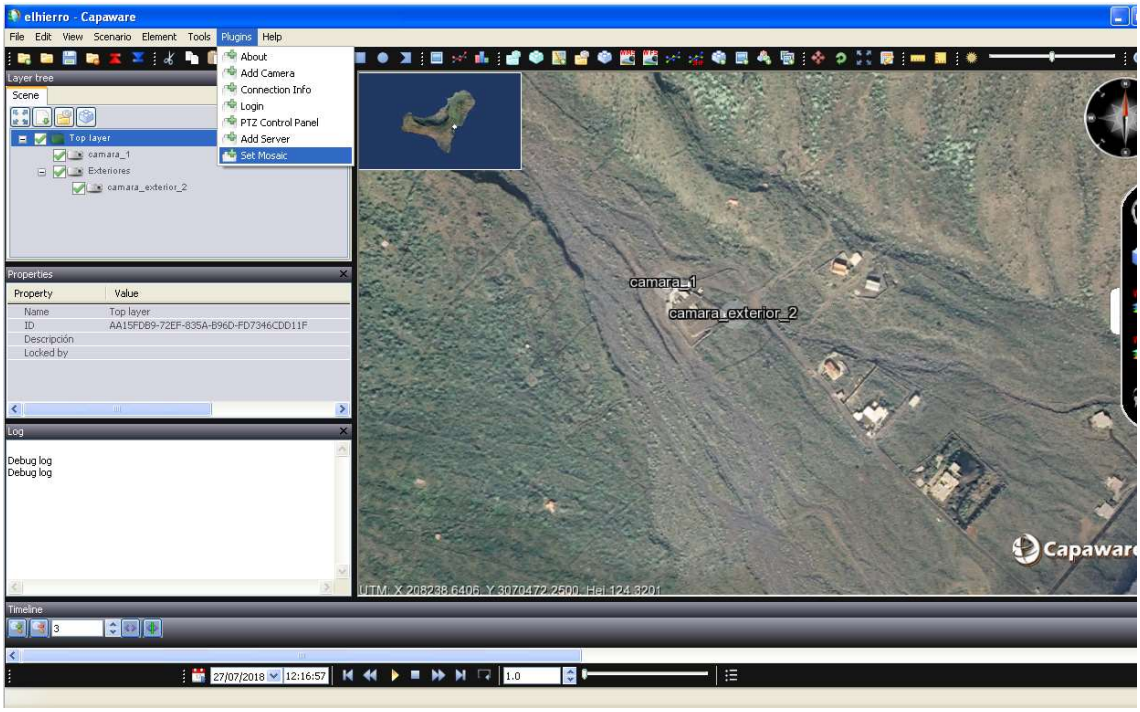

Ilustración 84: Menú Plugins – Configurar Mosaico.

Se nos abrirá una vista que consta de dos pestañas. En la primera pestaña, llamada *Camera List*, veremos dos listas: en la lista de la izquierda tendremos todas las cámaras de la escena, mientras que en la lista de la derecha tendremos las cámaras que vayamos seleccionando y pulsando en el botón Add (ver Ilustración 85).

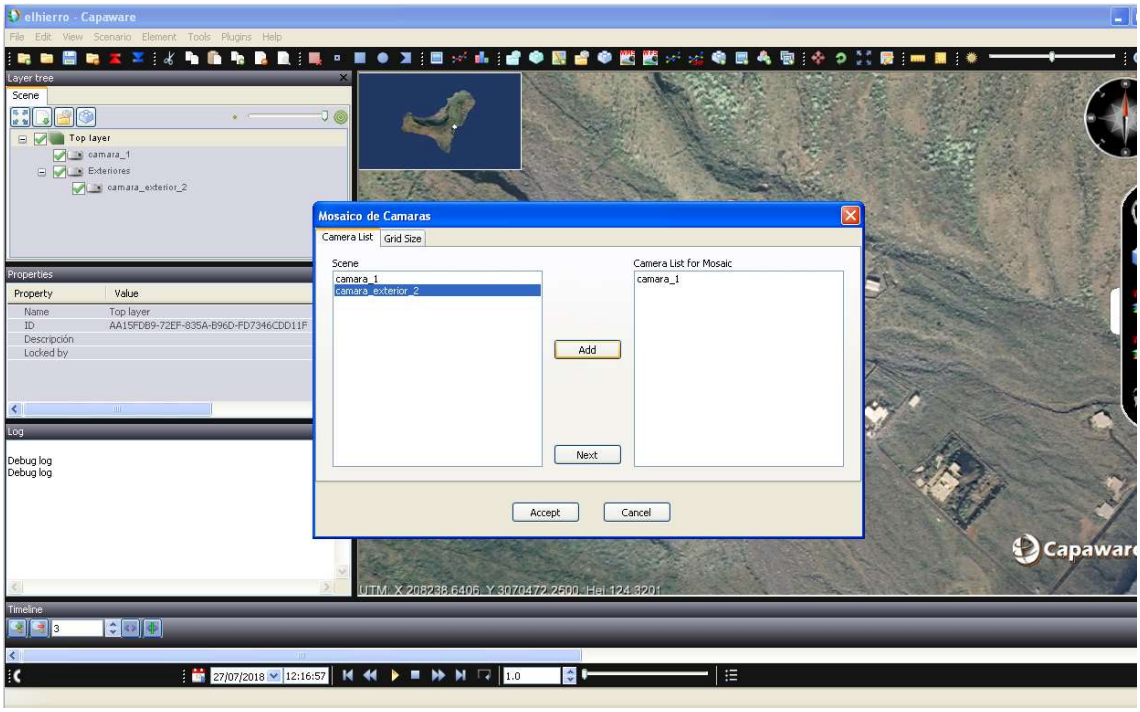

Ilustración 85: Configurar Mosaico – Listas de cámaras.

La segunda pestaña, llamada Grid Size, nos permitirá configurar el tamaño de la malla del mosaico, como puede verse en la siguiente imagen:

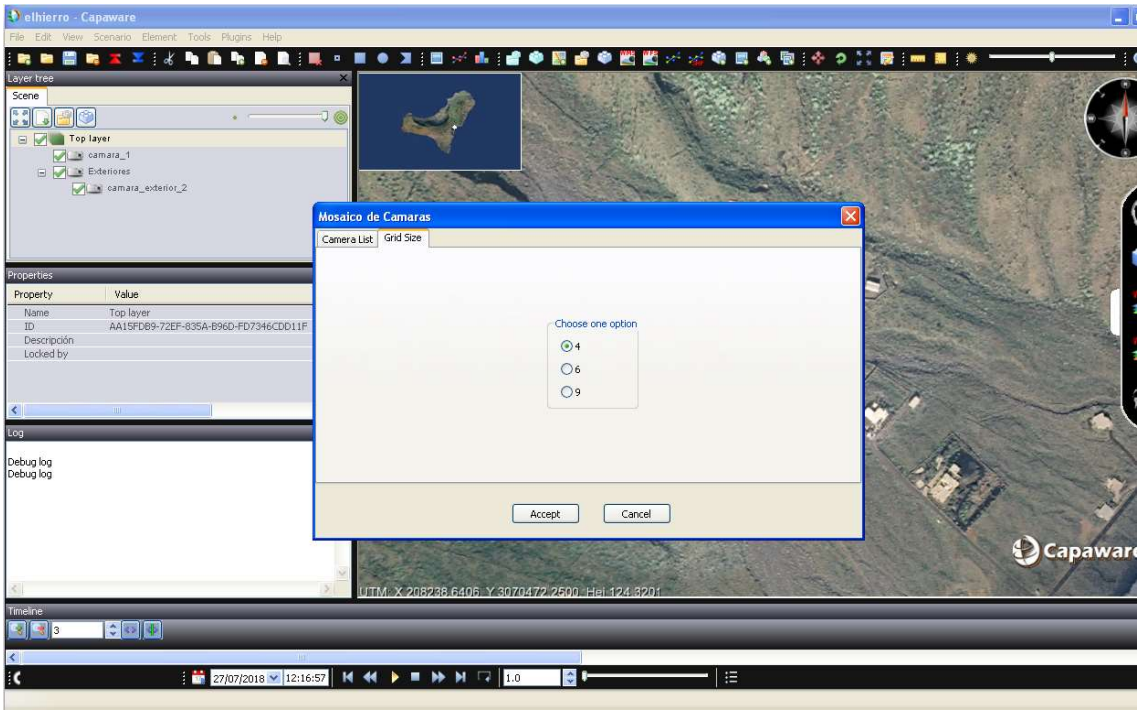

Ilustración 86: Configurar Mosaico – Tamaño de la malla.

Finalmente, pulsamos en *Accept* y se nos abrirá una subventana con las imágenes de video de las cámaras.

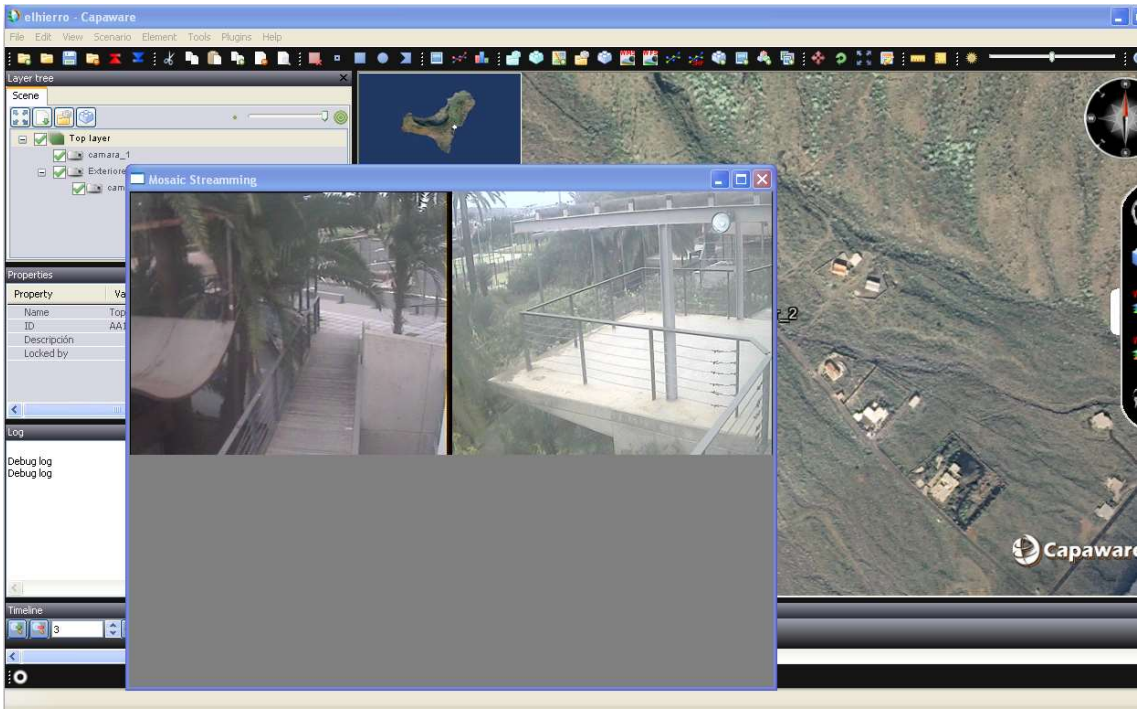

Ilustración 87: Vista de mosaico.

14.4.10 Ver video en histórico

Con una cámara añadida en la escena, y habiendo conectado, hacemos clic secundario en el árbol de la escena sobre la cámara y elegimos, dentro de Plugins, Historic Video Stream.

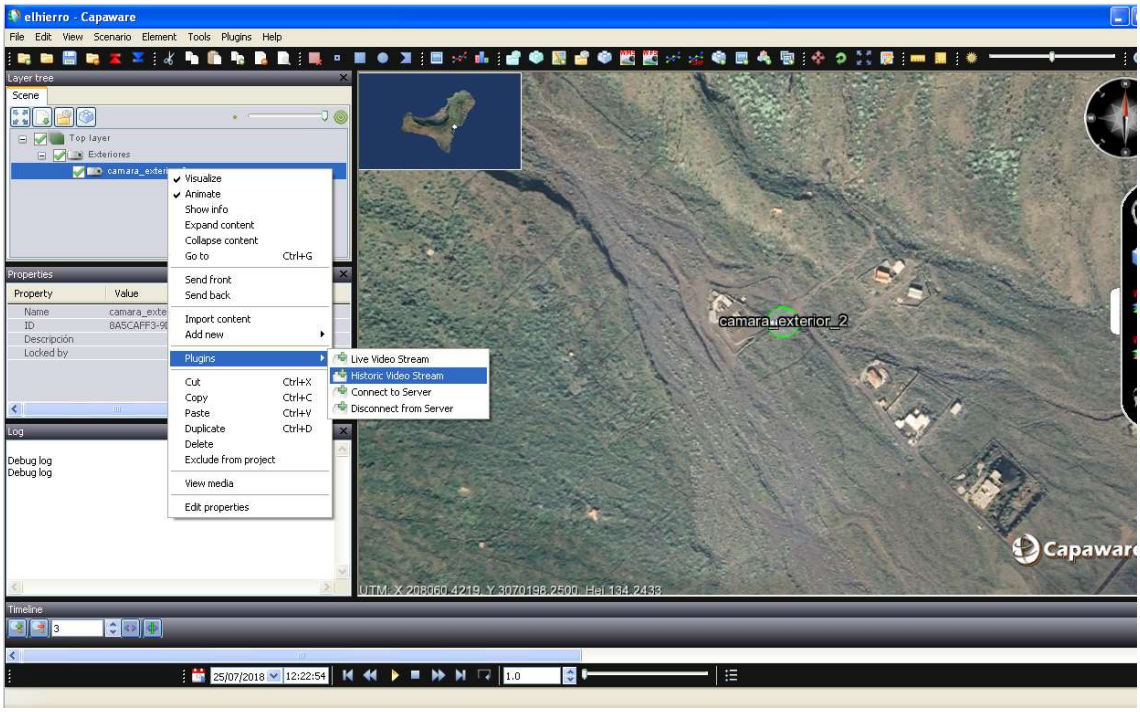

Ilustración 88: Submenú Plugins de Element3D – Historic Video Stream.

Nos aparecerá una pequeña ventana con un calendario, donde seleccionaremos la fecha que queramos mostrar, además de ir eligiendo la hora, minuto y segundo concreto, mediante los combobox.

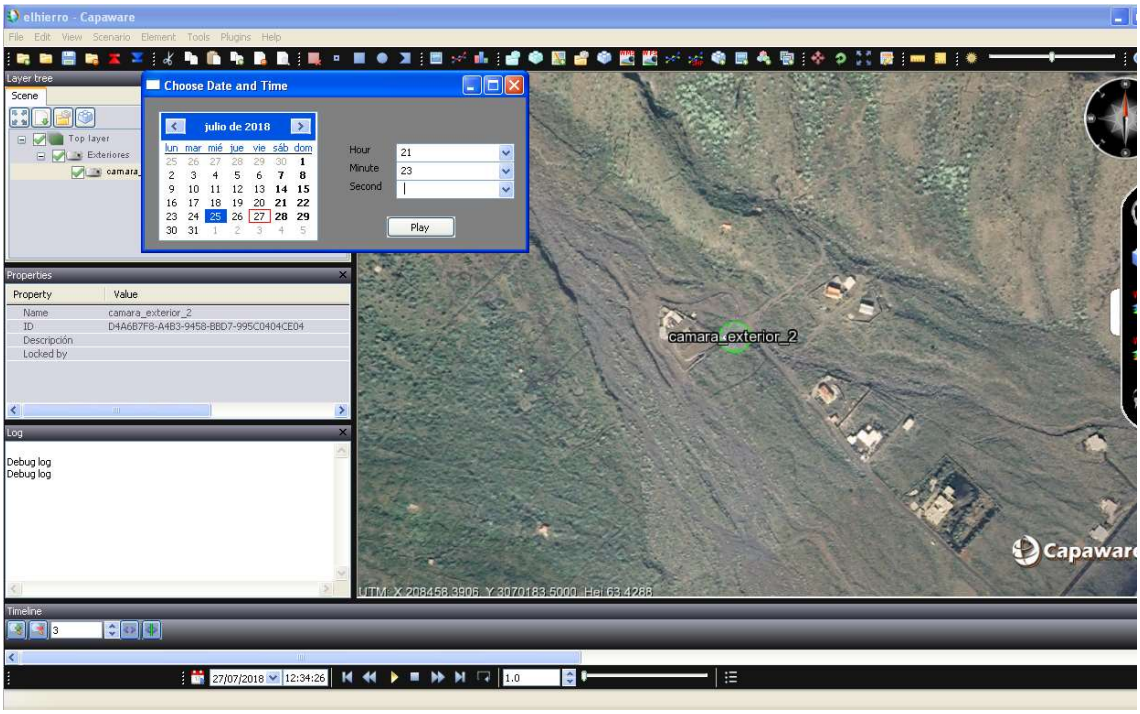

Ilustración 89: Selección de fecha y hora.

Una vez preparado el momento del tiempo, pulsamos en play y se mostrará una ventana similar a la de Live Video Stream, con la diferencia de disponer de unos controles básicos de tiempo debajo de la imagen.

Ilustración 90: Vista de histórico.

14.4.11 Menú About

La opción About dentro del menú Plugins solo contiene información sobre el título del PFC desarrollado y el nombre del Alumno, como puede verse en la siguiente imagen:

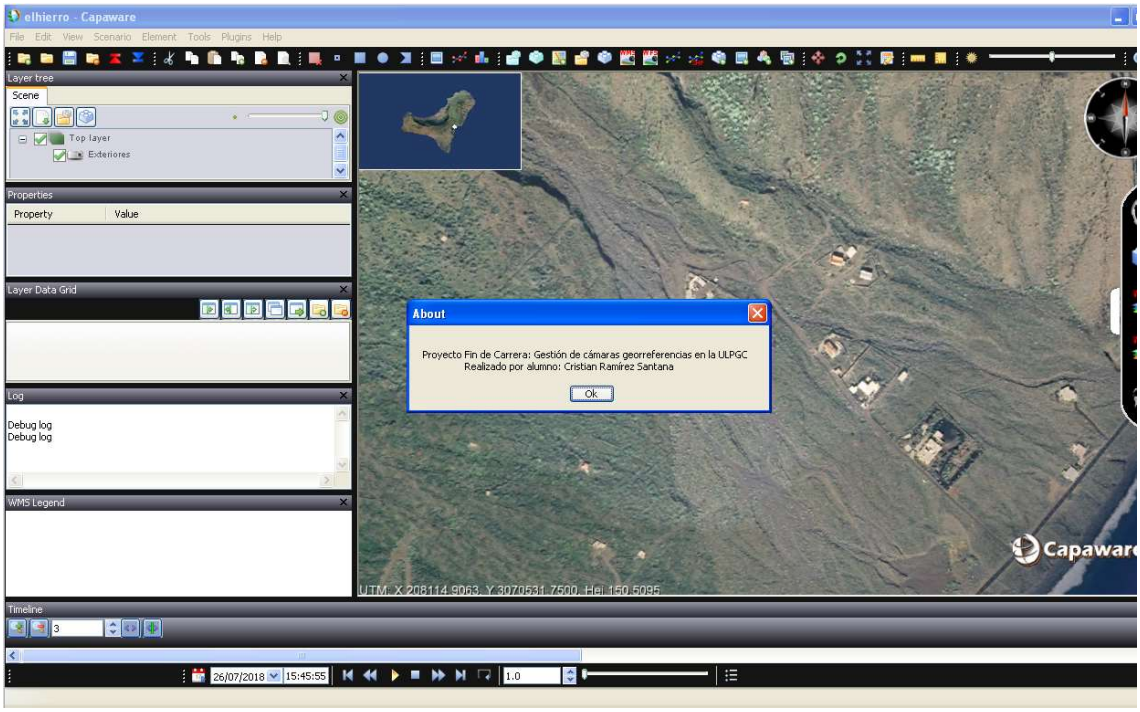

Ilustración 91: About.