UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Diseño e Implementación de un Sistema de Observación Remota en Dispositivos Móviles de Ecosistemas Biológicos para Niños Hospitalizados

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Álvaro Ponce Arévalo

Tutor: Francisco Javier Jaén Martínez

Cotutor: Fernando García Sanjuan

2014-2015

Resumen

Diversos estudios demuestran que las terapias con animales pueden ser usadas para aliviar el estrés emocional al que se ven sometidos los niños hospitalizados de larga duración. Sin embargo, las normas sanitarias de los centros pueden limitar el uso de animales en vivo en las terapias. En este trabajo se describe el análisis, diseño e implementación de HabitApp, un entorno de observación remota de animales, a través del control y visualización remotos de una cámara de vigilancia que será implantada en un zoológico real, así como a través de la reproducción en directo de recursos de vídeo en línea.

En concreto, el sistema incluye dos servidores, uno que gestiona a los usuarios y la comunicación con la cámara (tanto el acceso a la misma como las órdenes de control de movimiento) y otro que autentifica al usuario y le proporciona la información, así como una aplicación cliente para dispositivos Android que se comunica con los servidores anteriores y permite a los usuarios visualizar el vídeo y controlar el movimiento de la cámara, además de permitir a los administradores gestionar a los anteriores.

Palabras clave: Android, *Streaming*, Nodejs, Aplicación, Obervación remota, IGU (Interfaz Gráfica de Usuario).

Abstract

Several studies demonstrate that therapy involving animals can be used to relieve emotional stress that hospitalised children may suffer. However, use of live animals can be denied by the medical center. This dissertation describes the research, design and implementation of HabitApp. HabitApp is a collection of tools that provides remote viewing and control of a surveillance camera that could be placed in a zoo or animal park.

Two servers are included, one for user management and handling communication with the camera. The second provides authenticated access to the video streams. Finally there is a client for use on Android devices that communicates with both servers allowing the visualisation and control of cameras along with user administration.

Palabras clave: Android, Streaming, Nodejs, Application, Remote observation, GUI (Graphical User Interface).

Índice general

Glosario

Término	Descripción
Streaming	El *streaming* (también denominado transmisión, lectura en continuo, difusión en flujo, lectura en tránsito, difusión en continuo, descarga continua o mediaflujo) es la distribución digital de multimedia a través de una red de computadoras, de manera que el usuario consume el producto (generalmente archivo de video o audio) en paralelo mientras se descarga. La palabra *streaming* se refiere a una corriente continuada, que fluye sin interrupción.
Nonce	Normalmente es un número aleatorio utilizado en protocolos de autentificación para asegurar que las comunicaciones realizadas anteriormente no pueden ser reutilizadas de nuevo en un ataque de repetición.
Secreto	El secreto es el código o clave que se envía junto al *nonce* en una petición para asegurar las comunicaciones.
RTSP	Del inglés *Real Time Streaming Protocol*, es un protocolo de control de red que establece y controla uno o muchos flujos sincronizados de datos, ya sean de audio o de video actuando como un mando a distancia mediante la red para servidores multimedia.
Joystick	Dispositivo de control de dos o tres ejes.
Actividad	Las actividades en Android son clases públicas que representarán cada una de las pantallas de nuestra aplicación y que heredan de la clase base android.app.Activity.
Fragmento (Android)	Representa el comportamiento o una parte de la interfaz en una actividad.
Listener	Es un objeto que permanece atento al lanzamiento de determinado evento y realiza una acción cuando lo captura.

Callback	Es una función “A” que se usa como argumento de otra función “B”. Cuando se llama a “B”, ésta ejecuta “A”. Para conseguirlo, usualmente lo que se pasa a “B” es el puntero o una referencia a “A”.
API	La interfaz de programación de aplicaciones, o API (del inglés: *Application Programming Interface*), es el conjunto de subrutinas, 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.
Thread Pool	Un Thread Pool es un contenedor dentro del cual se crean y ejecutan hilos.

1. Planteamiento del problema y objetivos

1.1. Introducción

Según datos de la Encuesta de morbilidad hospitalaria 2013 del INE [1], el número de estancias hospitalarias pediátricas en España en el año 2012 ascendió a un total de 1.813.009. En particular, a modo de ejemplo de un hospital típico, esto se traduce en que anualmente se realizan en el en el Área Clínica de Enfermedades del Niño del Hospital Universitario y Politécnico La Fe (Valencia) un promedio de 54.000 estancias.

Diversos estudios psicológicos [2, 3, 4] demuestran que las estancias prolongadas en hospitales, así como las frecuentes visitas para recibir tratamientos a ciertas enfermedades crónicas, producen un aumento del nivel de estrés emocional en los niños, existiendo estudios previos [5] que han descrito además la aparición de efectos psicológicos a corto y largo plazo especialmente importantes. Estos periodos producen un profundo efecto en el modo de vida de los niños afectados los cuales deben afrontar no sólo los temores asociados a la enfermedad sino también la inseguridad derivada de la separación de sus progenitores y del entorno del hogar en el que desarrollan sus actividades rutinarias.

Por otro lado, existen antecedentes [6, 7] que demuestran que la actividad de juego permite a los niños afrontar los temores, conflictos y ansiedad derivados de la experiencia hospitalaria. En consecuencia, la mayor parte de los centros hospitalarios modernos incorporan unidades de ludo-terapia como elementos integrales de la acción hospitalaria que implementan diversas metodologías terapéuticas basadas en el juego. En este proyecto se ha decidido aplicar dichas terapias combinándolas con animales, una metodología probada eficaz [8] a la hora de intentar reducir dicho nivel de estrés y aumentar el bienestar de los niños dentro del hospital. Sin embargo, las normas sanitarias del hospital pueden limitar el uso de animales en vivo en las terapias, debiendo entonces buscar alternativas en las que los niños sientan la presencia de los animales sin tener necesariamente contacto directo con los mismos.

Es entonces donde entran en juego las superficies interactivas. La tecnología multi-táctil ha hecho grandes avances, desde los ochenta, incluso antes de la adopción de las interfaces gráficas de usuario, hasta su amplia aceptación hoy en día [9]. En la actualidad esta tecnología ofrece nuevos mecanismos de entrada y proceso que permiten a los usuarios una interacción más natural e intuitiva [10]. Esta forma de interactuar abre la puerta al desarrollo de aplicaciones para niños de muy temprana edad. Una muestra de esto la encontramos en el estudio de Rideout [11], donde se señala que los niños menores de 8 años son usuarios habituales de los medios digitales en Estados Unidos con un 38 % de ellos que ha utilizado un Smartphone, iPad o dispositivo similar al menos una vez. Dentro de este grupo el 10 % tenían menos de 23 meses de edad y el 39 % tenían entre 2 y 4 años. Además, los datos del informe Horizon [12] apoyan esta evidencia y ya se identifica a los dispositivos móviles (*Smartphones* y tabletas) como una de las tecnologías emergentes adecuadas para los niños de menos de dos años. Es por esto, por su alta popularización y accesibilidad actuales, así como su factor de forma que permite su fácil transporte, lo que convierte a estos dispositivos en unos candidatos ideales para ser implantados en entornos hospitalarios para realizar tareas ludo-terapéuticas, puesto que los niños pueden interactuar con ellos cómodamente ya sea desde la cama, desde el sillón, o sentados alrededor de una mesa.

En el siguiente trabajo se abordará el análisis, diseño e implementación de un servicio de visualización y control remotos de cámaras de vigilancia que servirá para realizar terapias con animales de manera remota aprovechando las ventajas ya mencionadas de las superficies interactivas multi-táctiles.

1.2. Trabajos relacionados

El presente trabajo de fin de grado está relacionado con dos áreas en las que se ha desarrollado una intensa actividad de investigación. Por un lado, los relacionados con el uso de las superficies interactivas para la educación infantil y, por otro, los que tienen que ver con el uso de las TICs para el fomento de la inteligencia emocional.

Superficies Interactivas y Educación Infantil

Uno de los aspectos más relevantes de este tipo de interacción es que permite la manipulación directa. Como definen Shneiderman y Plaisant [23], existen tres ideas detrás del concepto de manipulación directa: 1) la visibilidad de los objetos y las acciones de interés; 2) la sustitución de comandos escritos por acciones de señalado sobre objetos de interés; y por último, 3) acciones rápidas, reversibles e incrementales que ayudan a mantener la atención del usuario, dándole el control sobre la tecnología y evitando instrucciones complejas. En la actualidad podemos encontrar estudios interesantes que se centran en el estilo de manipulación directa

y que tienen a los niños como objetivo del estudio. Por ejemplo, Donker y otros [24] analizan si los niños de seis y siete años cometen más errores que los universitarios en tareas de arrastre y soltado (Drag & Drop) utilizando un ratón y revelan que los niños de primaria tienen menos habilidad y que la mayoría de los errores son propiciados por el tamaño del área de destino y la dirección del movimiento y no por el hecho de que se tenga que mantener el ratón pulsado. Otros trabajos se han centrado en el uso de tabletas mediante lápices electrónicos (stylus) para la manipulación directa. Por ejemplo, Terra y otros [25] con niños de 9 a 11 años y, Couse y Chen [26] con usuarios de entre 3 y 6 años, estudian si los niños son capaces de utilizar el stylus para actividades educativas. Los resultados demuestran que el periodo de aprendizaje de la interacción es relativamente corto y presenta diversas ventajas sobre los enfoques tradicionales. En esta línea encontramos el trabajo de Hourcade [27] en el que se señala que el toque directo es preferido a los dispositivos de señalado mediados como el ratón ya que el primero proporciona una forma más directa de seleccionar las opciones de la pantalla.

Por otra parte, existen diversos estudios enfocados a explorar la interacción con manipulación directa y toque directo. Por ejemplo, el trabajo de Harris y otros [28] con niños de 10 a 13 años evalúa las diferencias entre la interacción táctil simple y la múltiple y concluye que aunque el toque no afecta a la interacción en términos de equidad o frecuencia sí influye en la comunicación de los usuarios y fomenta la conversación entre ellos en sus acciones conjuntas. Kharrufa y otros [29] presentan un proceso de diseño, basado en interacción colaborativa y teorías de aprendizaje, para una aplicación de aprendizaje colaborativo sobre mesas interactivas dirigida a niños de entre 10 y 13 años. La experimentación muestra la imponente superioridad de las mesas interactivas para la creación de herramientas colaborativas de aprendizaje. Otro estudio como el de Mansor y otros [30], que compara la interacción de niños de 3 a 4 años de edad en una mesa interactiva y en una configuración física, concluye que los niños tienen dificultades para desplazar objetos sobre la superficie principalmente debido a la mala postura de los usuarios y sugiere que la interacción se haga de pie. Shoukry y otros [31] define un conjunto de guías de diseño para juegos educativos para niños.

A partir de los datos discutidos anteriormente se puede apreciar que la tecnología de superficies interactivas abre un gran abanico de posibilidades para el desarrollo de aplicaciones educativas para niños ya que ésta resuelve los problemas inherentes a otros métodos de interacción, basados por ejemplo en el ratón y el teclado. Sin embargo, como señala Ingram [12] la falta de un conjunto estandarizado y universalmente aceptado de gestos multi-táctiles hace que sea crucial el diseño de las interacciones. En particular, en el segmento de edad entre los 2 y los 10 años (usuarios típicos de los servicios de pediatría) es necesario abordar un estudio en profundidad de aspectos relativos a los tipos de interacción multi-táctil y mediante elementos tangibles que son capaces de realizar los niños desde las edades más tempranas y de los aspectos de adaptación, personalización y comunicabilidad para incrementar la efectividad y la autonomía de dichas interacciones. En este proyecto, por tanto se

diseñarás distintos mecanismos de interacción multitáctil que serán posteriormente evaluados con usuarios reales.

Inteligencia emocional y TICs

Los trabajos que relacionan inteligencia emocional y tecnología pueden dividirse en tres categorías. Por un lado, están aquellos que pretenden hacer que los ordenadores adquieran estas capacidades, lo cual es conocido como computación afectiva. Por otro, aquellos que estudian cómo la inteligencia emocional puede afectar a los seres humanos en su relación con la tecnología. Y, por último, aquellos trabajos que, a través del uso de la tecnología, pretenden conseguir que los usuarios adquieran una mayor inteligencia de este tipo.

La primera categoría queda claramente fuera de los objetivos de este proyecto dado que no pretendemos abordar la construcción de máquinas con capacidades de inteligencia emocional (computación afectiva).

Respecto a la segunda categoría, existen diversos trabajos que tratan de estudiar el impacto de la inteligencia emocional en el aprendizaje online. En este sentido, Berenson y otros [32] estudian una serie de predictores de la nota media entre los alumnos de una universidad pública de ámbito municipal estadounidense, demostrando que aquellos que están basados en la inteligencia emocional son los más efectivos en la predicción. Han y Johnson [33] estudian la relación entre la inteligencia emocional y las interacciones en una plataforma de aprendizaje en línea por parte de alumnos de máster universitario. Entre sus resultados, destacan que los alumnos con mayor inteligencia emocional se sienten más unidos socialmente con otros, y que también aquellos usuarios con mayor capacidad para percibir emociones a través de expresiones faciales interaccionan menos entre ellos vía mensajería síncrona de texto. Por otro lado, Al-Faouri [34] estudia el impacto de la inteligencia emocional en el proceso de aprendizaje tecnológico por parte de los empleados de 10 instituciones empresariales. Según sus resultados, aquellos empleados con un mayor nivel en ciertas dimensiones de inteligencia emocional presentaban una mayor capacidad de aprendizaje tecnológico. Sin embargo, sus resultados deben tomarse con cierta precaución ya que se basan únicamente en cuestionarios de autoevaluación. Estos trabajos tienen en común que todos muestran la inteligencia emocional como una característica positiva hacia el uso de tecnología. El trabajo de Kumar y otros [35], sin embargo, es un ejemplo contrario en este sentido. Estos investigadores estudian la actitud de ciertos estudiantes hacia el uso de ordenadores. Miden tres componentes: actitud afectiva (las emociones o sentimiento que los usuarios tienen hacia los ordenadores), actitud conativa (el comportamiento que manifiestan los usuarios hacia los ordenadores) y actitud cognitiva (el conocimiento de los ordenadores por parte de los estudiantes). Sus resultados muestran que no hay una relación significativa entre estas actitudes y el nivel de inteligencia emocional de los estudiantes. Estos estudios son de un gran interés en nuestro contexto, pero ninguno de ellos

aborda la relación entre este tipo de inteligencia en la edad infantil ni tampoco en un contexto hostil como el hospitalario donde el factor estrés emocional tiene una presencia muy importante. Es por ello que es necesario estudiar en este contexto si aparecen nuevas formas de relación entre tecnología e inteligencia emocional cuando se trata con sujetos en edad infantil bajo condiciones afectivas desfavorables como las que se dan en una situación de hospitalización.

Por último, existen trabajos que estudian la tercera categoría enunciada anteriormente. Esto es, el impacto de la tecnología en el aprendizaje de características relacionadas con la inteligencia emocional. En esta dirección, hay enfoques simplistas como el de Bresó y otros [36] que abordan el problema diseñando aplicaciones que sirven como test de inteligencia emocional para que el sujeto pueda tener un cierto grado de conocimiento de cuál es su nivel de inteligencia emocional, pero sin plantear técnicas que permitan a éstos adquirir aquellas capacidades emocionales de las que carecen. En cambio, hay otros trabajos como Lee y otros [37] que plantean el uso de la tecnología como forma de obtención de capacidades de inteligencia emocional como factor de mejora de las capacidades creativas y del rendimiento del trabajo de desarrolladores de sistemas de información. También cabe destacar el trabajo de Mazzone y otros [38] centrado en el colectivo de personas adolescentes y que diseñan un videojuego como producto de aprendizaje electrónico que pretende mejorar la inteligencia emocional de los mismos. Este juego aborda las cuatro habilidades definidas por Mayer y otros [39] (percibir, asimilar, entender y regular emociones), involucrando a los adolescentes en el diseño del juego. De esta forma, los jugadores deben, ya sea de forma individual o colectiva, etiquetar emociones, entender qué representan, usarlas y gestionarlas. De forma similar destaca el trabajo de Morris y otros [40] en el que involucran colectivamente a varios usuarios alrededor de grandes pantallas interactivas para que clasifiquen imágenes de acuerdo a las emociones que les suscitan, utilizando colores y sonidos. Durante la ejecución de las actividades, los autores observan que los usuarios hablan entre ellos de sus sentimientos hacia las imágenes, y también, antes de etiquetar una imagen de ellos mismos, auto-reflexionan y se preguntan cosas como: "¿Estoy triste o contento?". Según estos investigadores, sus exploraciones muestran el deseo de la gente por tener tecnologías que permitan la expresión compleja de las emociones en forma de juego.

Otros trabajos abordan el uso de la tecnología como elemento de adquisición de capacidades emocionales en sujetos con discapacidades cognitivas y muestran la potencialidad del uso de la tecnología para conseguir avances en el desarrollo social y emocional [41]. Existen numerosos trabajos en este sentido en relación a sujetos con trastornos del espectro autista (TEA). Según Hourcade y otros [42], el uso de tecnologías puede ayudar a aprender cómo funciona la mente de niños con TEA y cómo se relacionan con el mundo que los rodea. Además, afirman también que el uso de la tecnología en sí puede ser incentivo suficiente para mejorar la calidad de las interacciones sociales. El objetivo de la mayoría de los trabajos que relacionan tecnología con TEA es, principalmente, utilizar los dispositivos como profesores virtuales o conseguir aumentar el grado de socialización de niños con estos

trastornos, y no tanto hacerles razonar sobre las emociones para ayudarles a superar una situación personal desfavorable. Diversos estudios (p.ej., [43]) muestran cómo los niños con estos trastornos se sienten más cómodos interactuando con máquinas que con personas. Por esta razón, los ordenadores resultan idóneos para realizar tareas de aprendizaje con estos niños. Baldi [44] es un tutor animado por ordenador cuya finalidad es enseñar vocabulario y gramática a niños con autismo a través de la identificación de imágenes y de la producción oral de palabras. Davis y otros [45] presentan un software interactivo llamado TouchStory diseñado para identificar qué aspectos narrativos resultan más complejos para niños con autismo y a la vez mejorar su comprensión narrativa. El uso de ordenadores, sin embargo, limita en gran medida las capacidades colaborativas y sociales de los niños, puesto que la interacción con los mismos se ve restringida normalmente a un solo usuario. Las superficies interactivas permiten, por su propia naturaleza, dar soporte a actividades colaborativas. Por tanto, este tipo de actividades pueden ayudar a los niños con TEA a aumentar su nivel de socialización con otros niños. El proyecto ECHOES [46] pretende dar soporte a niños de 5-7 años con TEA en la exploración y adquisición de habilidades sociales a través de la interacción de los mismos con un entorno tecnológico. Los niños interactúan a través de una pantalla multitáctil con un agente virtual semi-inteligente en situaciones socialmente realistas. El entorno se compone también de tres cámaras que permiten detectar la posición y la rotación de la cabeza de los usuarios, así como también leer sus expresiones faciales y detectar hacia dónde dirigen la mirada. Gracias a esta información, el sistema puede adaptarse al estado emocional y detectar el foco de atención de los niños. Piper y otros [47] diseñan un juego colaborativo sobre mesas interactivas para niños con síndrome de Asperger, en el que pueden jugar cuatro a la vez. Los autores concluyen que este tipo de juegos facilitan el trabajo en equipo de los niños, y que la tecnología utilizada mantiene la motivación de los niños constantemente. Gal y otros [48] también hacen uso de una mesa interactiva para realizar tareas que promuevan la colaboración entre niños con TEA mientras cuentan historias. Los investigadores observan un aumento en el número de acercamientos entre los niños, una mayor carga de afecto positivo en éstos así como una mayor tendencia a expresar emociones.

Del conjunto de trabajos presentados anteriormente se deduce que, o bien el público objetivo de los estudios son adultos o adolescentes que participan en actividades de identificación (que no de puesta en práctica para la resolución de problemas) de características relacionadas con la inteligencia emocional, o bien son niños con trastornos del espectro autista con necesidades terapéuticas muy específicas. En este trabajo de fin de grado, en cambio, se intenta trabajar en un enfoque general para niños, en su mayor parte no afectados por trastornos de carácter emocional, que permita el aprendizaje de capacidades creativas y emocionales como dos elementos simbióticos como estrategia terapéutica de mitigación del estrés emocional asociado a procesos de hospitalización y de sus posibles secuelas posteriores.

1.3. Objetivos

En este proyecto se presenta un sistema compuesto por dos servidores y una aplicación. Mediante los servidores se ha proporcionado un servicio de gestión de usuarios para que un administrador sea capaz de controlar a dichos usuarios en todo momento, así como un sistema de autentificación junto con un proveedor de contenidos. Se ha creado HabitApp, una aplicación para Android desde la que poder ver y gestionar a los usuarios y a las cámaras de manera cómoda en el caso de un administrador, o de simplemente ver *streamings* de vídeo y controlar las cámaras cuando se posea el control en el caso de ser un usuario normal. Además, la aplicación provee otros servicios de *streaming* aparte de los de las cámaras de vigilancia.

Los objetivos del proyecto se resumen en los siguientes:

- Implementación de un servidor para la gestión de usuarios, donde se permita a varios dispositivos cliente tener acceso al *streaming* de vídeo pero se restrinja el control de la cámara a uno solo simultáneamente.
- Implementación de un mecanismo de gestión de usuarios que permita a un usuario administrador controlar qué usuario puede tener el control de la cámara en un momento determinado.
- Implementación de un modo automático de gestión de usuarios para que ante la ausencia de un administrador estos puedan mover las cámaras, y sea el propio sistema quien gestione qué usuario tiene el control.
- Implementación de un servidor de contenidos que además de la información de las cámaras provea de un servicio de autentificación para identificar a cada usuario dentro del sistema y así facilitar la gestión de los mismos por parte de los administradores.
- Comunicación entre el servidor y la cámara para el envío de las órdenes de movimiento de la segunda.
- Implementación de una aplicación cliente para dispositivos Android que reproduzca el *streaming* de vídeo procedente de la cámara con el mínimo retardo posible, además de permitir enviar comandos de movimiento al servidor de manera rápida y fluida para conseguir una buena experiencia de usuario.
- Implementación de un servicio adicional de reproducción de *streamings* de Youtube para los casos en los que las cámaras de vigilancia no estén disponibles.

En la figura **1.1** se puede observar cómo la aplicación interactúa con los dos servidores, uno que se hará cargo de proporcionar los contenidos y autentificar a los usuarios, y otro para la gestión y control de las cámaras propias:

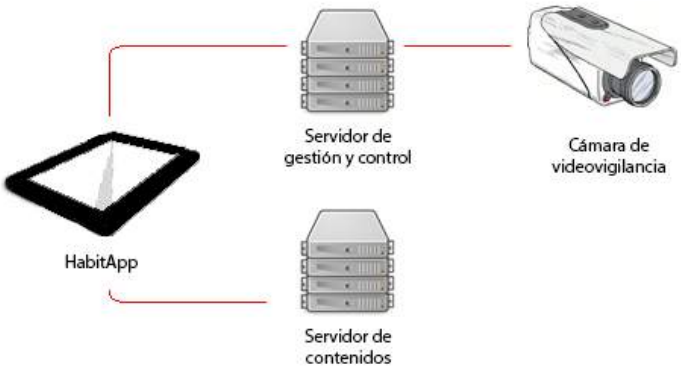

Figura 1.1: Componentes hardware del sistema

El resto del documento está organizado como sigue: En el capítulo 2 se habla sobre la aplicación HabitApp. A continuación, en el capítulo 3 se describen los servidores, para seguir con el capítulo 4 hablando sobre seguridad. Por último, se terminará con las conclusiónes y el trabajo futuro.

2. Aplicación

2.1. Enumeración de requisitos funcionales

En este apartado se expondrán los requisitos del producto de manera breve y agrupados en 4 grupos diferentes, para ser vistos con más detalle en el punto **2.3**.

Requisitos para la autentificación en la aplicación:

- **Iniciar sesión como administrador:** Un usuario debe ser capaz de autentificarse como administrador con el objetivo de realizar las tareas de gestión y control tanto de los usuarios como de las cámaras.
- **Iniciar sesión como usuario:** En caso de que se vaya a usar la aplicación con fines meramente lúdicos o didácticos, un usuario debe poder autentificarse sin la necesidad de hacerlo como administrador.

Requisitos para la visualización de un streaming:

- **Obtener listado de cámaras:** Debe ser posible recuperar una lista de *streamings* del servidor, ya sean de Youtube o de cámaras remotas.
- **Visualizar *streaming* de Youtube:** Un usuario debe ser capaz de reproducir un *streaming* alojado en Youtube.
- **Visualizar *streaming* de una cámara remota:** Un usuario debe ser capaz de reproducir el *streamings* de vídeo de una determinada cámara remota.

Requisitos para el control de las cámaras:

- **Mover cámara:** Si el usuario tiene el control de la cámara a la que está conectado, este debe ser capaz de controlarla mediante los controles que aparecen en la interfaz.

- **Ver lista de usuarios conectados:** Si un administrador ha activado el modo de gestión automático, un usuario normal con el control debe poder conocer los dispositivos que hay conectados a la misma cámara que él.
- **Ceder el control a otro usuario:** Si un usuario normal tiene el control y está el modo de gestión en automático, éste debe poder seleccionar a otro usuario que esté visualizando su misma cámara y otorgarle el control.

Requisitos para la gestión de usuarios:

- **Ver usuarios conectados a una cámara:** Si la cámara a la que se está conectado es teleoperada, el administrador deberá ser capaz de ver a los usuarios conectados a la misma, además de quien tiene el control en ese momento.
- **Dar/Quitar el control de la cámara a un usuario:** El administrador debe poder seleccionar a cualquier usuario para otorgarle o revocarle el control de una determinada cámara.
- **Cambiar modo de gestión de usuarios:** El administrador puede poner este modo en automático para no tener que ser el que tenga que gestionar el control de la cámara entre los usuarios en caso de que no pueda hacerse cargo él mismo, haciendo que el control pase a ser gestionado por el servidor y el resto de los usuarios.

2.2. Características de los actores y perfiles de los usuarios

Usuario no autentificado: Este usuario será el que no haya iniciado sesión todavía. Lo único que puede hacer es intentar autentificarse.

Usuario autentificado: El usuario autentificado es el cual ha conseguido iniciar sesión satisfactoriamente. Puede realizar cualquier acción relacionada con la reproducción de un *streaming* de vídeo.

Usuario con control: El usuario que tiene el control hereda las funciones que puede realizar del usuario autentificado. Además de ser capaz de seleccionar o reproducir un *streaming* de vídeo determinado por ejemplo, puede tanto mover la cámara del mismo como cambiar el modo de control o el tipo de desplazamiento.

Usuario administrador: Hereda del usuario autentificado en el sentido de que puede hacer lo mismo que él, además de todo lo relacionado con la gestión de los usuarios.

Usuario administrador con control: Este tipo de usuario hereda del usuario administrador y del usuario con control, pudiendo realizar todas las funciones de ambos.

Usuario normal con control: El usuario normal con control puede realizar todas las funciones del usuario con control. Se hace esta diferenciación del administrador con control debido a que el usuario normal con control puede ceder el control de la cámara a otros usuarios.

Servidor de contenidos: El servidor de contenidos es el actor encargado tanto de validar los inicios de sesión como de proveer el listado de *streamings* de vídeo que podrá seleccionar el usuario.

Servidor de gestión y control de cámaras: Es el actor encargado de gestionar todos los comandos de movimiento de una cámara, al igual que de hacerse cargo de otorgar y revocar el control de la misma según dicte un administrador o un usuario normal con control.

Cámara: Es el encargado de proveer el *streaming* de vídeo de una determinada cámara.

Youtube: Es el encargado de proveer los *streamings* de vídeo alojados en su servicio.

2.3. Especificación de requisitos funcionales

En el siguiente apartado se especifican los requisitos funcionales identificados en la sección **2.1**. Para cada uno de los diferentes grupos se comenzará con un diagrama de casos de uso, seguido de una descripción del mismo, la entrada, y cómo este debe reaccionar a ella. Seguidamente se mencionarán las restricciones, para acabar con una breve explicación de cómo funciona.

2.3.1. Requisitos para la autentificación en la aplicación:

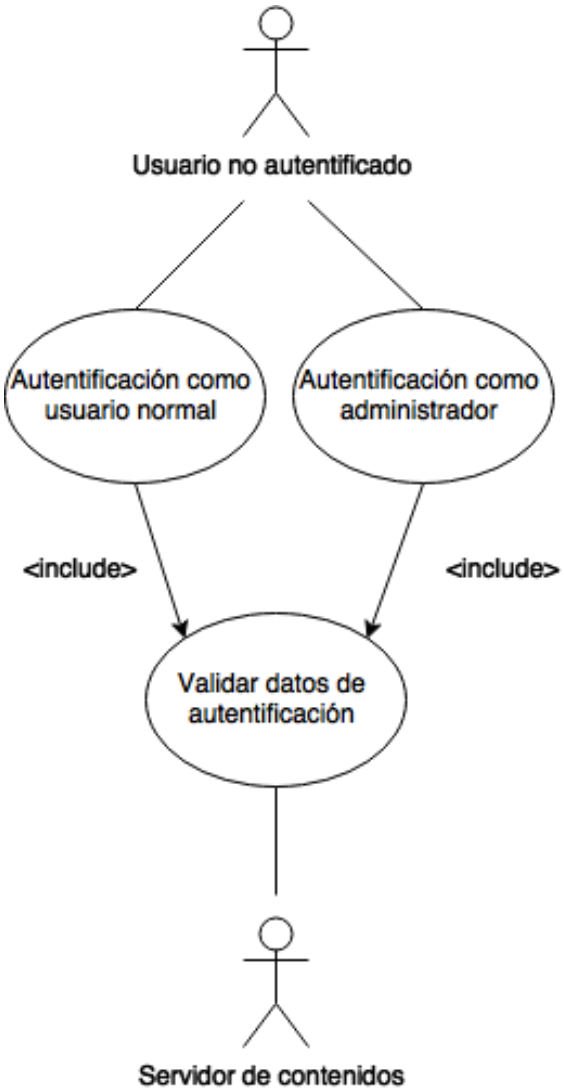

Figura 2.1: Diagrama de casos de uso para la autentificación

- Caso de uso 1.1: Autentificación como administrador:

DESCRIPCIÓN

Una vez el usuario entre en la pantalla de login este podrá iniciar sesión como administrador introduciendo un nombre de usuario y una contraseña.

ENTRADA

Un nombre de usuario y una contraseña.

SALIDA

Se pueden dar 3 casos:

- **No se ha podido realizar la petición o esta ha fallado:** En este caso se muestra una alerta que informa al usuario de que no se ha podido realizar la petición debido posiblemente a una mala conexión de internet o a que el servidor no está disponible.
- **Se ha podido realizar la petición satisfactoriamente:** Cuando esto sucede pueden darse dos situaciones:
  - **El nombre del usuario y la contraseña son correctos:** Se permite al usuario autentificarse como administrador y se notifica a la aplicación. Además, pasa a ser un "usuario administrador".
  - **No se ha insertado una combinación nombre de usuario/contraseña correcta:** El servidor devuelve un código de error y se muestra una alerta informando al usuario.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

Cuando se inserta un nombre de usuario y su correspondiente contraseña se hace una petición al servidor para obtener un *nonce*. Posteriormente, se realizará otra petición con el secreto (que estará guardado localmente en la aplicación) y el *nonce* combinados, junto con los datos de la autentificación (nombre de usuario y contraseña). Si la combinación del *nonce* y el secreto es correcta, se pasará

a validar la autentificación, y si no lo es simplemente se ignorará la petición. Cabe destacar que la cadena de texto formada por la combinación del *nonce* y el secreto estará cifrada. Hablaremos de ellos con más detalle en el apartado de seguridad (Sección **4**).

ACTORES QUE INTERVIENEN

- Usuario no autentificado.

- Caso de uso 1.2: Autentificación como usuario normal:

DESCRIPCIÓN

A diferencia del inicio de sesión como administrador, el usuario en este caso sólo necesitará ingresar un nombre de usuario para identificarse.

ENTRADA

Un nombre de usuario.

SALIDA

Se pueden dar 2 casos:

- **No se ha podido realizar la petición o esta ha fallado:** De la misma manera que con el inicio de sesión como administrador, se muestra una alerta que informa al usuario de que no se ha podido realizar la petición debido posiblemente a una mala conexión de internet o a que el servidor no está disponible.
- **Se ha podido realizar la petición satisfactoriamente:** Dado que no hay contraseña que revisar, se autentifica correctamente al usuario y se notifica a la aplicación se realiza una transición a la siguiente ventana. Además, pasa a ser un "usuario autentificado"

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

Se usa el mismo método de inicio de sesión que con el de administrador, simplemente con un campo menos (el de contraseña), por lo que se autentifica automáticamente al usuario en el caso de que no haya ningún problema con el *nonce* y el secreto.

ACTORES QUE INTERVIENEN

- Usuario no autentificado.

- Caso de uso 1.3: Validar datos de autentificación:

DESCRIPCIÓN

Cuando un usuario se autentifica el servidor debe ser capaz de validarlo, de manera que este último comunique a la aplicación si los datos con los que se ha intentado autentificar el usuario han sido correctos o no.

ENTRADA

Dependiendo del tipo de autentificación pueden darse dos entradas:

- **Autentificación como administrador:** La entrada estará compuesta por un nombre de usuario y una contraseña, además de un secreto y un *nonce* todos encriptados.
- **Autentificación como usuario normal:** La entrada estará formada por un nombre de usuario, además de un secreto y un *nonce* todos encriptados.

SALIDA

Se pueden dar 2 casos:

- **Los datos de inicio de sesión son incorrectos:** Se envía un código de error a la aplicación informando de que los datos ingresados son incorrectos.
- **Los datos de inicio de sesión son correctos:** Se notifica a la aplicación de que los datos eran correctos y se valida la autentificación.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

Cuando el servidor recibe la petición de inicio de sesión pasa a comprobar que tanto el *nonce* como el secreto son correctos. Estos están encriptados en una cadena de texto, la cual el servidor se encarga de desencriptar y separar. Si ambos son válidos se pasa a comprobar los datos de inicio de sesión, en cuyo caso puede darse cualquiera de las dos salidas previamente descritas después de desencriptar dichos datos. En caso contrario, se ignorará por completo la petición.

ACTORES QUE INTERVIENEN

- Servidor de contenidos.

2.3.2. Requisitos para la visualización de un streaming:

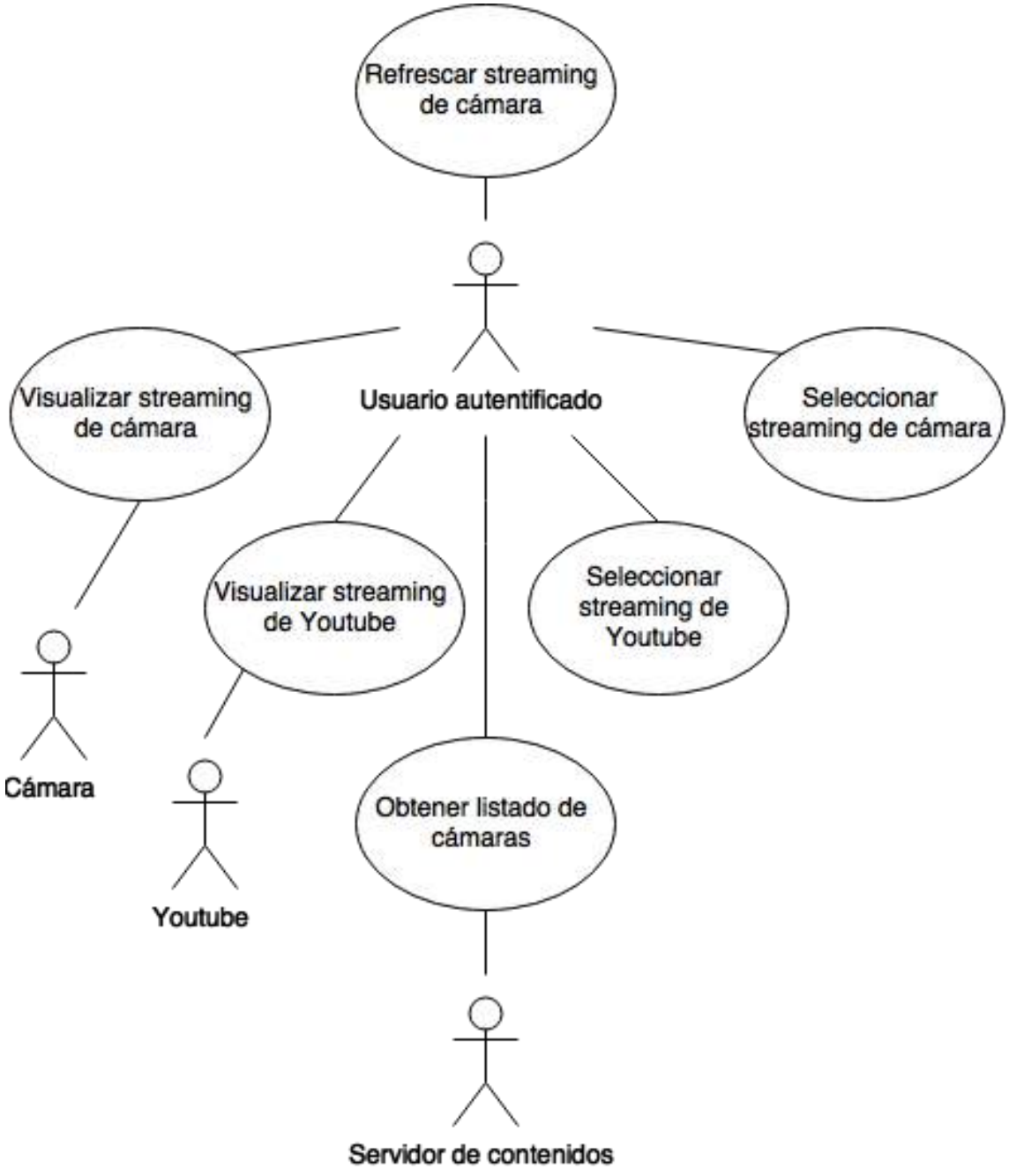

Figura 2.2: Diagrama de casos de uso de visualización de un *streaming*

- Caso de uso 2.1: Obtener listado de cámaras:

DESCRIPCIÓN

Una vez el usuario se ha autentificado, este debe ser capaz de elegir un *streaming* de vídeo de entre una lista. Nuestra aplicación debe poder obtener esa lista mediante el servidor de contenidos, el cual se hace cargo de proveerla mediante un archivo JSON guardado en el directorio del servidor.

ENTRADA

No tiene ninguna entrada.

SALIDA

Un archivo JSON con el listado de *streamings*, tanto de cámaras remotas como de retransmisiones en Youtube.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

Cuando el usuario se autentifica la aplicación manda una petición al servidor de contenidos, el cual recupera el archivo con el listado de *streamings* del directorio y se lo envía a la aplicación. Una vez la aplicación tiene dicho archivo, se encarga de procesarlo y cargar la vista con la lista multinivel que usarán los usuarios para seleccionar un determinado *streaming* de vídeo.

ACTORES QUE INTERVIENEN

- Usuario autentificado.
- Servidor de contenidos

- Caso de uso 2.2: Seleccionar *streaming* de Youtube:

DESCRIPCIÓN

El usuario dispondrá de una lista multinivel en la que podrá seleccionar un *streaming* de vídeo. Podrá interactuar con ella desplegando o plegando los distintos niveles y presionando en el *stream* que quiera reproducir, que en este caso será uno alojado en Youtube.

ENTRADA

URL del *streaming* de Youtube seleccionado por el usuario.

SALIDA

Se cargará el reproductor de Youtube en la ventana principal con el *streaming* de vídeo que se haya seleccionado.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

Cuando el usuario elige un *streaming* de Youtube, la aplicación carga el fragmento que implementa el reproductor que hace uso de la API de Youtube en la pantalla principal. El fragmento toma como parámetros la url del *streaming* y lo prepara para su reproducción.

ACTORES QUE INTERVIENEN

- Usuario autentificado.

- Caso de uso 2.3: Seleccionar *streaming* de cámara remota:

DESCRIPCIÓN

De la misma manera que el usuario puede seleccionar un *streaming* de Youtube de la lista multinivel, también puede elegir otro que pertenezca a una cámara remota.

ENTRADA

URL del *streaming* de la cámara remota seleccionada por el usuario.

SALIDA

Se pueden dar 2 casos diferentes:

- **El modo de gestión automático está desactivado o está activado y otro usuario tiene el control:** Se cargará la interfaz con el reproductor de vídeo.
- **El modo de gestión automático está activado y no hay ningún usuario con el control:** Se cargará el reproductor de vídeo junto con los controles de la cámara.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

En el momento que se presiona una de las celdas que representan un *streaming* de una cámara remota, el fragmento que contiene el reproductor de *streamings* RTSP es cargado en la pantalla principal. Al igual que con el de Youtube se pasa como parámetro la URL, pero en este caso del *streaming* de una cámara.

ACTORES QUE INTERVIENEN

- Usuario autentificado.

- Caso de uso 2.4: Refrescar *streaming* de cámara:

DESCRIPCIÓN

Esta funcionalidad permite al usuario reiniciar el reproductor de vídeo. Dado que son *streamings* en directo puede darse el caso de una desconexión por parte del emisor. En el supuesto de que haya ocurrido algún problema de línea con la *tablet* esta dispone de medios para recuperar el vídeo, pero si son causas ajenas al dispositivo el botón de *refresh* es la solución para volver a resumir la reproducción del *stream*.

ENTRADA

No tiene ninguna entrada.

SALIDA

El reproductor de vídeo se reinicia, intentando volver a reproducir el *streaming*.

RESTRICCIONES

- Haber seleccionado un *streaming* de la cámara móvil, ya que esta característica no está disponible con los alojados en Youtube.

FUNCIONAMIENTO

Para reiniciar el vídeo se pulsa el botón que representa dicho comando. Este botón hace dos llamadas al objeto que representa el reproductor de vídeo, que son *"Pause"* y *"Play"*. Simplemente pausando el vídeo y volviendo a reproducirlo reiniciamos el reproductor consiguiendo que ante cualquier bloqueo este pueda recuperarse y vuelva a mostrar el *streaming* en caso de estar disponible y disponer de una conexión estable.

ACTORES QUE INTERVIENEN

- Usuario autentificado.

- Caso de uso 2.5: Visualizar *streaming* de Youtube:

DESCRIPCIÓN

El usuario es capaz de visualizar el *streaming* de vídeo una vez se ha cargado el reproductor de Youtube alojado en el fragmento insertado en la pantalla principal. Dicho reproductor ofrece la posibilidad de reproducir la retransmisión pulsando el botón de play.

ENTRADA

No tiene ninguna entrada.

SALIDA

Se comenzará a reproducir el vídeo en la pantalla principal.

RESTRICCIONES

- Haber seleccionado una retransmisión de Youtube.

FUNCIONAMIENTO

Cuando el reproductor de Youtube alojado en el fragmento ha terminado de cargar, el usuario puede pulsar el botón de play, el cual hace que se comience a reproducir el vídeo.

ACTORES QUE INTERVIENEN

- Usuario autentificado.
- Youtube.

- Caso de uso 2.6: Visualizar *streaming* de cámara remota:

DESCRIPCIÓN

A diferencia del fragmento que gestiona el reproductor de Youtube, este comienza la reproducción del vídeo automáticamente ha terminado de cargarse, por lo que el usuario no debe presionar ninguna clase de botón para comenzar a reproducir el vídeo.

ENTRADA

No tiene ninguna entrada.

SALIDA

Se comienza a reproducir la retransmisión de la cámara remota seleccionada en la pantalla principal.

RESTRICCIONES

- Haber seleccionado la retransmisión de una cámara remota.

FUNCIONAMIENTO

Cuando el fragmento que gestiona el reproductor de una cámara remota ha terminado de cargar, este comienza la reproducción de la retransmisión de la cámara seleccionada automáticamente.

ACTORES QUE INTERVIENEN

- Usuario autentificado.
- Cámara.

2.3.3. Requisitos para el control de las cámaras:

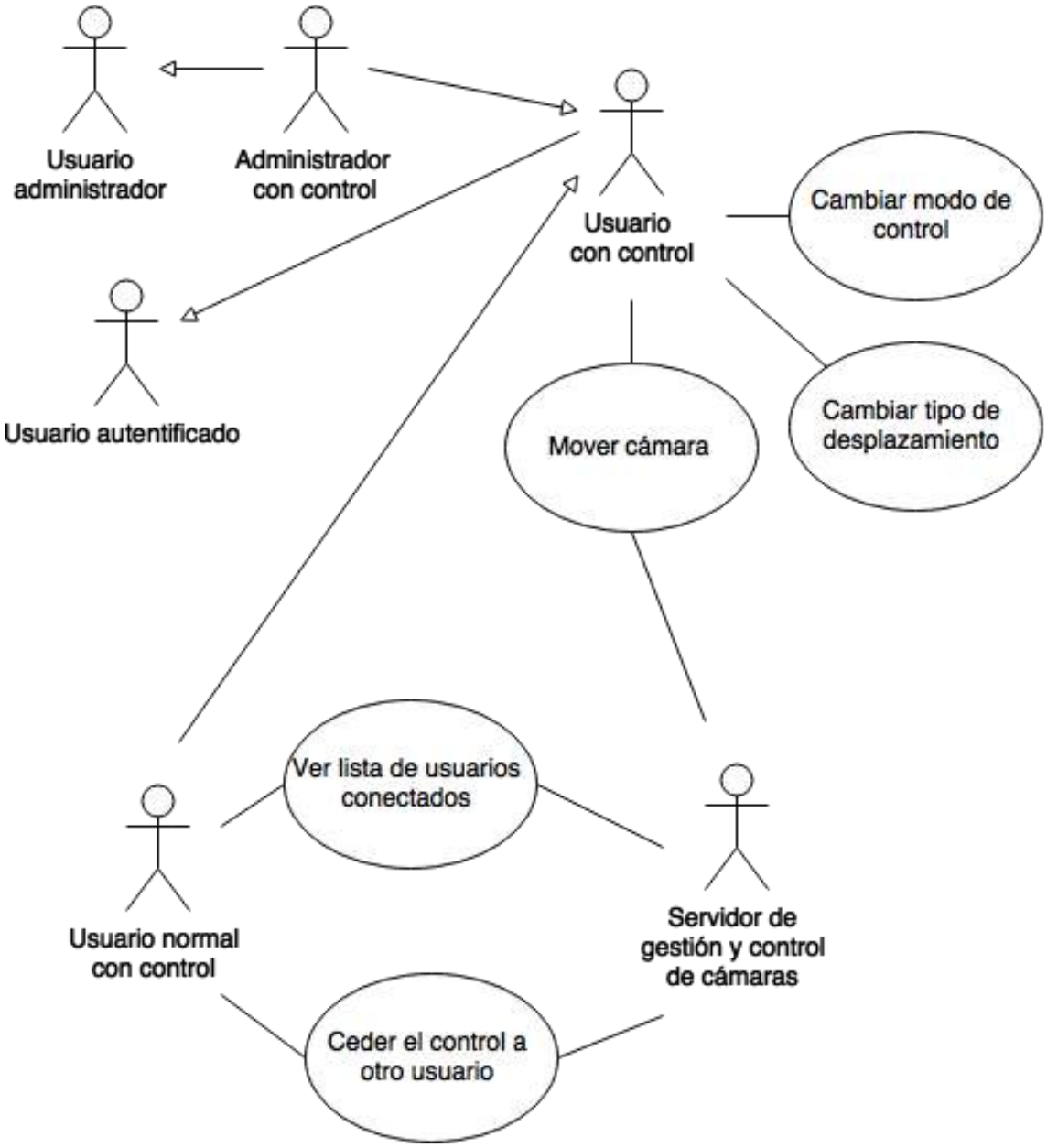

Figura 2.3: Diagrama de casos de uso para el control de las cámaras

- Caso de uso 3.1: Mover cámara:

DESCRIPCIÓN

Cuando el usuario está visualizando el *streaming* de una cámara móvil, existe la posibilidad de mover la cámara del mismo siempre y cuando se le haya concedido el control. Esto se hará mediante una interfaz que aparece solamente cuando se dispone del permiso para desplazar la cámara.

ENTRADA

La entrada será uno de los distintos comandos:

- Desplazar hacia la izquierda, derecha, arriba o abajo y para el movimiento.
- Acercar o alejar la visión (*Zoom in/out*).
- Enfocar o desenfocar.

SALIDA

En el reproductor de vídeo se podrán visualizar cada uno de los movimientos que se realicen, siempre en tiempo real con un retardo de menos de un segundo en condiciones normales.

RESTRICCIONES

- El usuario tiene que seleccionar una cámara móvil.

FUNCIONAMIENTO

La gestión de quién lleva el control de la cámara en un determinado momento es llevada por el servidor. este es el que se encarga de dar el control a un dispositivo cuando el administrador otorga el control a un usuario.

Cuando un usuario se conecta a una cámara móvil, el dispositivo abre un socket.io, del que se hablará en más detalle en la sección **3.3.2**.

Si el usuario quiere mover la cámara ya sea en modo continuo o simple, el dispositivo mediante el socket.io manda el comando al servidor, el cual se encarga de transmitirlo a la cámara para que esta se mueva.

ACTORES QUE INTERVIENEN

- Usuario con control.
- Servidor de gestión y control de cámaras.

- Caso de uso 3.2: Cambiar modo de control (normal/avanzado):

DESCRIPCIÓN

Como ya se ha comentado anteriormente, existen el modo de control normal y el avanzado. El modo avanzado dará la opción al usuario de ajustar el enfoque de la cámara manualmente, mostrando dos botones extra en la interfaz para enfocar a un punto más lejano u otro más cercano.

Estos modos están disponibles en un menú, junto con un *checkbox*, donde el usuario puede cambiar entre el modo normal y el avanzado según sus preferencias.

ENTRADA

El nuevo estado del modo de control. Si se ha marcado el *checkbox*, avanzado, y si se ha desmarcado, normal.

SALIDA

Si pasa de normal a avanzado se marcará el *checkbox* y aparecerán los botones de enfoque. En el caso inverso se desmarcará el *checkbox* y se ocultaran dichos botones.

RESTRICCIONES

- El usuario tiene que seleccionar una cámara móvil.

FUNCIONAMIENTO

Cuando el usuario activa el modo avanzado, los botones que en ese momento estaban ocultos y desactivados, pasan a estar activos y habilitados. En el caso inverso se sigue el proceso contrario.

ACTORES QUE INTERVIENEN

- Usuario con control.

- Caso de uso 3.3: Cambiar tipo de desplazamiento (continuo/simple) :

DESCRIPCIÓN

Existen dos tipos de desplazamiento: el continuo y el simple. Cuando el modo continuo está activo, se añade el botón de parada a la interfaz de control. El primer tipo de desplazamiento consiste en dejar que la cámara se mueva sin parar una vez se envía un comando, y no se detenga hasta que se puse el botón de parada. En el segundo la cámara se moverá siempre y cuando se mantenga pulsado el botón del comando que se quiera enviar a la cámara, parándose en el momento que el usuario levanta el dedo de la pantalla.

Al igual que los modos normal y avanzado, esta opción está disponible en un menú y puede ser activada simplemente pulsando en su celda, la cual dispone de un *checkbox*.

ENTRADA

El nuevo estado del tipo de desplazamiento. Si se ha marcado el *checkbox*, continuo, y si se ha desmarcado, simple.

SALIDA

Si pasa de simple a continuo se marcará el *checkbox* y aparecerá el botón de parada. En el caso inverso se desmarcará el *checkbox* y se ocultará el botón.

RESTRICCIONES

- El usuario tiene que seleccionar una cámara móvil.

FUNCIONAMIENTO

Como en el caso del modo normal y avanzado, el botón de parada está oculto y desactivado por defecto, pasando a habilitarse y a mostrarse cuando se activa el modo continuo. En el caso inverso ocurre lo contrario.

ACTORES QUE INTERVIENEN

- Usuario con control.

- Caso de uso 3.4: Ver lista de usuarios conectados:

DESCRIPCIÓN

Cuando el modo de gestión automático está activo, un usuario que posee el permiso para mover la cámara puede abrir un panel en el que aparecerá una lista mostrando a los usuarios que están conectados a la cámara en ese mismo momento.

ENTRADA

Una lista de los usuarios conectados.

SALIDA

Aparece un panel en el centro de la pantalla con la lista de usuarios conectados a la cámara.

RESTRICCIONES

- El modo de gestión automático deberá estar activo.
- El *streaming* que se está reproduciendo debe ser de una cámara móvil.

FUNCIONAMIENTO

El dispositivo se encarga de pedir un listado con los usuarios conectados a la misma cámara al servidor.

Cuando este es recibido, se muestra un panel con una lista mostrando todos los usuarios conectados.

ACTORES QUE INTERVIENEN

- Usuario normal con control.
- Servidor de gestión y control de cámaras.

- Caso de uso 3.5: Ceder el control a otro usuario:

DESCRIPCIÓN

Cuando el usuario abre el panel para ver la lista de usuarios conectados a la misma cámara que él, puede ceder el control de la cámara a otro.

ENTRADA

Los datos del usuario al que se le va a ceder el control.

SALIDA

Desaparece la interfaz de control, tanto los botones como el menú con los distintos modos de movimiento o el botón para mostrar el panel con los usuarios conectados a la cámara. El que ha perdido el control pasa de ser un usuario normal con control a ser un usuario autentificado y el que recibe el control un usuario con control. Al final el usuario tendrá en pantalla el vídeo solamente.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

El usuario que pasa el control a otro envía un mensaje al servidor con sus datos.

El servidor entonces procederá a mandar dos mensajes, uno al usuario al que se le revoca el permiso ocultando su interfaz y otro al que se le otorgará el control de la cámara mostrando la interfaz con los correspondientes comandos.

ACTORES QUE INTERVIENEN

- Usuario normal con control.
- Servidor de gestión y control de cámaras.

2.3.4. Requisitos para la gestión de usuarios:

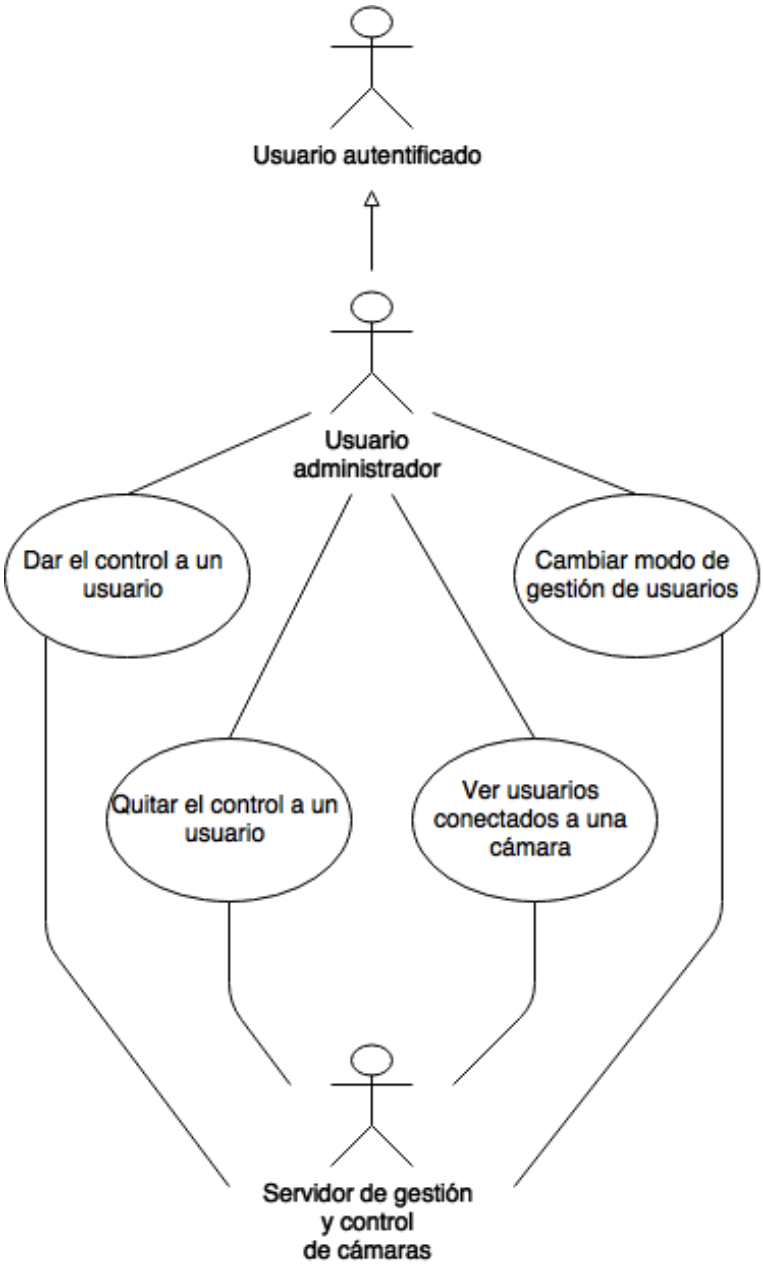

Figura 2.4: Diagrama de casos de uso para el login

- Caso de uso 4.1: Ver usuarios conectados a una cámara:

DESCRIPCIÓN

El administrador dispone de un panel de control diferente al que tiene el usuario normal cuando tiene el control, desde el que poder ver los usuarios conectados a una determinada cámara y quién tiene el control en el caso de que algún usuario disponga de él.

Además, en este panel puede tanto activar y desactivar el modo de gestión automático, como otorgar o revocar el control a un determinado usuario.

ENTRADA

Lista de los usuarios conectados a la cámara.

SALIDA

Se muestra el panel de administrador en el centro de la pantalla con los usuarios conectados a la cámara en ese mismo instante junto con una marca si alguno de ellos tiene el control, además de un botón para activar o desactivar el modo de gestión automático.

RESTRICCIONES

- El *streaming* que se está reproduciendo debe ser de una cámara móvil.

FUNCIONAMIENTO

El funcionamiento es el mismo que con el panel de usuario simple, se pide un listado con los usuarios conectados y una vez se recibe, se muestra el panel con la lista de los usuarios.

ACTORES QUE INTERVIENEN

- Usuario administrador.
- Servidor de gestión y control de cámaras.

- Caso de uso 4.2: Dar el control a un usuario:

DESCRIPCIÓN

Un administrador puede dar el permiso para controlar una cámara. En la ventana del usuario hay varios botones, uno de los cuales permite hacer lo mencionado previamente.

ENTRADA

Los datos del usuario al que se le va a conceder el control.

SALIDA

Se hace una transición de la vista que muestra el usuario y los botones de dar y quitar el control a la lista con los usuarios conectados a la cámara, ya actualizada mostrando quién tiene el control.

El administrador puede darse el control a sí mismo, haciendo que también aparezca la interfaz con todas las opciones para los controles. El usuario al que se le dé el control pasará a ser un usuario con control.

RESTRICCIONES

No hay ninguna restricción a especificar.

FUNCIONAMIENTO

El funcionamiento es igual al de conceder el control a otro usuario desde el panel de usuario normal para el caso de dar permiso para controlar la cámara. Mediante el socket.io, el usuario que pasa el control a otro envía un mensaje al servidor con sus datos. El servidor entonces procederá a mandar dos mensajes, uno al usuario al que se le revoca el permiso ocultando su interfaz y otro al que se le otorgará el control de la cámara mostrando la interfaz con los correspondientes comandos. En este caso el mensaje de revocar el control se lo mandará al que lo tenga actualmente.

ACTORES QUE INTERVIENEN

- Usuario administrador.
- Servidor de gestión y control de cámaras.

- Caso de uso 4.3: Quitar el control a un usuario:

DESCRIPCIÓN

De la misma manera que un administrador puede dar el control a un usuario cualquiera, también puede quitarlo pulsando el botón correspondiente.

ENTRADA

Los datos del usuario al que se le va a revocar el control.

SALIDA

La misma que al dar el control a otro usuario en cuanto a lo siguiente: Se hace una transición de la vista que muestra el usuario y los botones de dar y quitar el control a la lista con los usuarios conectados a la cámara, ya actualizada mostrando quién tiene el control. En el caso de que un administrador se quite el control a sí mismo le desaparecerá la interfaz con las opciones de los controles. El usuario al que se le quite el control pasará a ser un usuario autentificado o administrador dependiendo del tipo de cuenta.

RESTRICCIONES

- Que el usuario al que se le quite el control lo tenga en ese momento.

FUNCIONAMIENTO

Para revocar el control a un usuario se envía un mensaje con sus datos al servidor, y este se encargará de mandarle otro a dicho usuario que hará que deje de mostrársele la interfaz de control.

ACTORES QUE INTERVIENEN

- Usuario administrador.
- Servidor de gestión y control de cámaras.

- Caso de uso 4.4: Cambiar modo de gestión de usuarios:

DESCRIPCIÓN

Un administrador puede activar o desactivar el modo de gestión de usuarios automático. Este modo hará que el servidor junto con los usuarios gestionen quién tiene el control para mover cualquier cámara con esta opción activa.

Cabe destacar que aunque dicho modo esté activo, el administrador podrá seguir otorgando o revocando el control a los usuarios.

ENTRADA

Nuevo estado del modo de gestión, que puede ser automático o manual.

SALIDA

Cambiará el texto del botón mostrando "Desactivar el modo automático" en caso de que esté activado o "Activar el modo automático" en caso de que no lo esté.

Además, si el administrador que lo activa es el primero que se conectó a la cámara, obtendrá el control y aparecerá la interfaz para manejar la cámara. El usuario que reciba el control pasará a ser un usuario con control.

RESTRICCIONES

Ninguna restricción a especificar.

FUNCIONAMIENTO

Se manda un mensaje al servidor para que este comience a gestionar el control entre los usuarios.

El servidor escogerá al primer usuario de su lista de usuarios conectados de la cámara correspondiente y le mandará otro mensaje para otorgarle el control y que se muestre la interfaz para manejar la cámara.

ACTORES QUE INTERVIENEN

- Usuario administrador.
- Servidor de gestión y control de cámaras.

2.4. Requisitos no funcionales

- Nuestra aplicación será la única que podrá interactuar con nuestras cámaras, además de que las cuentas de usuario irán cifradas.
- Sólo una persona como máximo puede controlar una determinada cámara al mismo tiempo, debido principalmente a los conflictos que se generarían entre los usuarios.
- La aplicación usará la menor cantidad de recursos posible en cada momento, traduciéndose en un buen rendimiento.
- La aplicación tendrá un buen desempeño durante periodos prolongados de uso.
- En miras de posibles añadidos la aplicación será fácilmente extensible, facilitando al desarrollador la adición de nuevas características.
- El aprendizaje y el uso de la aplicación serán sencillos, consiguiendo una experiencia agradable para el usuario.
- La aplicación podrá ser usada en dispositivos con Android 4.0 y superior, debido en parte a las librerías que hemos empleado, como la de Youtube que Google ofrece para desarrolladores, siendo entonces el lenguaje utilizado Java junto con el SDK de Android.
- Las interfaces de usuario estarán optimizadas para dispositivos de 10.1". En concreto para el testeo de la aplicación se usarán tabletas BQ Edison 3.
- Para nuestro caso particular, nos hemos encontrado con una cámara cuyas librerías sólo eran compatibles con el lenguaje C, además de la necesidad de que el programa de la misma se debía ejecutar en la misma red de la cámara.

2.5. Diseño del sistema

2.5.1. Diseño de la interfaz de usuario

En este apartado se mostrarán las distintas interfaces de la aplicación, tanto las del usuario normal como las del administrador.

Pantalla de login

En la pantalla de login pueden observarse los dos campos de texto que sirven para ingresar el nombre del usuario y la contraseña, además del botón para intentar autentificarse con los datos introducidos en dichos campos.

Figura 2.5: Pantalla de login

Sidebar lateral

La barra lateral se puede abrir haciendo un gesto de *swipe* (desplazar el dedo del borde izquierdo de la pantalla hacia el centro) o pulsando arriba a la izquierda sobre el icono de Android. En la barra lateral se muestra una lista desplegable de 3 niveles totalmente configurable como veremos en el punto **3.4.5** por la que el usuario puede navegar pulsando sobre cada una de las filas.

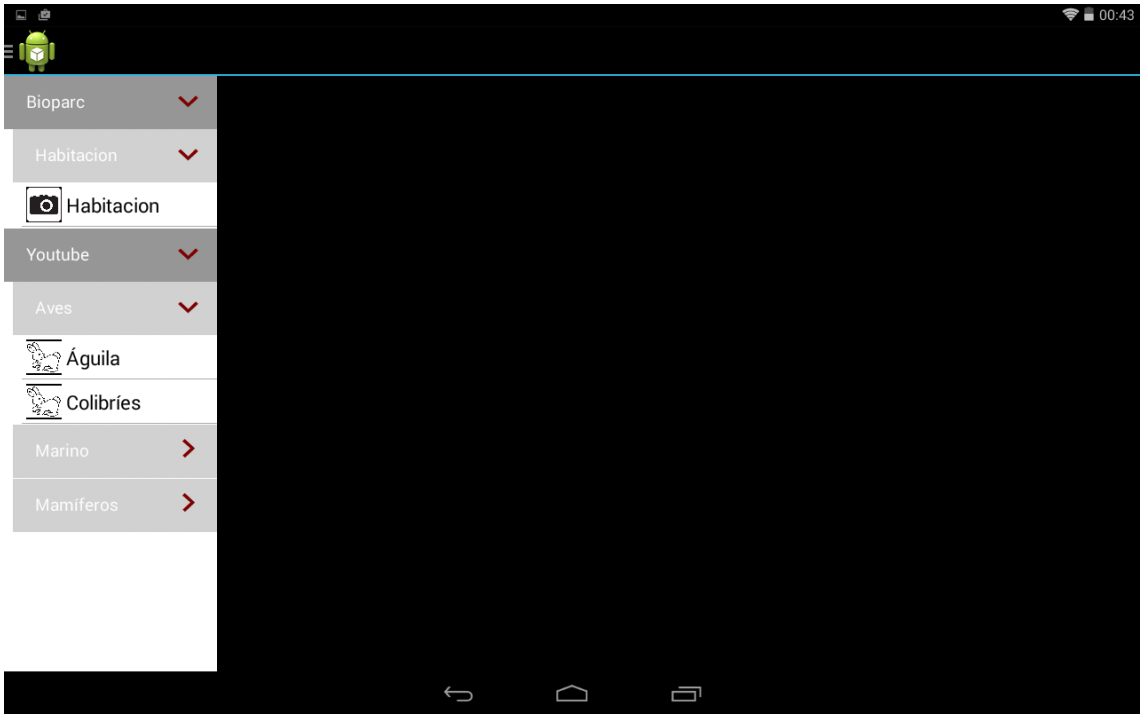

Figura 2.6: Sidebar con los distintos vídeos

Stream de Youtube

En la figura **2.7** se muestra el fragmento que contiene el reproductor de Youtube. En dicho reproductor se puede reproducir o pausar una retransmisión de vídeo, así como ponerlo a pantalla completa.

Figura 2.7: Stream de youtube

Stream de cámara remota

En la figura **2.8** se puede ver el *streaming* de una cámara remota, el cual a no ser que se tenga el control solamente se verá el vídeo como se muestra en la imagen. También se puede observar el botón que hay en la esquina superior derecha con una flecha en forma de círculo, el cual sirve para refrescar el vídeo.

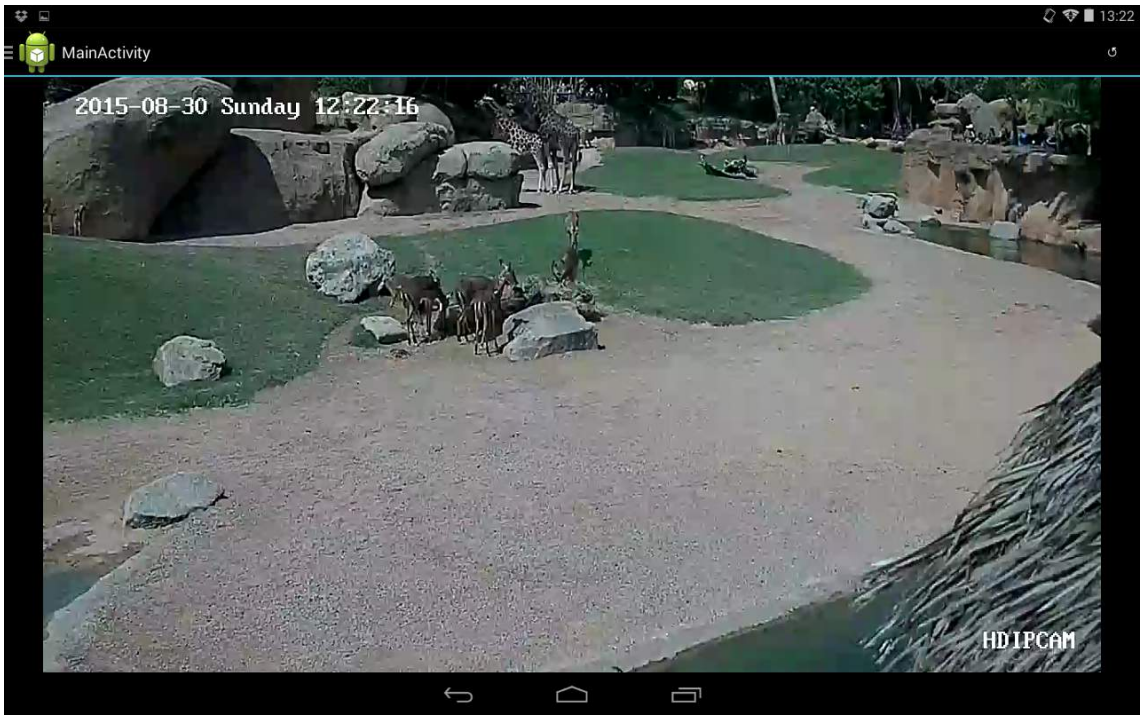

Figura 2.8: Stream de cámara móvil

Menú con los distintos modos/tipos de control

En la parte superior derecha de la figura **2.9** se puede ver el menú para cambiar entre los distintos tipos de movimiento y modos de control. En este caso ambos están marcados, por lo que están los botones de enfocar y los de *stop*.

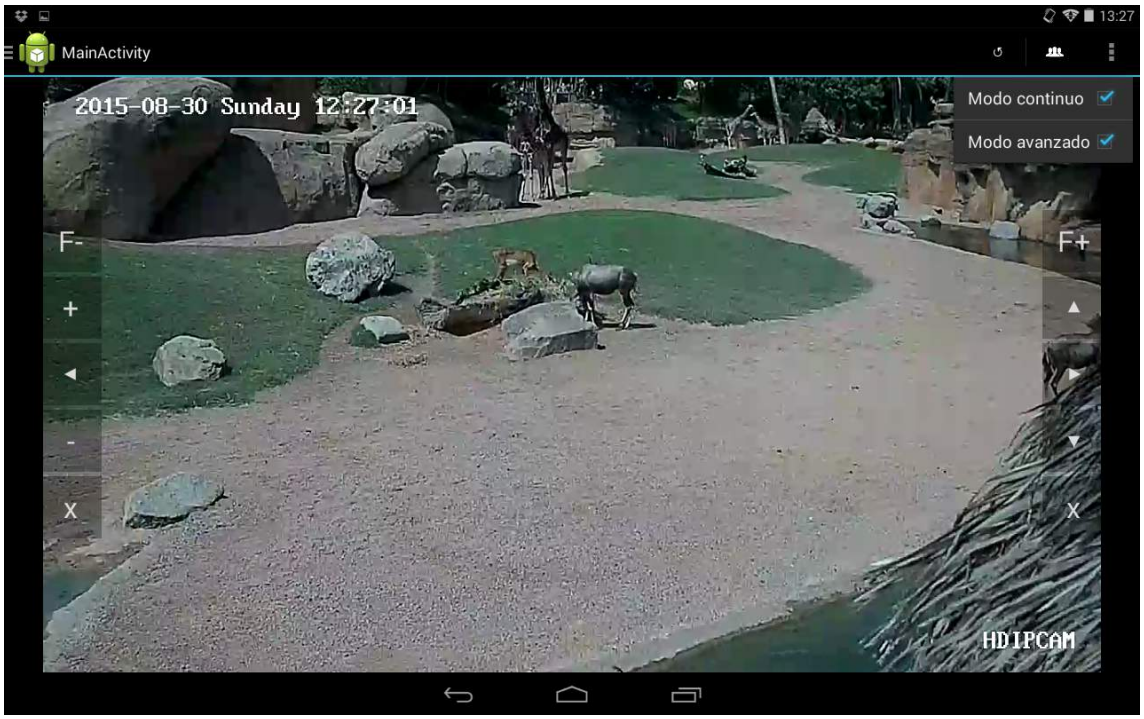

Figura 2.9: Menú con los distintos modos/tipos de control

Panel de administrador

El panel de admin visto en la figura **2.10** contiene una lista de los usuarios conectados a la cámara en ese mismo instante. En este caso sólo está conectado el administrador, por lo que es el único que aparece en esa lista. Además, aparece el botón para cambiar el modo de gestión de usuarios entre la cabecera de la lista y la lista de usuarios.

También se puede observar que el usuario de la interfaz tiene el control, por lo que aparece marcado con una X.

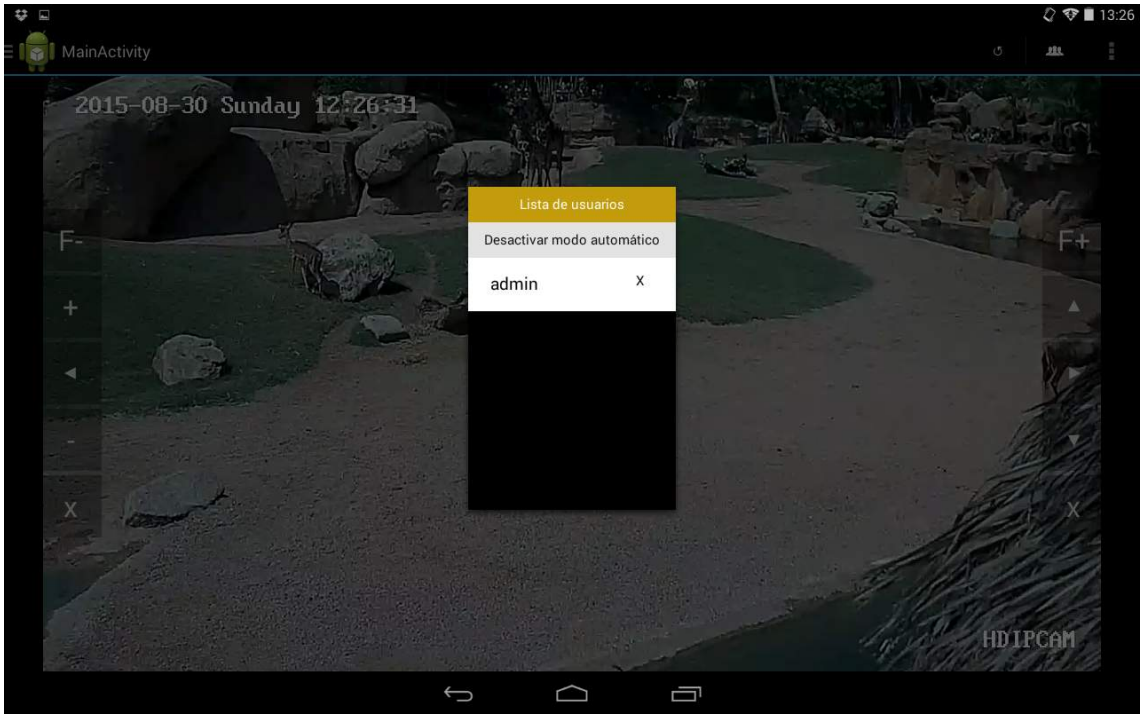

Figura 2.10: Panel de administrador

Panel de usuario

La vista de la figura **2.11** es la que aparece cuando un administrador selecciona a uno de los usuarios de la lista que aparece en el panel. Sigue apareciendo el botón para cambiar el modo de gestión de usuarios, además de una imagen, el nombre del usuario, dos botones para dar y revocar el control y un último botón para volver a la lista de usuarios.

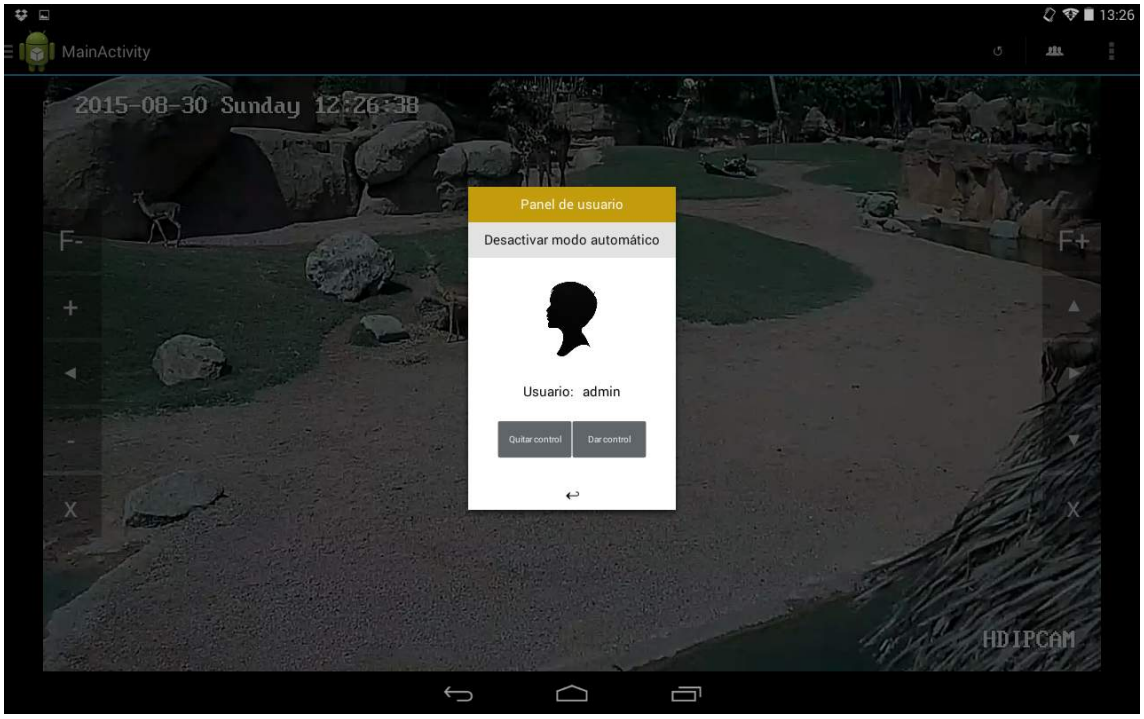

Figura 2.11: Panel de usuario

Panel de usuario normal para conceder el control a otro

A diferencia del panel de administrador, en el panel de la figura **2.12** solamente aparece un listado con los usuarios conectados. Por otra parte, detrás del panel se puede ver cómo el usuario tiene el control en ese momento, por lo que una X aparece en la fila de dicho usuario. Además, al pulsar en un usuario no se pasa a la vista de la figura **2.11**, sino que se cede al control a dicho usuario y la ventana se cierra haciendo desaparecer todos los controles y botones como el de abrir el panel o el de desplegar el menú de usuarios.

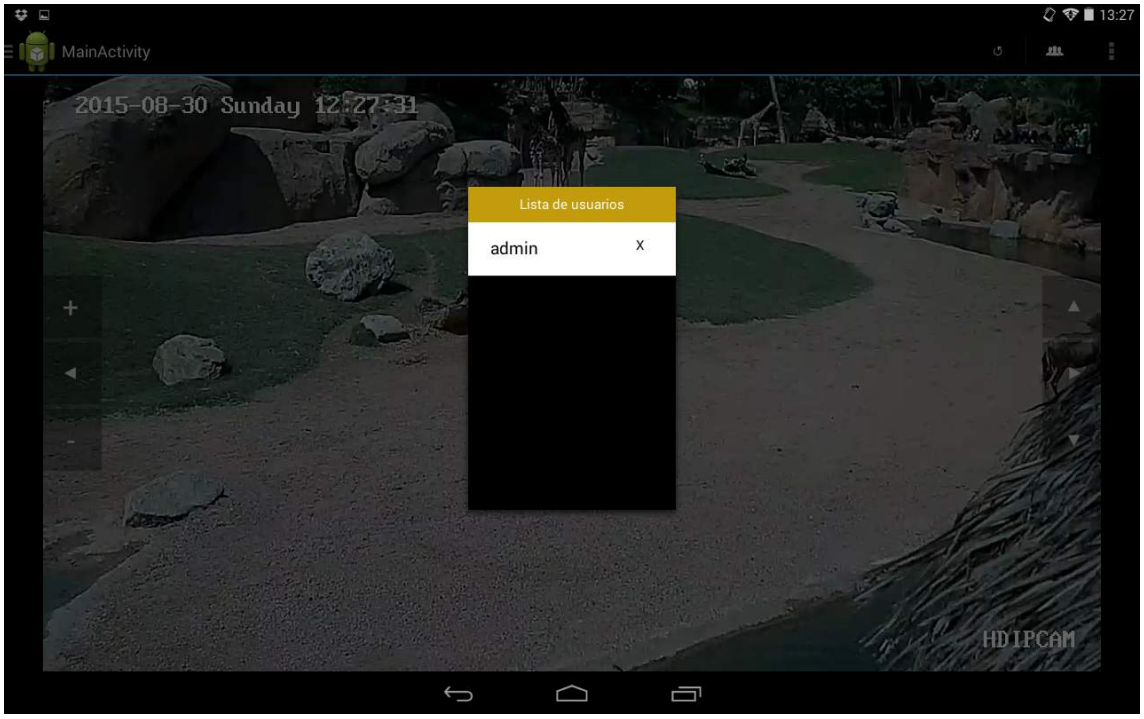

Figura 2.12: Panel de usuario normal para conceder el control a otro

2.5.2. Interfaces de control de la cámara

En esta sección se mostrarán las distintas interfaces de control que se han desarrollado, y que en un futuro se usarán para realizar un estudio de usabilidad con niños hospitalizados con el fin de ver cuál de ellas se adapta mejor a sus necesidades.

Interfaz de control con botones en la parte inferior

En estas interfaces se muestran los distintos botones disponibles cuando se posee el control de una cámara remota.

En la figura **2.13** se muestra una basada en la botonera clásica situada en la parte inferior de la pantalla. El tipo de desplazamiento está en simple y el modo de control en normal, mientras que en la figura **2.14** puede observarse que el modo de control está puesto en avanzado y el tipo de desplazamiento es continuo, mostrándose los botones de enfocar (F+/F-) y de parar(X). Los botones son semitransparentes, y se oscurecen ligeramente cuando son pulsados.

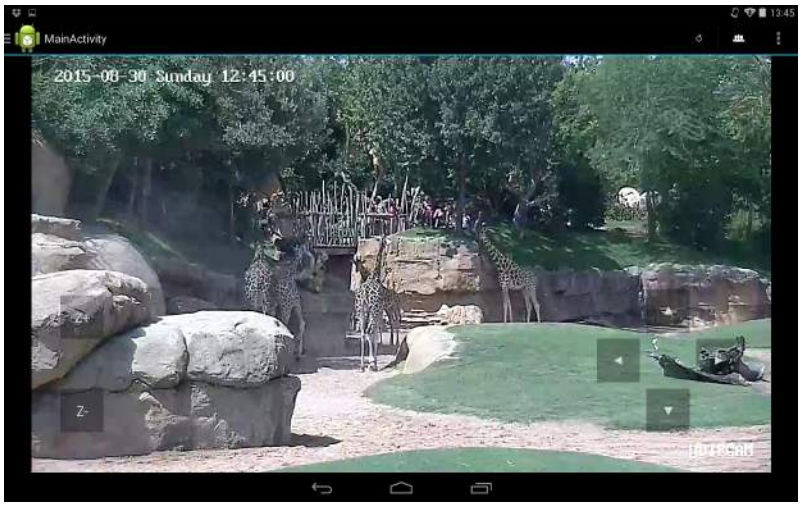

Figura 2.13: Interfaz con botones en la parte inferior, tipo de desplazamiento simple y modo normal.

Figura 2.14: Interfaz con botones en la parte inferior, tipo de desplazamiento continuo y modo avanzado.

Interfaz de control con botones en la parte lateral

La siguiente interfaz con botones consiste en una distribución diferente de los botones en la pantalla, situándose en este caso en la parte lateral de la misma. En la figura **2.15** podemos ver la interfaz con el tipo de desplazamiento simple y el modo normal, mientras que en la **2.16** aparece la misma interfaz con el tipo de desplazamiento continuo y el modo avanzado.

Figura 2.15: Interfaz con botones en la parte lateral, tipo de desplazamiento simple y modo normal.

Figura 2.16: Interfaz con botones en la parte lateral, tipo de desplazamiento continuo y modo avanzado.

Interfaz de control con *joysticks* virtuales

Los joysticks al igual que los botones sirven para mover las distintas cámaras. En el modo simple la parada se produce cuando sueltas el botón, mientras que en el continuo se debe mover el botón rojo al centro. En la figura **2.17** se puede observar una de las interfaces con *joysticks*. El modo de control podemos observar que es el normal, dado que no aparecen las etiquetas F+/F- en el *joystick* izquierdo. En cambio en la figura **2.18** en la cual la interfaz está en modo avanzado sí podemos ver dichas etiquetas.

Figura 2.17: Interfaz con *joysticks* virtuales y modo normal.

Figura 2.18: Interfaz con *joysticks* virtuales y modo avanzado.

Interfaz de control con *joysticks* físicos

La siguiente interfaz implementa los *joysticks* de la misma manera que los de la figura **2.17**, además las etiquetas de F+/F- funcionan igual también. La diferencia en este caso es que no se muestra los círculos interiores dado que no se usan directamente los dedos para mover los *joysticks*, sino que la idea es usar uno físico con tecnología conductiva como el de la figura **2.19**. Por ello, tampoco se implementa el modo continuo, ya que no hay posibilidad de mantener el *joystick* fijo en una posición si se suelta.

Figura 2.19: *Joystick* físico

Podemos observar la interfaz en modo normal en la figura **2.20** y en modo avanzado en la **2.21**.

Figura 2.20: Interfaz con *joysticks* físicos y modo normal.

Figura 2.21: Interfaz con *joysticks* físicos y modo avanzado.

2.5.3. Estructura de clases

Diagrama de clases de la actividad LoginWindow

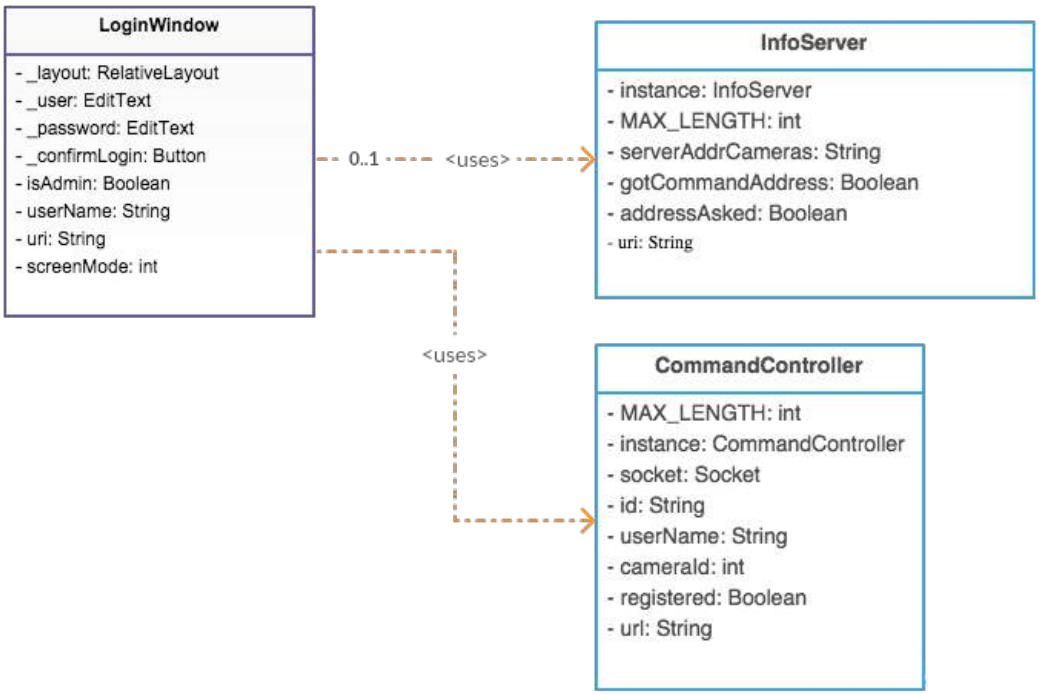

Figura 2.22: Relaciones del LoginWindow con el resto de clases

LoginWindow

La clase LoginWindow es la encargada de gestionar la autentificación del usuario. Hace uso de la clase InfoServer para las peticiones de autentificación, además de establecer el valor de la variable registered de CommandController a false.

Métodos y descripciones
onCreate
En este método se inicializa la actividad. En él se carga la UI y se enlaza a los distintos objetos de la clase para interactuar con ellos.
createAlert
Método para crear una alerta cuando se ha fallado al autentificarse.
tryLogin
Este método intenta autentificar al usuario mandando la petición con los datos al servidor.
login
Si la autentificación fue correcta este método realiza la transición a la actividad principal MainActivity.

InfoServer

La clase InfoServer se encarga de conectar a la aplicación con el servidor de contenidos. Es la encargada de enviar las peticiones de autentificación, además de pedir el listado de *streamings*.

Métodos y descripciones
InfoServer
Constructor de la clase.
getInstance
Retorna la instancia singleton de la clase.
setMainActivity
Asigna un valor a la variable mainActivity de la clase.
authenticate
Hace una petición para intentar autentificar al usuario.
getSidebarData
Recupera la lista de *streamings* de vídeo del servidor.
setLoginWindow
Asigna un valor a la variable loginWindow de la clase.
getUri
Devuelve el valor de la dirección del servidor de control y gestión ya obtenida del servidor.
getData
Método encargado de hacer las peticiones, dependiendo de la entrada hará la de autentificación, o la de pedir la lista de *streamings*.
convertInputStreamToString
Convierte el *stream* de datos recibido por la petición en una cadena de texto lista para ser usada por la aplicación.
doInBackground
Ejecuta el método getData en un hilo en segundo plano.
onPostExecute
Se encarga de procesar la información recibida por una petición.
encrypt
Cifra una cadena de texto dada.
decipher
Descifra una cadena de texto dada.

CommandController

CommandController es la clase que conecta a la aplicación con el servidor de control y gestión. Tiene toda la funcionalidad necesaria para enviar los comandos que se usan tanto para gestionar usuarios como para las cámaras.

Métodos y descripciones
CommandController
Constructor de la clase.
randomId
Método que genera una id aleatoria de tamaño MAX_LENGTH, que actualmente es 50.
setRegisteredAsFalse
Asigna el valor *false* a la variable registered.
resetSingleton
Reinicializa el singleton.
connect
Conecta el *socket*.
disconnect
Desconecta el *socket*.
close
Cierra el *socket*.
getId
Recupera la *id* del usuario.
isSocketNull
Si el *socket* no ha sido inicializado retorna *true*.
setMainActivity
Asigna un valor a la variable mainActivity.
sendMovement
Método a cargo de enviar los movimientos al servidor.
requestCameraUserListData
Método que recupera la lista de usuarios conectados a una cámara dada.
registerSocketOnCamera
Método que registra al usuario que llama a este método en la cola de una determinada cámara.
unregisterSocketOnCamera
Método que remueve al usuario que llama a este método de la cola de una determinada cámara.
giveControlToUser
Otorga el control de una determinada cámara a un usuario dado.
revokeControlToUser
Revoca el control de una determinada cámara a un usuario dado.
changeAutoMode
Cambio el modo de gestión de usuarios.
encrypt
Cifra una cadena de texto dada.
decipher
Descifra una cadena de texto dada.

Diagrama de la actividad MainActivity junto con todas las clases que la integran

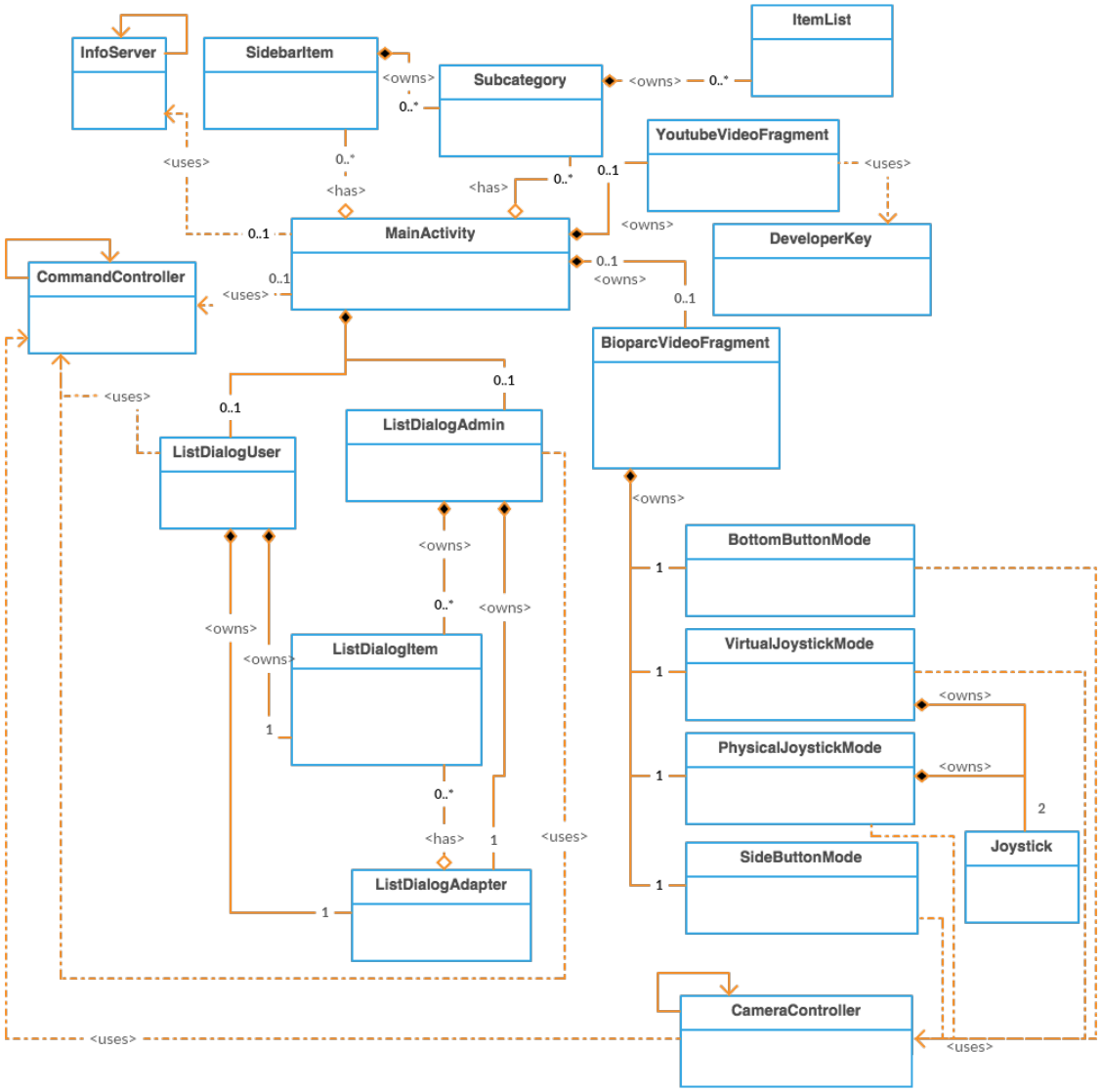

Figura 2.23: Diagrama de la actividad principal junto con todas las clases implicadas

La figura **2.23** incluye una visión general de las relaciones generales de la actividad MainActivity.

MainActivity y relaciones directas

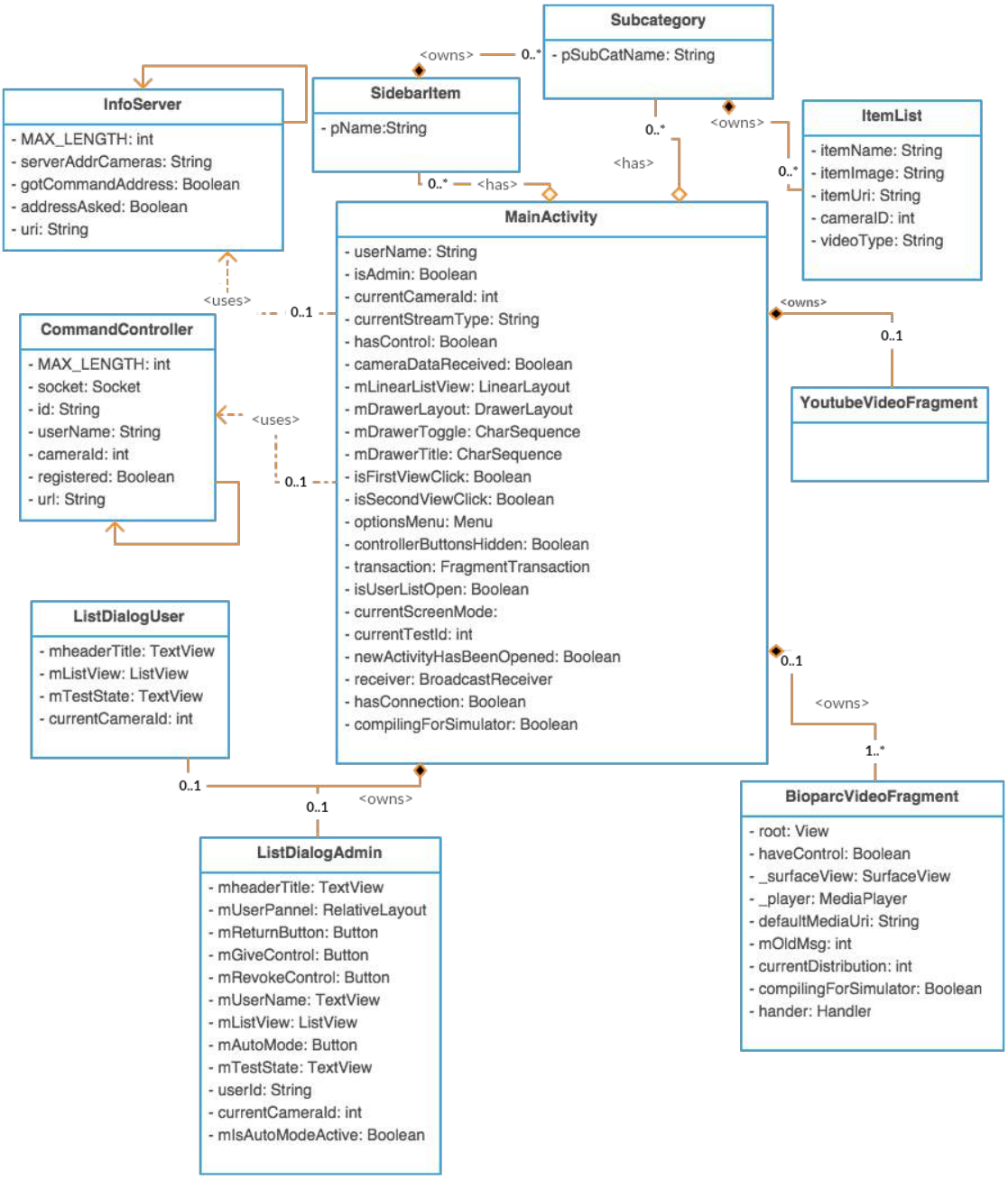

Figura 2.24: Diagrama que muestra todas las relaciones de la actividad principal MainActivity.

MainActivity

MainActivity es la actividad y clase principal de la aplicación. Es la encargada de implementar la lista multinivel de la barra lateral con todos los *streamings* de vídeo devueltos por el servidor de contenidos, de ahí su relación con InfoServer. Esto se puede ver mediante las relaciones con las clases SidebarItem y Subcategory. También debe gestionar los fragmentos que implementan los reproductores de vídeo BioparcFragment y YoutubeFragment. Por otra parte, también controla los paneles de control para usuarios y administradores recuperando las listas de dichos usuarios mediante la clase CommandController.

Métodos y descripciones
onCreate
En este método se inicializa la actividad. En él se carga la UI y se enlazan a los distintos objetos de la clase para interactuar con ellos.
onDestroy
Método destructor de la actividad.
onCreateOptionsMenu
Es el encargado de inicializar los botones de refrescar el vídeo, el menú desplegable con los *checkboxes* del tipo de desplazamiento y el modo de control y el botón para abrir el panel de administrador o de usuarios.
onOptionsItemSelected
Se encarga de procesar las interacciones de los usuarios con cada uno de los *items* del menú inicializado con onCreateOptionsMenu.
setTitle
Cambia el título en la *action bar* de la actividad.
showUserListDialog
Se encarga de abrir el panel de usuarios correspondiente al usuario, si es un administrador se abrirá el panel de control de administrador, mientras que si no está autentificado como tal se abrirá el panel de usuario normal.
getIsAdmin
Si el usuario es administrador retorna *true*, en caso contrario *false*.
getCurrentCameraId
Retorna la id a la que está conectado el usuario actualmente.
getCurrentStreamType
Retorna “bioparc” si el streaming que se está reproduciendo es de una cámara o “Youtube” si está alojado en dicha plataforma.
getAdminListDialogFragment
Retorna una referencia al panel de administrador.
getUserListDialogFragment
Retorna una referencia al panel de usuario.
setVideoFragment
Asigna un valor a la variable videoFragment.
getVideoFragment
Retorna una referencia a la variable videoFragment.
setupDrawer
Inicializa la vista de la barra lateral junto a todas las vistas y *listeners*.
setCameraData
Este método es el que está a cargo de procesar los datos de la barra lateral obtenidos mediante una petición. Extrae los datos del JSON y los convierte en objetos que pueden usarse para popular la vista.
generateData
Con los objetos devueltos por setCameraData este método se encarga de poblar la lista multinivel.

getIsUserListOpen
Retorna *true* si hay algún panel de control abierto o *false* si no lo hay.
setIsUserListOpen
Asigna un valor a la variable isUserListOpen. Ya que el servidor refresca esta lista cada vez que un usuario se conecta o desconecta, debemos controlar que no se intente modificar la vista que la controla cuando no está abierta.
createAlert
Método que muestra una alerta cuando un administrador intenta abrir el panel de control sin tener seleccionada una cámara.
videoFragmentManager
Método encargado de cargar un fragmento de vídeo ("Youtube" o "bioparc") dependiendo del tipo que sea.
showDDMenu
Hace que el botón del menú desplegable del tipo de desplazamiento y el modo de control sea visible.
hideDDMenu
Hace que el botón del menú desplegable del tipo de desplazamiento y el modo de control no sea visible.
showRefreshButton
Hace que el botón de refrescar el vídeo sea visible.
hideRefreshButton
Hace que el botón de refrescar el vídeo no sea visible.
showControlButton
Hace que el botón de abrir el panel de control sea visible.
hideControlButton
Hace que el botón de abrir el panel de control no sea visible.

SidebarItem

Es la clase del modelo del primer *item* de la lista multinivel.

Métodos y descripciones
SidebarItem
Constructor de la clase.
getpName
Retorna el valor de la variable pName.
setpName
Asigna un valor a la variable pName.
getmSubCategoryList
Retorna el valor de la variable mSubCategoryList.
setmSubCategoryList
Asigna un valor a mSubCategoryList.

Subcategory

Es la clase del modelo del segundo *item* de la lista multinivel.

Métodos y descripciones
SubCategory
Constructor de la clase.
getpSubCatName
Retorna el valor de la variable pSubCatName.
setpSubCatName
Asigna un valor a la variable pSubCatName.
getmItemListArray
Retorna el valor de la variable mItemListarray.
setmItemListArray
Asigna un valor a mItemListArray.

ItemList

Es la clase del modelo del tercer *item* de la lista multinivel y la que guarda la información de cada *streaming* de vídeo.

Métodos y descripciones
ItemList
Constructor de la clase.
getItemName
Retorna el valor de la variable itemName.
setItemName
Asigna un valor a la variable itemName.
getItemImage
Retorna el valor de la variable itemImage.
setItemImage
Asigna un valor a la variable itemImage.
getItemUri
Retorna el valor de la variable itemUri.
setItemUri
Asigna un valor a la variable itemUri.
getCameraId
Retorna el valor de la variable cameraId.
setCameraId
Asigna un valor a la variable cameraId.
getVideoType
Retorna el valor de la variable videoType.
setVideoType
Asigna un valor a la variable videoType.

Diagrama de BioparcVideoFragment y clases relacionadas

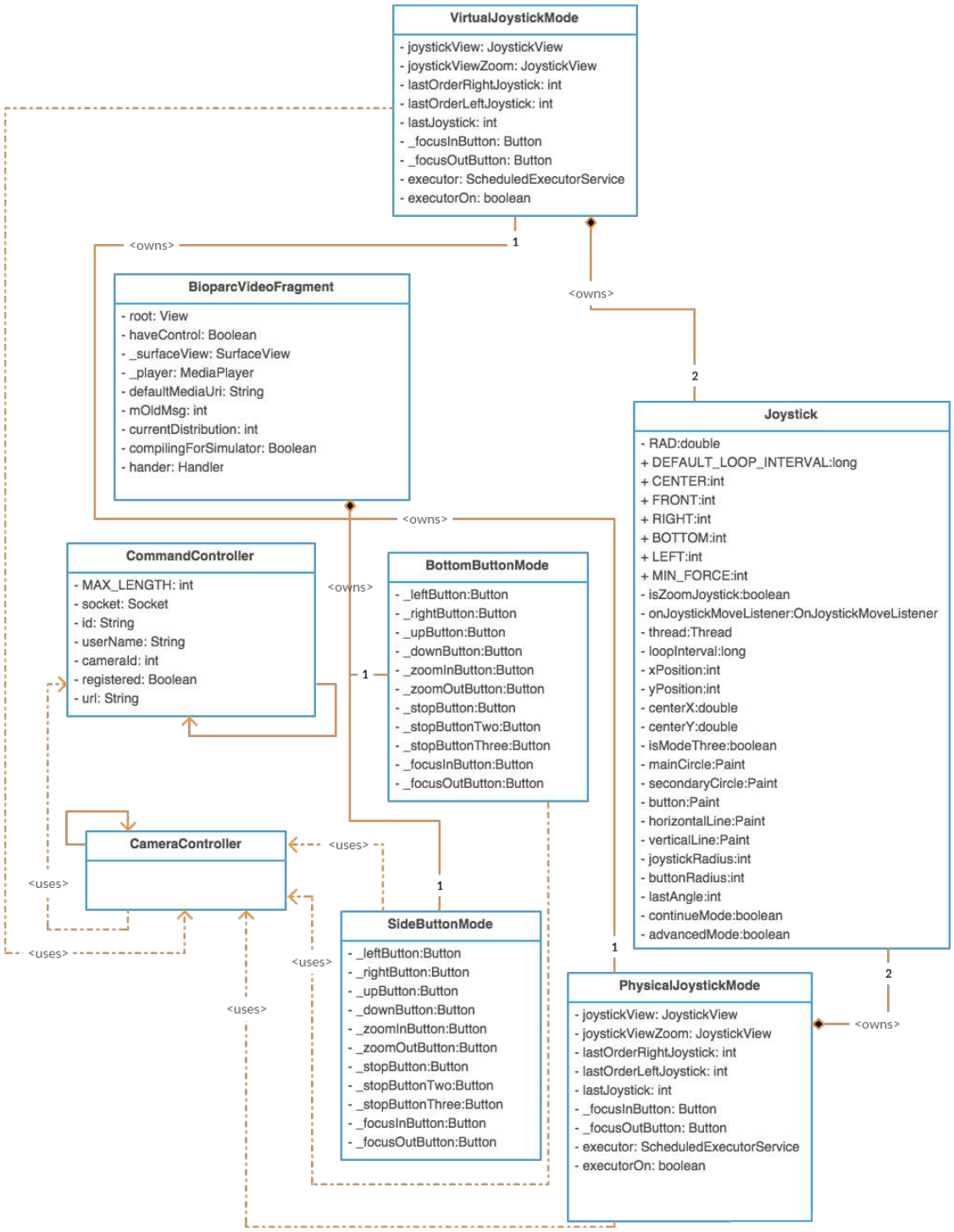

Figura 2.25: Diagrama que muestra las relaciones del fragmento para visualizar retransmisiones de cámaras y controlarlas

BioparcVideoFragment

La clase BioParcVideoFragment es la encargada de gestionar la reproducción de los *streamings* de vídeo de cámaras remotas además de las interfaces de control, a las cuales contiene.

Métodos y descripciones
onCreateView
Inicializa la vista del fragmento.
onPause
En este método se gestiona la salida de la aplicación a segundo plano, como cuando el usuario la minimiza por ejemplo.
onResume
Se encarga de recuperar el estado de la aplicación después de haber sido pausada, como cuando el usuario abre la aplicación después de haberla minimizado.
onDestroy
Método destructor del fragmento.
newPlayer
Crea un nuevo objeto reproductor de vídeo.
resetVideo
Reinicia el reproductor.
status
Retorna el estado del reproductor tras cada cambio.
onReceiveData
Monitoriza la información del *streaming* de vídeo en todo momento registrando el buffer en cada fragmento de información recibido.
newUri
Asigna una nueva dirección del *streaming* de vídeo.
setModeOne/setModeTwo/setModeThree/setModeFive
Selecciona la interfaz número BottomButtonMode/SideButtonMode/VirtualJoystickButton/PhysicalJoystickButton como el modo de control actual.
hideCurrentDistribution
Oculta la interfaz de control.
showCurrentDistribution
Muestra la interfaz de control.
setMode
Se encarga de hacer las llamadas a los métodos de setModeX para seleccionar las distintas interfaces.
setContinueMode
Cambia el tipo de desplazamiento (simple, continuo).
setAdvancedMode
Cambia el modo de control (normal, avanzado).

BottomButtonMode y SideButtonMode

Se encargan de implementar los botones de las interfaces de control, tanto de cargarlos como de configurar los *listeners*.

Métodos y descripciones
BottomButtonMode/SideButtonMode
Constructor de la clase.
init
Inicializa variables y llama a los métodos para configurar botones.
setupLeftButton, setupRightButton, setupUpButton, setupDownButton, setupZoomIn, setupZoomOut, setupFocusIn, setupFocusOut, setupStop.
Los diferentes métodos cargan sus respectivos botones y configuran los *listeners*.
setContinueMode
Hace visibles o invisibles los botones del tipo de desplazamiento (los de *stop*).
setAdvancedMode
Hace visibles o invisibles los botones del modo avanzado (los de enfocar y desenfocar).

VirtualJoystickMode/PhysicalJoystickMode

Se encargan de añadir los joysticks a la interfaz y de implementar el *callback* para decidir qué acción realizar tras cada movimiento.

Métodos y descripciones
VirtualJoystickMode/PhysicalJoystickMode
Constructor de la clase.
init
Inicializa los *joysticks* y sus *listeners*.
startExecutor
Crea un nuevo hilo que mandará la posición del *joystick* al servidor con una frecuencia de 0.25 segundos.
stopExecutor
Destruye el hilo creado con startExecutor.
setContinueMode
Hace una llamada al método setContinueMode de los dos *joysticks*.
setAdvancedMode
Hace visibles o invisibles las etiquetas enfocar y desenfocar en el *joystick* izquierdo y una llamada al método setAdvancedMode del *joystick* izquierdo.

Joystick

Esta clase es la encargada de implementar el joystick al completo excepto por el *callback* en el que se decide qué hacer tras cada movimiento, que es implementado por la clase que instancia el joystick.

Métodos y descripciones
JoystickView
Constructor de la clase.
modeInit
Guarda en una variable el tipo de interfaz en la que se usarán los *joysticks*, la del modo virtual o el físico.
initJoystickView
Inicializa la vista.
onDraw
Se encarga de dibujar la vista al completo.
onTouchEvent
Captura los eventos de cada toque dentro del *joystick* y hace la llamada al *callback* que implementan las clases que añaden a los *joysticks* con los datos de getAngle, getPower y getDirection.
measure
Define el tamaño del *joystick*.
resetPosition
Mueve el *joystick* al centro de la zona.
isInCenter
Devuelve *true* si el *joystick* está en el centro de la zona y *false* si no lo está.
getAngle
Obtiene el ángulo en el que se sitúa el *joystick* con respecto al centro de la zona.
getPower
Obtiene la distancia desde la posición del *joystick* con respecto al centro de la zona y la normaliza a un número entre 0 y 100.
getDirection
Mapea la posición del *joystick* usando los valores predefinidos LEFT, RIGHT, BOTTOM, FRONT.
setOnJoystickMoveListener
Configura el *listener* del *joystick* para que se ejecute la función de recalcular la posición cada vez que hay un cambio de la misma.
setContinueMode
Hace que el *joystick* se quede en el sitio una vez se deja de pulsar la pantalla, haciendo que la cámara siga ejecutando el comando que se estaba realizando a la hora de levantar el dedo.
setAdvancedMode
Hace que en el caso del *joystick* izquierdo el usuario pueda desplazar el mismo por los ejes horizontales (los que representan los comandos de enfocar y desenfocar)

CameraController

Esta clase funciona como un intermediario entre las interfaces que implementan los distintos modos de control y la clase que envía los comandos al servidor. Debido a que hay varias interfaces resulta más conveniente redirigir los eventos de todos los botones a un método en común encargado de enviar los comandos pertinentes. En caso de que haya que hacer algún cambio en lo que se envía al servidor para mover la cámara, sólo habrá que modificar esta clase y no las otras interfaces.

Métodos y descripciones
CameraController
Constructor de la clase.
turnLeft, turnRight, turnUp, turnDown, zoomIn, zoomOut, focusIn, focusOut, moreSpd, lessSpd, focusIn, focusOut, stop.
Todos estos métodos mandan su determinado comando a la función sendMovement del singleton CommandController.

YoutubeVideoFragment

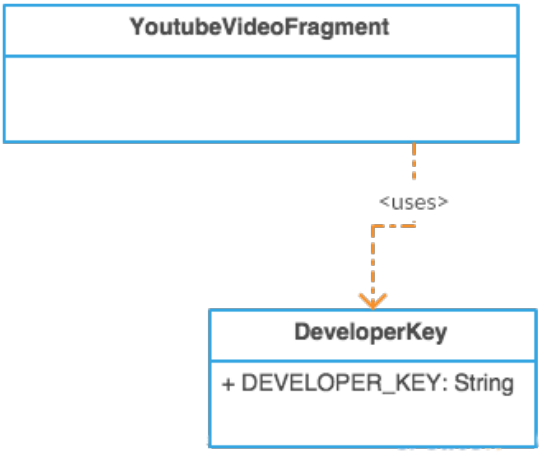

Figura 2.26: Diagrama que muestra las relaciones del fragmento para visualizar retransmisiones de Youtube

Esta clase es la que se encarga de implementar el reproductor de Youtube usado para reproducir los vídeos de ese servicio. Hace uso de una variable estática guardada en la clase DeveloperKey para validar la conexión, necesario dado que por ahora esta API sólo está disponible para desarrolladores.

Métodos y descripciones
YoutubeVideoFragment
Constructor de la clase.
newInstance
Crea una nueva instancia del reproductor con la configuración pertinente.
init
Valida la instancia creada en newInstance con la clave de desarrollador que provee Youtube, mostrando un mensaje en caso de error o inicializando el reproductor si el proceso se realiza con exito.

Diagrama de ListDialogAdmin y ListDialogUser y sus relaciones

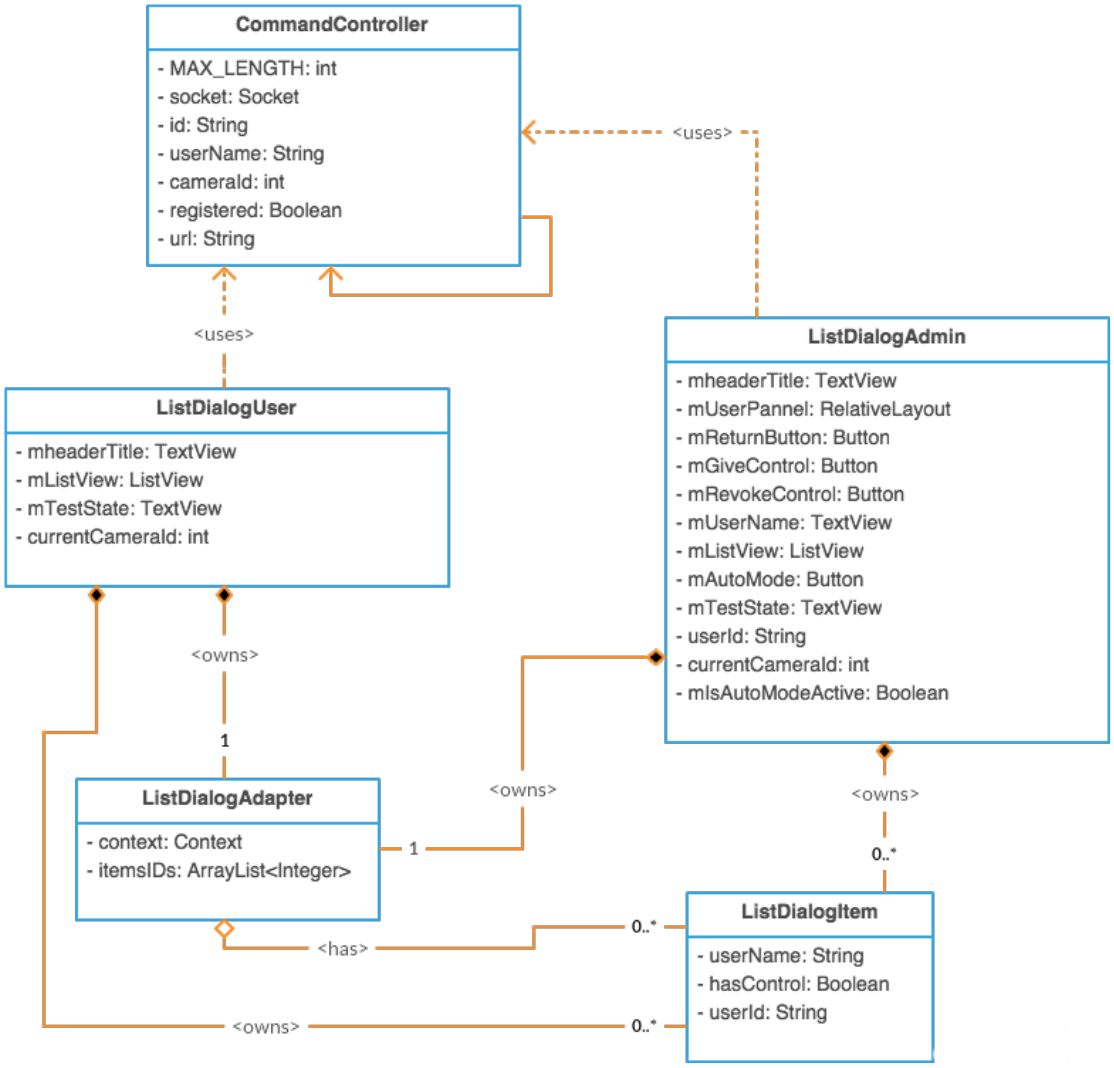

Figura 2.27: Diagrama que muestra las relaciones de los paneles que usan los usuarios para ceder el control y los administradores para gestionar usuarios.

ListDialogAdmin

Esta clase se encarga de gestionar el panel que los administradores utilizan para gestionar a los usuarios y las cámaras. Hace uso del adaptador ListDialogAdapter para no tener que hacerse cargo de interactuar con el modelo ListDialogItem, tarea que realiza dicho adaptador.

Métodos y descripciones
ListDialogAdmin
Constructor de la clase.
onCreateView
Carga la vista del panel.
onActivityCreated
Se encarga de crear el adaptador y configurar la lista de usuarios con él.
onDestroy
Destructor de la clase.
onResume
Cuando la lista se muestra después de estar en segundo plano por ejemplo manda una petición para refrescar la lista de usuarios.
onItemClick
Captura los eventos de pulsar una celda de la lista. En este caso mandará un mensaje al servidor para cederle el control al usuario determinado.
refreshList
Método que hace una llamada a la función refreshData del adaptador con una lista nueva de usuarios.
changeCameraMode
Cambia el texto del botón del modo de gestión automático.
listToPannel
Hace una transición desde la lista de usuarios al panel con los botones para otorgar y revocar el control, además del que hace la llamada a pannelToList.
pannelToList
Hace una transición desde el panel con los botones para otorgar y revocar el control, además del que hace la llamada a pannelToList a la lista de usuarios .

ListDialogUser

Esta clase se encarga de gestionar el panel que utilizan los usuarios normales para ceder el control a otros. Al igual que la clase ListDialogAdmin esta clase también utiliza el adaptador de la clase ListDialogAdapter, el cual se encarga de interactuar con el modelo abstrayéndolo por completo.

Métodos y descripciones
ListDialogUser
Constructor de la clase.
onCreateView
Carga la vista del panel.
onActivityCreated
Se encarga de crear el adaptador y configurar la lista de usuarios con él.
onDestroy
Destructor de la clase.
onResume
Cuando la lista se muestra después de estar en segundo plano por ejemplo hace una llamada al método listToPannel.
onItemClick
Captura los eventos de pulsar una celda de la lista. En este caso mandará un mensaje al servidor para cederle el control al usuario determinado.
refreshList
Método que hace una llamada a la función refreshData del adaptador con una lista nueva de usuarios.

ListDialogAdapter

El adaptador es el encargado de actuar como intermediario entre las clases ListDialogAdmin/ListDialogUser y el modelo ListDialogItem, proveyendo la funcionalidad necesaria para gestionar la lista abstrayendo por completo el modelo de la vista.

Métodos y descripciones
ListDialogAdapter
Constructor de la clase.
getUserName
Retorna el nombre de un usuario dada su posición en la lista.
getUserId
Retorna la id de un usuario dada su posición en la lista.
refreshData
Actualiza la lista actual con una nueva y notifica a la vista del cambio.
getCount
Devuelve el número de usuarios actualmente en la lista del panel.
getView
Se encarga de configurar el aspecto de la celda que representará a cada usuario.
getUserId
Retorna la id de un usuario dada su posición en la lista.

ListDialogItem

Esta clase representa el modelo de un usuario. Es usado por el adaptador para crear las vistas y acceder a los atributos de cada usuario.

Métodos y descripciones
ListDialogItem
Constructor de la clase.
getUserName
Retorna el valor de la variable userName.
setUserName
Asigna un valor a la variable userName.
getHasControl
Retorna el valor de la variable hasControl.
setHasControl
Asigna un valor a la variable hasControl.
getUserId
Retorna el valor de la variable userId.
setUserId
Asigna un valor a la variable userId.

2.5.4. Patrones de diseño

En la ingeniería del software, los patrones de diseño son la base para encontrar soluciones a diversos problemas que aparecen en el desarrollo de una aplicación. Su uso es similar al de una plantilla o descripción de cómo solucionar un problema que puede ocurrir en diversas situaciones.

En esta aplicación se han usado dos patrones, que describiremos a continuación:

Singleton

El patrón de diseño Singleton [13] es una solución para aquellas aplicaciones en las que sólo se necesita una instancia de una clase. El objetivo de aquellas clases que se basan en este diseño es controlar la creación de objetos, limitando este número a uno como hemos comentado previamente, además de ofrecer un punto de entrada global para el uso de ese único objeto.

Normalmente existen dos tipos de inicialización, la eager y la lazy:

- **Eager initializacion:** La instancia se crea antes de que vaya a ser usada, normalmente cuando la aplicación se inicia.
- **Lazy initialization:** La instancia se crea en el momento que ésta va a ser usada. En su primer uso se crea el objeto y en posteriores se pasa una referencia al mismo.

En nuestra aplicación hemos usado la inicialización lazy. Nos ha sido útil a la hora de gestionar una clase que crea el socket necesario para interactuar con el servidor, evitando que se creen múltiples instancias del mismo y otorgando un único punto de acceso a la capa que se encarga de la comunicación.

MVA (Model-View-Adapter)

MVA o Mode-View-Adapter [13] es un patrón usado para manejar grandes cantidades de datos buscando conseguir que el modelo de dichos datos y la interfaz estén completamente separados. A diferencia del patrón MVC que organiza el controlador, la vista y el modelo en una forma triangular (figura **2.15**), el patrón MVA lo hace en una línea posicionando el adaptador entre el modelo y la vista sin que exista ninguna conexión entre estas dos (figura **2.16**). La vista como ya se ha comentado está totalmente desacoplada del modelo de manera que ésta sólo puede interactuar con él por medio del adaptador. De esta manera sólo el adaptador conoce de la existencia tanto de la vista como del modelo, siendo entonces su responsabilidad la de mediar entre ambos.

Con esta implementación se consigue poder reorganizar los datos del modelo sin la necesidad de modificar la interfaz, o modificar la interfaz sin tener que cambiar el modelo. Además, existe la posibilidad de que varias vistas puedan usar un mismo adaptador, o de que varios adaptadores puedan usar también el mismo modelo, evitando repetir código en algunas ocasiones.

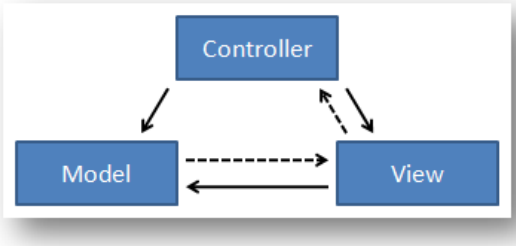

Figura 2.28: Diagrama del patrón MVC

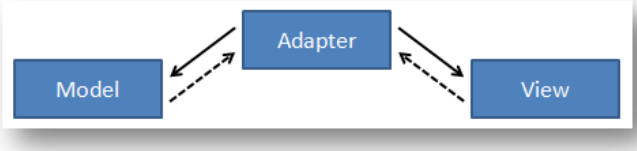

Figura 2.29: Diagrama del patrón MVA

En nuestro caso particular hemos usado este patrón para gestionar las listas de los paneles de usuarios, aprovechando que con Android puedes hacer un adaptador a medida y configurar dichas listas con él. De esta manera hemos conseguido desacoplar completamente el modelo de los usuarios que aparecen en la lista de la propia interfaz que la muestra como podemos ver en el diagrama de la figura **2.27**, en el que las clases ListDialogUser y ListDialogAdmin únicamente guardan los valores de las listas para pasárselos al adaptador a la hora de crearlo para que este las gestione.

3. Servidores

Además de la aplicación, existen dos servidores con los que nuestra aplicación interactúa para ofrecer al usuario todas las funciones ya descritas. Son estos los que se encargan de proveer al usuario la información necesaria junto con la posibilidad de controlar las distintas cámaras, además de permitir a los administradores gestionar el control de las mismas. Cada uno de ellos cumplirá una función específica:

- Servidor de contenidos

  El servidor de contenidos es el primer servidor con el que interactúa la aplicación. Es el que le permite autentificarse para posteriormente otorgarle tanto la información de los distintos streamings de vídeo como la de las direcciones de los servidores de gestión y control de las cámaras.

- Servidor de gestión y control de cámaras

  Este servidor será el encargado de gestionar a los distintos usuarios como dicten los administradores, además de hacerse cargo de todo el sistema de movimiento de la cámara.

Un dato importante a conocer es que para controlar la cámara se debe estar en la misma red que ella, por ello la funcionalidad para su control deberá estar alojada en un servidor en su misma red. Usando dos servidores y separando funciones se consigue que aunque el dueño de las cámaras cierre el servidor de control, los usuarios todavía tendrán acceso a los distintos streamings alojados en Youtube, ofreciendo una alternativa para cuando el servicio de cámaras móviles no esté disponible. Existe la posibilidad de usar un servidor que cumpla todas las funciones, pero puede darse el caso de que el acceso a las cámaras remotas no sea controlado por el administrador de la aplicación, por lo que si se apaga el servidor para que las cámaras remotas no sean accesibles, también dejarán totalmente inservible la aplicación, al perder también las funciones del servidor de contenidos dejando al usuario sin la opción de autentificarse y ver *streamings* de Youtube, cosa que no pasa usando dos servidores.

En cuanto al hardware de los servidores, el estacionado en la UPV y que opera como servidor de contenidos cuenta con el descrito en la figura **3.1**, y el que está en el Bioparc y actúa como servidor de gestión y control con el especificado en la **3.2**.

Tabla 3.1: Hardware del servidor de contenidos

Servidor	**Servidor de contenidos**
CPU	Intel(R) Xeon(R) CPU X3430 @ 2.40GHz
RAM	512MiB System memory

Tabla 3.2: Hardware del servidor de gestión y control

Servidor	**Servidor de gestión y control**
CPU	Intel(R) Xeon(R) CPU E5-2450L v2 @ 1.70GHz
RAM	1GiB System Memory

Por último, en lo que atañe al sistema operativo se ha usado Ubuntu 14.04 64-bits (Trusty Tahr).

3.1. Cámara

La cámara ha sido uno de los puntos centrales de todo el servicio, ofreciéndonos la posibilidad de tener nuestros propios streamings de vídeo además de la opción de cambiar su orientación e inclinación. Para la implementación de nuestro servicio, hemos usado la siguiente[1]:

7Inch HD 720P High Speed PTZ IP Camera 20X Zoom IR 120m ONVIF Outdoor

Figura 3.1: Foto de la cámara usada.

La cámara dispone de unas opciones limitadas de movimiento. Es capaz de moverse en 360º horizontalmente y 90º verticalmente, pero no ofrece la posibilidad de realizar movimientos usando un número determinado de grados. Además, los movimientos de la cámara son continuos, significando que al mandar un comando la cámara efectuará la acción pertinente al mismo hasta que se le indique que realice otra diferente. Por otra parte, retransmite en 720p y el acceso tanto a dicha retransmisión como al controlador de la cámara están protegidos mediante unas credenciales definidas por el administrador.

[1] http://www.camera2000.com/es/7inch-hd-720p-high-speed-ptz-ip-camera-20x-zoom-ir-120m-onvif-outdoor.html

Para mandar los comandos a la cámara se ha usado un programa escrito en C. El funcionamiento de dicho programa se puede observar en la figura **3.2**.

```
      Conectar a cámara
      datos <- Información de autentificación guardada
en fichero
      Si datos incorrectos
         Entonces ir a fin
ptz:
      comando <- entrada estándar
      Si comando es igual a "exit"
         Entonces ir a fin
      Si comando correcto
         Entonces enviar comando a cámara
         Si no mostrar error
      ir a ptz
fin:
```

Figura 3.2: Posible estructura para el listado de cámaras.

Lo primero que hace el programa es establecer una conexión con la cámara. Lo siguiente es obtener los datos de autentificación, que están guardados en un archivo en el mismo directorio que el programa. Cuando se han conseguido se intenta autentificar al usuario, saliendo del programa en el caso de que los datos no sean válidos o entrando en el bucle que nos permitirá introducir los comandos si lo son. Una vez en el bucle el programa espera a que se introduzca un comando, el cual puede ser "exit" con lo que se terminará el programa u otro diferente, en cuyo caso si es válido se enviará a la cámara y si no lo es se mostrará un error. Si el comando introducido no ha sido 'exit', en el momento que se realice la acción pertinente se volverá a esperar a que el usuario introduzca otro comando repitiendo el bucle de nuevo.

3.2. Nodejs

3.2.1. Introducción a nodejs

Nodejs (también conocido como Node) [15] es una manera de usar JavaScript en el servidor. Está basado en el motor Open Source V8, el mismo intérprete de JavaScript que utiliza el navegador Google Chrome[2]

Para entender mejor cómo funciona, es conveniente conocer de qué manera servidores tradicionales como los implementados con Ruby (Ruby on Rails + Unicorn/Puma/Thin) [16] o LAMP (Linux + Apache + MySQL + PHP) [17] funcionan, asignando hilos a las distintas peticiones como se muestra en la figura **3.3**. Esto quiere decir que si una petición conlleva las siguientes tareas (que en este caso son dependientes las unas con las otras en el orden que se indica):

1. Petición a la base de datos para ver si el usuario existe.
2. Petición a la base de datos para crear el nuevo artículo.
3. Almacenar en el disco duro la imagen adjunta al artículo.
4. Enviar al usuario una respuesta con el código de estado 200 (todo correcto).

El servidor asignará un hilo para que se haga cargo de todas ellas.

Figura 3.3: Esquema sobre la delegación de tareas al pool de hilos.

[2]Google Inc. The V8 JavaScript engine, http://code.google.com/p/v8

Node en cambio funciona con un único hilo como puede observarse en la figura **3.4**, haciendo uso de eventos y nunca bloqueando el hilo principal. Para ello, al iniciarse el servidor se crea una piscina con un número predefinido de hilos que quedan a la espera, y, en el momento en el que se recibe una petición, Node envía las tareas asíncronas que intervienen en ella a la piscina de hilos, siendo estos los que se encargan de procesarlas y de disparar eventos cuando éstas han finalizado para que el hilo principal se haga cargo.

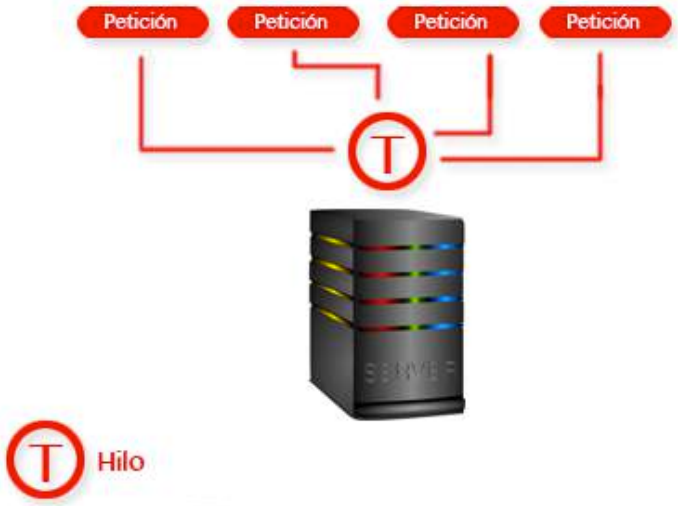

Figura 3.4: Esquema sobre la delegación de tareas al pool de hilos.

La principal diferencia de Node con respecto a los ya mencionados servidores tradicionales es su capacidad para procesar las distintas tareas de las peticiones por separado, que en muchas ocasiones permite procesarlas en un periodo corto de tiempo comparado con los otros servidores.

Un posible ejemplo sería con dos servidores, uno implementado con Ruby y otro con Node. Suponiendo que ambos tienen 5 hilos y que llegara una petición en la que hiciese falta escribir 500 mensajes en 500 *sockets*, el servidor implementado con Ruby asignaría la tarea a a un único hilo, mientras que el implementado en Node dividiría las 500 escrituras entre los 5 disponibles.

De todas formas, un servidor en Node puede bloquearse de manera sencilla, al fin y al cabo dispone de un único hilo de ejecución y éste puede verse fácilmente superado si se intentan hacer cálculos complejos o procesamientos pesados de datos en el hilo principal.

Por último, cabe destacar que Node dispone de una gran cantidad de módulos que extienden su funcionalidad básica, proporcionando a los desarrolladores nuevas herramientas para facilitar el trabajo. Hablaremos sobre dos de esos módulos en los puntos siguientes, ya que han sido de gran importancia durante el desarrollo de la infraestructura de nuestro servicio.

3.2.2. socket.IO

Socket.io[3] es un módulo o librería programada completamente en JavaScript y externa a Node que permite una comunicación bidireccional, ofreciendo la posibilidad de crear aplicaciones en tiempo real para navegador o dispositivo móvil.

Mediante esta librería podemos enviar información del servidor al cliente sin que éste la solicite y viceversa, haciendo uso de un sistema de eventos en el que estos tienen un nombre o identificador que nos resultan de utilidad para ejecutar las porciones de código que deseemos cuando enviemos un mensaje.

En nuestro caso especialmente, esta librería nos ha sido de gran utilidad por varios motivos. El más importante ha sido la gestión de los usuarios, ya que para otorgar permisos a un usuario por ejemplo, es necesario que el servidor pueda ser capaz de enviar mensajes a los dispositivos. Socket.IO permite establecer una comunicación bidireccional, que junto a poder guardar referencias a todos los sockets creados hace posible enviar mensajes por ellos en cualquier momento.

Otra de las ventajas de esta librería es el poco retardo que proporciona entre el envío de un mensaje desde un punto y la recepción del mismo en el destino. En ciertas ocasiones un retardo ligeramente bajo no resulta de mucha importancia, pero cuando estamos hablando de enviar comandos para mover una cámara, la más mínima latencia influye enormemente en la experiencia del usuario.

3.2.3. Modo de gestión del control automático

Como se ha comentado anteriormente, el servidor que controla la gestión de los usuarios a una determinada cámara dispone de un modo en el que se hace cargo por sí mismo de otorgar y revocar el control de la cámara entre usuarios.

En cada nueva conexión, el servidor se encarga de registrar al nuevo usuario conectado en una cola, siempre al final de la misma. Ésta es usada exclusivamente cuando el modo de gestión del control automático está activo.

El funcionamiento es el siguiente. Cuando un administrador activa este modo, el servidor le otorga el control al usuario que está en la primera posición de la cola. Este usuario puede entonces otorgarle el control a otro. En caso de que el usuario que está en primera posición se desconecte, se eliminará de la cola y el control pasará al segundo. Si el modo es desactivado, el usuario que tenía el control pasa al final de la cola y se le revoca el control. Por otra parte, también existe la posibilidad de poner un límite de tiempo para cada usuario, después del cual el que tiene el control será automáticamente puesto al final.

[3]Socket.io, http://socket.io/

3.2.4. child_process

El módulo child_process[4] nos permite invocar subprocesos e interactuar con ellos por medio de las entradas y salidas estándar.

Debido a que las librerías de nuestra cámara estaban solamente disponibles para C, nos fue necesario el uso de este módulo, ya que con él podemos ejecutar programas escritos en otros lenguajes en un proceso separado y comunicarnos con él a través de la entrada estándar.

El programa que se ha usado para este caso se puede ver en la sección **3.1**, en la que se ha puesto un pseudocódigo mostrando su funcionamiento.

3.2.5. Almacenamiento de los datos

Para esta tarea hemos usado archivos JSON (**J**ava**S**cript **O**bject **N**otation) [5] .

La figura **3.5** muestra una posible estructura para un fichero con el listado de cámaras.

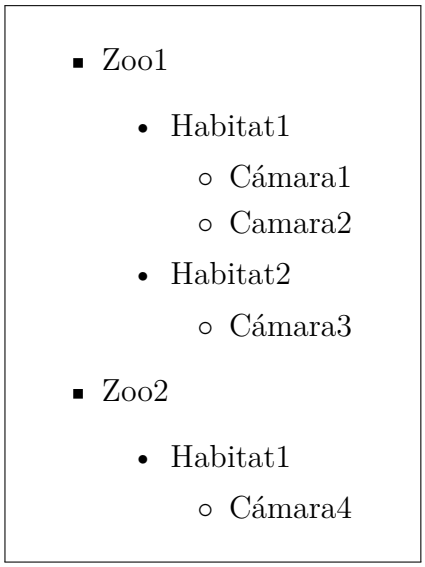

Figura 3.5: Posible estructura para el listado de cámaras.

A continuación se muestra parte del archivo que usamos para las cámaras de nuestro servicio, el cual se basa en retransmisiones en directo sobre animales y sus habitats.

[4]Child_process, https://nodejs.org/api/child_process.html
[5]JSON, http://json.org/

```
{
  "Youtube":{
    "name":"Youtube",
    "habitats":{
      "Mamíferos":{
        "name":"Mamíferos",
        "cameras":[
          {
            "uri":"k9T7t9uZnUY",
            "image":"http://www.dibujalia.com/data/media/
               29/conejo.gif",
            "name":"Perros 1",
            "type":"youtube",
            "id": "6"
          },
          {
            "uri":"7ltzAXvcpzk",
            "image":"http://www.dibujalia.com/data/media/
               29/conejo.gif",
            "name":"Perros 2",
            "type":"youtube",
            "id": "7"
          }
        ]
      },
      "Aves":{
        "name":"Aves",
        "cameras":[
          {
            "uri":"rxGoGg7n77A",
            "image":"http://www.dibujalia.com/data/media/
               29/conejo.gif",
            "name":"Águila",
            "type":"youtube",
            "id": "1"
          }
        ]
      }
    }
  },
  "Bioparc":{
    "name":"Bioparc",
    "habitats":{
      "Habitacion":{
```

```
42          "name":"Habitacion",
43          "cameras":[
44            {
45              "uri":"rtsp://somertsplink/mpeg4",
46              "image":"http://www.clker.com/cliparts/3/3/6/
                   4/12074316411296807266camera%20white.svg.hi
                   .png",
47              "name":"Habitacion",
48              "type":"bioparc",
49              "id": "0"
50            }
51          ]
52        }
53      }
54    }
55  }
```

Como se puede observar, el primer nivel está formado por los objetos “Youtube” y “Bioparc”, ambos con un nombre y un objeto “habitats” que es el que contiene los items de segundo nivel. Los habitats al igual que los objetos de primer nivel poseen un nombre, pero además tienen un array de cámaras, que son las que conforman el tercer y último nivel de nuestra lista multinivel. Las cámaras tienen una id, un nombre, la dirección de la retransmisión, un enlace directo a una imagen y un tipo para diferenciar las que son streamings de youtube o de cámaras de vigilancia (en nuestro caso bioparc).

4. Seguridad

Uno de los problemas principales a la hora de gestionar las cámaras es intentar que ninguna persona ajena a nuestro servicio pueda acceder a ellas. Para ello se decidió usar una combinación de técnicas que consiguiesen dificultar dicha tarea a cualquiera que intentase conectarse al servidor que gestiona las cámaras y los usuarios.

Podría darse la situación de que una persona se conectase al servidor de gestión de cámaras y usuarios quitando el control a otros usuarios, controlando una cámara él mismo, o realizando cualquier tarea gestionada por el servidor. Debido a esto se implementó un sistema de seguridad en el que cada cliente dispone de un tiempo preestablecido de unos pocos segundos para validar esa conexión.

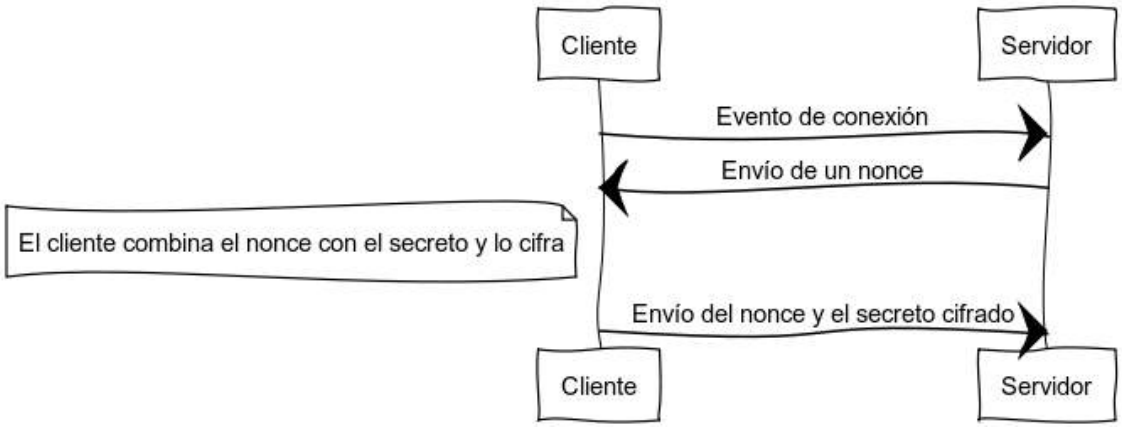

Figura 4.1: Diagrama de secuencia de comprobación de seguridad para el servidor de gestión de cámaras y usuarios

Como se puede apreciar en la imagen, el proceso para comprobar que la conexión viene de nuestra aplicación se basa en el uso de un *nonce* y un secreto cifrados. El cliente una vez recibe el *nonce* lo combina junto con el secreto y los cifra. Una vez hecho esto lo envía al servidor que se encarga de descifrarlo y comprobar que tanto el secreto como el *nonce* son válidos. Si en el tiempo que se da para realizar este proceso (actualmente 15 segundos) no se envía una combinación correcta el servidor cerrará la conexión automáticamente.

El uso de los *nonce* es vital para evitar ataques de repetición, en el cual una persona puede captar los datos de la petición y enviar una copia haciéndose pasar por un usuario de nuestro servicio. Para evitarlo se mantiene un registro de los *nonces* que se han enviado y están activos y borrándolos tanto en el momento que son usados correctamente como cuando se cumple el límite de tiempo para la comprobación.

En cuanto al servidor de contenidos, únicamente se cifra la información de la autentificación de los usuarios. Con el resto de peticiones y datos se decidió no utilizar estas técnicas ya que no se obtenía ningún beneficio, además de que podría dar pie a un fallo de seguridad en el que una persona podría realizar un ataque a nuestro servicio llamado *Chosen-ciphertext attack* [18], que consiste en obtener la clave privada mediante el uso del texto cifrado y del mensaje descifrado. Se podrían cifrar dichas transmisiones, pero dado que los datos que provee el servidor son datos de cámaras a las que conectarse, podemos asumir que cualquier persona capaz de monitorizar dichas transferencias de datos también sería capaz de detectar todas y cada una de las conexiones a las cámaras de dicho archivo. Este servidor sirve simplemente para autentificarse y enviar la información de dichos streamings de vídeo, por lo que como ya hemos dicho no obtenemos ninguna ventaja a la hora de cifrar datos aparte de los de autentificación, como el id o la contraseña especialmente. Cabe destacar que el proceso de autentificación sigue unos pasos parecidos al del servidor de gestión de usuarios y cámaras.

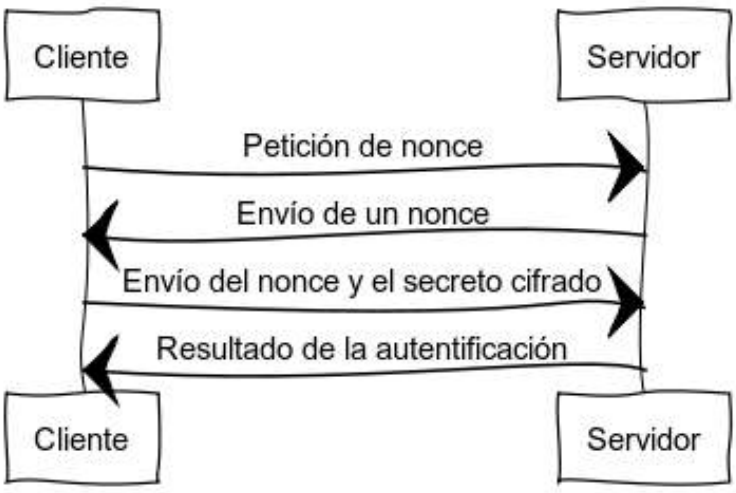

Figura 4.2: Diagrama de secuencia de comprobación de seguridad para el servidor de contenidos

El proceso de cifrado y descifrado es el mismo con ambos servidores. En cambio la obtención del *nonce* aquí se realiza con una petición explícita, no automáticamente como pasa cuando se conecta el socket. Por lo tanto el servidor cada vez que crea un *nonce* le asigna una tiempo de caducidad de 15 segundos al igual que el otro servidor, eliminándolo una vez transcurrido este plazo.

Si la combinación de secreto y *nonce* es válida se responderá a la petición ya sea ésta correcta o no con el código oportuno. Si la combinación no es válida simplemente se ignorará y no se responderá de ninguna manera.

Para la encriptación de los mensajes transmitidos se ha utilizado la combinación de tres técnicas de encriptación. Al desconocer el atacante el orden en el que se han aplicado, se consigue añadir un problema más para romper el mecanismo de seguridad [19]. A continuación se explican las diversas técnicas y algoritmos usados.

4.1. Algoritmo RSA

RSA [20] es un algoritmo de criptografía asimétrica usado hoy en día para encriptar y desencriptar mensajes. Esto implica que en el proceso intervienen dos claves diferentes, una pública y otra privada. La gran ventaja de este algoritmo reside en la dificultad de obtener los factores primos de un número grande.

La clave pública es un valor auxiliar que se distribuye junto con el producto de dos números primos, los cuales deben mantenerse en secreto. Cualquiera puede usar la clave pública para encriptar un mensaje, el cual se puede desencriptar haciendo uso de la clave privada que va asociada a la clave pública con la que se ha encriptado dicho mensaje.

El método principal para romper este encriptado es haciendo uso de la clave pública, el producto de los dos primos y del texto cifrado para obtener la clave privada. En nuestro servicio los únicos que van a intervenir van a ser nuestra aplicación y el servidor, por lo que podemos guardar las claves en ellos para no tener que transmitirlas. Además, existen otras posibles maneras de atacar nuestro servicio, como el *Chosen-ciphertext attack* que ya hemos mencionado en la sección **4.1**.

4.2. Inserción de datos falsos

Para dificultar la obtención del texto original se insertan datos de relleno siguiendo un patrón determinado (esteganografía) [21], que junto a los otros dos métodos dificulta un poco más a cualquier persona que intente descifrar nuestro texto encriptado.

4.3. Cifrado por sustitución

El cifrado por sustitución [22] consiste en reemplazar caracteres de un texto por otros siguiendo un determinado sistema. Estas sustituciones pueden ser de cualquier número de caracteres, ya sea por pares, tríos, o una letra (el más común). Para descifrar un mensaje cifrado con esta técnica, simplemente se sigue el proceso inverso al de cifrado. Los ataques para este tipo de cifrado se suelen hacer por fuerza bruta o teniendo en cuenta la frecuencia de cada letra o grupos de letras.

Existen varios tipos de algoritmos de sustitución:

- **Cifrado por sustitución simple:** Se sustituye una sola letra en todo momento y se divide además en dos tipos:
  - **Monoalfabético:** Se dice que un cifrado por sustitución simple es monoalfabético para cuando una letra A, siempre tiene la misma sustitución B en todo el texto.
  - **Polialfabético:** Al contrario que en el monoalfabético, en el polialfabético una letra A puede tener distintas susticiones en distintas partes del texto, como por ejemplo B y C.
- **Cifrado por sustitución poligráfico:** Se sustituyen grupos de letras.

En nuestro caso hemos usado un cifrado de sustitución simple polialfabético.

5. Conclusiones y trabajo futuro

En este trabajo se ha abordado el análisis, el diseño y la implementacion de un servicio de visualización y control remotos de cámaras de vigilancia para dispositivos Android. El sistema consta de una aplicación móvil y de dos servidores, uno situado en la Universitat Politècnica de València y que provee de un mecanismo de autentificación, además de proporcionar el listado de *streamings* de vídeo a la aplicación, y otro situado en Bioparc Valencia, el cual gestiona el acceso a una cámara montada en uno de los ecosistemas del zoológico.

También se han diseñado e implementado diversas interfaces gráficas para el control de la cámara, las cuales serán evaluadas en el futuro por parte de niños hospitalizados en el Hospital La Fe de Valencia, en el contexto de un proyecto para fomentar la inteligencia emocional y aliviar los efectos negativos del estrés que sufren los niños en esa situación. En dicho proyecto, se utilizará HabitApp para realizar actividades con los niños basadas en la observación remota de animales en directo a través de dispositivos móviles.

Como aspecto a mejorar, quedaría la parte de la interfaz gráfica, en la cual es posible implementar diseños más vistosos y llamativos para hacer la aplicación más atractiva a los niños.

6. Bibliografía

[1] Encuesta de Morbilidad Hospitalaria Año 2013,
http://www.ine.es/prensa/np878.pdf

[2] Coyne, I. "Children's experiences of hospitalization." Journal of Child Health Care 10(4), pp. 326-36, 2006.

[3] Nagera, H. "Children's reactions to hospitalization and illness." Child Psychiatry and Human Development, 9(1): 3-19, 1978.

[4] Skipper, J.K., Leonard, R. "Children, Stress, and Hospitalization: A Field Experiment." Journal of Health and Social Behavior, 9(4): 275-287, 1968.

[5] Bonn, M. The effects of hospitalisation on children: a review. Curationis 17(2), 20-4, 1994.

[6] Kaminski, M., Pellino, T., Wish, J. "Play and pets: the physical and emotional impact of child-life and pet therapy on hospitalized children." Children's Health Care, 31(4), 321–335, 2002.

[7] Kaplan, H. I. "Comprehensive textbook of psychiatry (6th ed.)." Baltimore: Williams & Wilkins, 1995.

[8] Ruth, R. "Animals Are Helping Children Overcome Physical and Emotional Challenges". Interactions 10(1), pp. 16-18, 1992.

[9] Buxton, B. Multi-touch systems that I have known and loved. 2013.
http://billbuxton.com/multitouchOverview.html [Último acceso 26/09/2014].

[10] Smith, S.P., Burd, E., and Rick, J. "Developing, evaluating and deploying multi-touch systems." International Journal of Human-Computer Studies 70, 10, 653–656, 2012.

[11] Rideout, V. "Zero to Eight: Children's Media Use in America." Common Sense Media, 2011.

[12] Johnson, L., Adams, S., and Cummins, M. "The NMC Horizon Report: 2012 K-12." The New Media Consortium, Texas, 2012.

[13] Gamma, E, Helm, R, Johnson, R, Vlissides, J: “Design Patterns: Elements of Reusable Object-Oriented Software”. Addison-Wesley, 1995.

[15] Mardan, A., “Practical Node.js: Building Real-World Scalable Web Apps 1st Edition”.

[16] https://devcenter.heroku.com/articles/deploying-rails-applications-with-the-puma-web-server.

[17] Gerner, J., Naramore, E., Owens, M., Warden, M., “Professional LAMP: Linux, Apache, MySQL and PHP5 Web Development”.

[18] Victor, S., “Why Chosen Ciphertext Security Matters”. IBM Research Division, Zurich, 1998.

[19] Himanshu, G., “Designing some multiple and multiphase encryption techniques for the enhancement of data security”. Department of Computer Science, Haridwar University, India, 2014.

[20] Kurose, J., Ross, K., “Redes de Computadoras”, 5ta Edición, capítulo 8.

[21] Pye, P., Tun, M. “A Novel Secure Combination Technique of Steganography and Cryptography”. University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar, Computer University (Pathein), Myanmar, 2014.

[22] Trappe, W., Clinton L., Introduction to Cryptography with Coding Theory (2nd Edition).

[23] Shneiderman, B., Plaisant, C., Cohen, M., and Jacobs, S. “Designing the User Interface: Strategies for Effective Human-Computer Interaction”. Prentice Hall, 2009.

[24] Donker, A. and Reitsma, P. “Drag-and-drop errors in young children’s use of the mouse.” Interacting with Computers 19, 2, 257–266, 2007.

[25] Gutwin, C., Greenberg, S. “A descriptive framework of workspace awareness for real-time groupware”. Computer Supported Cooperative Work, 11(3), pp. 411-446, 2002.

[26] Couse, L.J. and Chen, D.W. “A Tablet Computer for Young Children? Exploring Its Viability for Early Childhood Education.” Journal of Research on Technology in Education 43, 1, 75–98, 2010.

[27] Hourcade, J.P. “Interaction Design and Children.” Foundations and Trends® in Human-Computer Interaction 1, 4, 277–392, 2007.

[28] Harris, A., Rick, J., Bonnett, V., Yuill, N., Fleck, R., Marshall, P. Rogers, Y. “Around the table: are multiple-touch surfaces better than single-touch for children’s collaborative interactions?” Proc. CSCL’09, 335–344.

[29] Kharrufa, A., Leat, D., and Olivier, P. “Digital mysteries: designing for learning at the tabletop.” Proc. ITS’10, 197–206.

[30] Mansor, E.I., De Angeli, A., and De Bruijn, O. “Little fingers on the tabletop: A usability evaluation in the kindergarten.” 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, 2008, 93–96.

[31] Shoukry, L., Sturm, C., and G.H., G.-E. “Pre-MEGa: A Proposed Framework for the Design and Evaluation of Preschoolers’ Mobile Educational Games.” The International Conference on Engineering Education, Instructional Technology, Assessment, and E-learning, 2012.

[32] Berenson, R., Boyles, G., Weaver, A. “Emotional intelligence as a predictor for success in online learning”. The International Review of Research in Open and Distance Learning, 9(2), 2008.

[33] Han, H., Johnson, S. D. “Relationship between students’ emotional intelligence, social bond, and interactions in online learning”. Educational Technology & Society, 15(1), pp. 78-89, 2012.

[34] Al-Faouri, A. H. A. “Investigating the impact of emotional intelligence on technology learning”. International Journal of Engineering & Technology, 11(3), pp. 58-78, 2011.

[35] Kumar, J. A., Muniandy, B., Yahaya, W. A. J. W. “The relationship between emotional intelligence and students’ attitude towards computers: A study on polytechnic engineering students”. International Journal of Modern Education and Computer Science, 4(9), pp. 14-22, 2012.

[36] Bresó, E., Ferrer, A., Giorgi, G. “MEIT (Mobile Emotional Intelligence Test): Una novedosa metodología para evaluar la percepción de emociones haciendo uso de dispositivos móviles”. Ansiedad y Estrés, 19(2-3), pp. 185-200, 2013.

[37] Lee, K., Joshi, K., Kim, Y. “Person-job fit as a moderator of the relationship between emotional intelligence and job performance”. Proc. SIGMIS CPR’08, pp. 70-75, 2008.

[38] Mazzone, E., Read, J. C., Beale, R. “Design with and for disaffected teenagers”. Proc. NordiCHI’08, pp. 290-297, 2008.

[39] Mayer, J. D., Caruso, D. R., Salovey, P. “Emotional intelligence meets traditional standards for an intelligence”. Intelligence, 27(4), pp. 267-298, 2000.

[40] Nacher, V., Jaen, J., Navarro, E., Catala, A., González, P. “Multi-touch gestures for pre-kindergarten children.” International Journal of Human-Computer Studies 73, 37-51, 2015.

[41] Wainer, A. L., Brooke, R. I. “The use of innovative computer technology for teaching social communication to individuals with autism spectrum disorders”. Research in Autism Spectrum Disorders, 5(1), pp. 96-107, 2011.

[42] Hourcade, J. P., Bullock-Rest, N. E., Hansen, T. E. “Multitouch tablet applications and activities to enhance the social skills of children with autism spectrum disorders”. Personal Ubiquitous Computing, 16(2), pp. 157-168, 2012.

[43] Tartaro, A., Cassell, J. “Playing with virtual peers: Bootstrapping contingent discourse in children with autism”. Proc. ICLS’08, pp. 382-389, 2008.

[44] Bosseler, A., Massaro, D. W. “Development and evaluation of a computer-animated tutor for vocabulary and language learning in children with autism”. Journal of Autism and Developmental Disorders, 33(6), pp. 653-672, 2003.

[45] Davis, M., Dautenhahn, K., Nehaniv, C., Powell, S. D. “TouchStory: Towards an interactive learning environment for helping children with autism to understand narrative”. Computers Helping People with Special Needs, Miesenberger et al., Eds., Springer Berlin-Heidelberg, pp. 785-792, 2006.

[46] Porayska-Pomsta, K., Frauenberger, C., Pain, H., Rajendran, G., Smith, T., Menzies, R., Foster, M. E., Alcorn, A., Wass, S., Bernadini, S., Avramides, K., Keay-Bright, W., Chen, J., Waller, A., Guldberg, K., Good, J., Lemon, O. “Developing technology for autism: an interdisciplinary approach”. Personal and Ubiquitous Computing, 16(2), pp. 117-127, 2012.

[47] Piper, A. M., O’Brien, E., Morris, M. R., Winograd, T. “SIDES: A cooperative tabletop computer game for social skills development”. Proc. CSCW’06, pp. 1-10.

[48] Gal, E., Bauminger, N., Goren-Bar, D., Pianesi, F., Stock, O., Zancanaro, M., (Tamar) Weiss, P. L. “Enhancing social communication of children with high-functioning autism through a co-located interface”. AI & Society, 24(1), pp. 75-84, 2009.