Sistema de Realidad Aumentada para servicios de emergencia

Marc Codina Barberà
Máster universitario en Desarrollo de aplicaciones para dispositivos móviles

Roger Montserrat Ribes

3 de enero de 2018

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Sistema de Realidad Aumentada para servicios de emergencia
Nombre del autor:	Marc Codina Barberà
Nombre del consultor:	Roger Montserrat Ribes
Fecha de entrega (mm/aaaa):	01/2018
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*

Resumen del Trabajo (máximo 250 palabras):

El trabajo realizado forma parte de un proyecto de empresa orientado a facilitar la tarea a los equipos de emergencia en el interior de edificios. Este trabajo se enfoca en el desarrollo de una aplicación para dispositivos Android orientada a la realidad aumentada.

El objetivo es ofrecer información en tiempo real a las personas de emergencia que estén dentro de un edificio. Esta información se proyectará en las pantallas de sus dispositivos de realidad aumentada ofreciendo localización de objetos que puedan ser útiles para desempeñar su tarea, así como información en tiempo real de la posición de sus compañeros.

Además, ofrece la posibilidad de ofrecer a la unidad de control del operativo información de video en tiempo real. Mediante una petición del coordinador del equipo se podrá acceder a las imágenes que están viendo los agentes dentro del edificio facilitando el trabajo y la coordinación con el resto de agentes.

Abstract (in English, 250 words or less):

The work carried out is part of a company project aimed at facilitating the task for emergency teams inside buildings. This work focuses on the development of an application for Android devices oriented towards augmented reality.

The goal is to provide real-time information to emergency responders inside a building. This information will be projected on the screens of their augmented reality devices offering location of objects that may be useful to perform their task, as well as real-time information on the position of their colleagues.

In addition, it offers the possibility of providing the control unit with real-time video information in real time. By means of a request from the team coordinator, it will be possible to access the images that the agents are seeing inside the building, facilitating the work and coordination with the rest of agents

Palabras clave (entre 4 y 8):

Android, Realidad aumentada, Posicionamiento, retransmisión de video

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

La propuesta a desarrollar consiste en un sistema para ayudar a los equipos de emergencia a navegar y monitorizar a sus efectivos dentro de un edificio.

Parte de mi jornada laboral está destinada a la investigación y realización de este proyecto y de ahí que lo proponga como trabajo final de máster. En este proyecto intervienen 3 empresas y un centro de investigación de una universidad

La idea del proyecto nace en un consorcio a nivel europeo para ofrecer a los servicios de emergencia distintas herramientas para facilitar su labor dentro de edificios. A causa de las restricciones económicas de los *partners* europeos, solo el consorcio español consigue financiación y se reduce el alcance del proyecto. [1]

Aunque en el proyecto empresarial se van a realizar pruebas de concepto en resorts hoteleros, a causa de las fechas en la realización de este trabajo, se realizaran las pruebas de concepto en el edificio de uno de las empresas colaboradoras. Es el mismo edificio que se usa para comprobar el estado del proyecto y realizar las reuniones de seguimiento. El objetivo final del proyecto sería poder desplegar el sistema en cualquier edificio que este mapeado para poder obtener información de todo el entorno.

Actualmente los equipos de emergencia utilizan sus sistemas de comunicaciones y mapas del edifico para realizar las tareas de seguimiento del personal y localización de los servicios de emergencia de que disponga el edificio.

Con este trabajo pretendo mostrar parte del trabajo realizado en el proyecto de investigación en el que estoy colaborando, mostrando un sistema de visualización de datos y una nueva forma de comunicación entre los servicios de emergencia.

1.2 Objetivos del Trabajo

- Comunicación entre un servidor y un dispositivo Android para obtención de datos

- Creación de un entorno de realidad aumentada para mostrar los datos obtenidos

- Actualización de los datos y del sistema de visualización en tiempo real.

- Envío de video y audio al centro de mando bajo petición

- Creación de un sistema web para visualizar el flujo de video e interactuar con el audio

- Comunicación Bluetooth entre un sistema Android y un dispositivo electrónico para obtención de datos de sensores de temperatura y humedad para cálculo del choque térmico.

Como objetivos adicionales, el proyecto de empresa debe implementar más funcionalidades que debido al tiempo del trabajo final no será posible desarrollar en el tiempo establecido. A continuación, se exponen otras funcionalidades que deberá tener la aplicación una vez completado el producto:

- Un sistema de navegación en interiores, informando de la ruta más rápida para llegar desde un punto A hasta un punto B

- Información en tiempo real de los sistemas de alarmas dentro del edificio.

Los objetivos marcados son en función del calendario establecido en el proyecto en la empresa. En caso de sufrir modificaciones se indicarán los puntos a cambiar de los objetivos establecidos y se llegará a un consenso con el tutor del trabajo final de master.

Requerimientos funcionales:

- Creación de objetos virtuales sobre el entorno real. La interfaz de usuario permitirá visualizar objetos 3D encima de objetos del mundo real, permitiendo su fácil localización, incluso con paredes entremedio. Esta funcionalidad es básica y por tanto la más importante del sistema.

- Comunicación con dispositivos Bluetooth para obtener información de distintos sensores. Aparte de las gafas de realidad aumentada, el usuario llevará un dispositivo electrónico que transmitirá información de temperatura y humedad a las gafas mediante Bluetooth. Aparte este dispositivo será usado para

realizar la localización en el interior del edificio. Esta tarea tendría una prioridad media, ya que para el correcto funcionamiento se podría prescindir de la conexión Bluetooth.

- Comunicaciones con un sistema de mensajes para enviar y recibir peticiones al centro de control. Esta funcionalidad permite recibir información y peticiones desde el centro de control, así como mandar datos de nuestro punto de vista, para que el centro de control sepa en que dirección estamos mirando y poder usar estos datos para generar una reconstrucción virtual. Esta tarea tiene una prioridad elevada, puesto que es imprescindible tener comunicación con el centro de control.

- Envío de video en tiempo real al centro de control. Desde el centro de control pueden realizar una petición de activar la cámara de nuestro dispositivo. Esta orden llega por medio del sistema de mensajes, con lo cual esta función es dependiente del funcionamiento del sistema de mensajes.

- Aplicación web para realizar una petición de video y mostrarlo. Para que desde el centro de control puedan visualizar nuestra cámara, necesitan reproducir el video que recibirán. Esta funcionalidad depende de la función de video con lo cual será la última funcionalidad a implementar.

Requerimientos no funcionales:

- Interfaz limpia, mostrando solo los datos necesarios para el usuario. La aplicación realizará muchas operaciones de forma automatizada que no será necesario mostrar al usuario. Estará conectada al sistema de mensajes por el que llegarán muchos datos que no estarán dirigidos a un usuario especifico por lo tanto habrá que mostrar solo los datos que van dirigidos al usuario siempre que no se puedan usar estos datos para realizar la tarea de forma automática.

- Obtención de tiempos de refresco de datos los más próximos posibles a tiempo real. Dependerá de los sistemas asociados al procesado de datos. Cada cierto tiempo se actualizará la posición del usuario, así como la posición de todos sus compañeros en la escena. Cuanto más actualicemos los datos más precisión conseguiremos en situar a todos los agentes en la escena 3D

1.3 Enfoque y método seguido

Al ser un proyecto colaborativo entre distintas empresas se ha elegido utilizar la tecnología ya usada en sus productos para la realización del trabajo. Parte de esta tecnología se ha tenido que modificar para poder cumplir con los requisitos del proyecto.

La parte de aplicación móvil y realidad aumentada, en la que nos centraremos en este trabajo final de máster, se generará desde cero. Se han valorado distintas formas para el desarrollo y se han descartado después de investigar las distintas tecnologías para implementar AR y valorar la dificultad para desarrollar aplicaciones con estas *API*. Finalmente se ha optado por realizar una implementación desde cero usando la librearía gráfica OpenGL [2]. En el apartado de estado del arte de la tecnología explicaré las distintas tecnologías valoradas.

1.4 Planificación del Trabajo

El trabajo de desarrollo de la aplicación Android se realizará mediante el IDE Android Studio. Para poder reproducir el video en web se intentará usar HTML5 y JavaScript. La implementación del entorno AR, como ya se ha comentado, se utilizará OpenGL. El coste del software para implementar la aplicación será de cero euros.

En cuanto al hardware usado. Los colaboradores del proyecto ofrecen su sistema de hardware gratuitamente para poder desarrollar todo el sistema. Como dispositivo para visualización de AR se han comprado unas gafas Epson BT-300 cuyo coste es de 850€ y ha sido pagado por el centro en el que estoy trabajando. Aparte, se usará tanto el dispositivo smartphone personal como el de algún compañero para comprobar el funcionamiento del sistema con distintos dispositivos de forma simultánea.

PEC1 – Plan de trabajo: 20 de setiembre al 11 de octubre

- Propuesta de trabajo a realizar. (5 horas)
- Búsqueda de información sobre el trabajo propuesto. (15Horas)
- Objetivos y alcance del proyecto. (5 horas)
- Planificación de las tareas a realizar según el calendario de entregas. (10 horas)
- Realización de la documentación a entregar en el hito 1. (10 horas)

PEC2 – Diseño: 12 de octubre al 1 de noviembre

- Análisis de usuario. (15 Horas)
- Diagrama de casos de uso. (20 Horas)
- Diseño de la arquitectura. (15 Horas)

- Diseño del prototipo en alto nivel (15 Horas)
- Documentación a entregar para el hito 2 (10 horas)

PEC3 – Implementación: del 2 de noviembre al 13 de diciembre

- Codificación de la aplicación (70 Horas)
- Pruebas de funcionamiento de la aplicación (15 Horas)
- Depuración de errores en la aplicación (10 Horas)
- Pruebas de funcionamiento entre la aplicación y el servicio de recepción de video (20 Horas)
- Depuración de errores en la comunicación entre el servidor de datos y el dispositivo (10 Horas)
- Realización de la documentación a entregar en el hito 3 (15 Horas)

PEC4 – Entrega final: del 14 de diciembre al 3 de enero

- Revisión de la memoria final. (25 Horas)
- Entrega de la memoria final. (2 Horas)
- Entrega de la aplicación desarrollada. (2 Horas)
- Elaboración de la presentación. (25 Horas)
- Entrega del video de presentación del trabajo realizado. (2 Horas)

Para la realización de este trabajo se estima una dedicación de 4 horas diarias en días laborables, dedicando los fines de semana 4 horas para realizar la documentación, pudiendo añadir horas en fines de semana para el diseño y desarrollo del trabajo en caso de ser necesario. La estimación de horas totales para la realización del trabajo es de:

	Fecha de inicio	Fecha de fin	Horas
PEC 1	20/09/2017	11/10/2017	45
PEC 2	12/10/2017	1/11/2017	75
PEC 3	2/11/2017	13/12/2017	140
PEC 4	14/12/2017	3/01/2018	56
		Horas totales	316

A continuación, se puede ver el diagrama de Gantt, marcando los hitos de entregas y la estimación del diseño y desarrollo del trabajo a realizar:

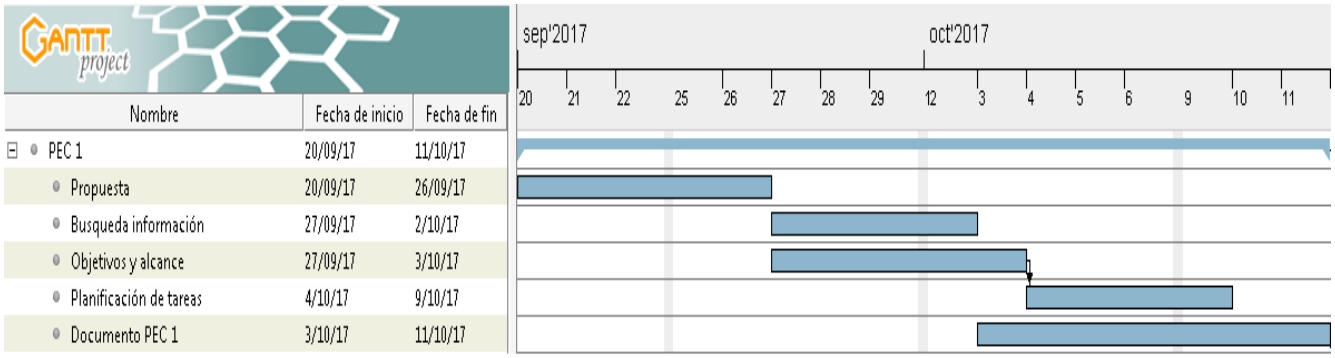

Ilustración 1 - Gantt PEC1

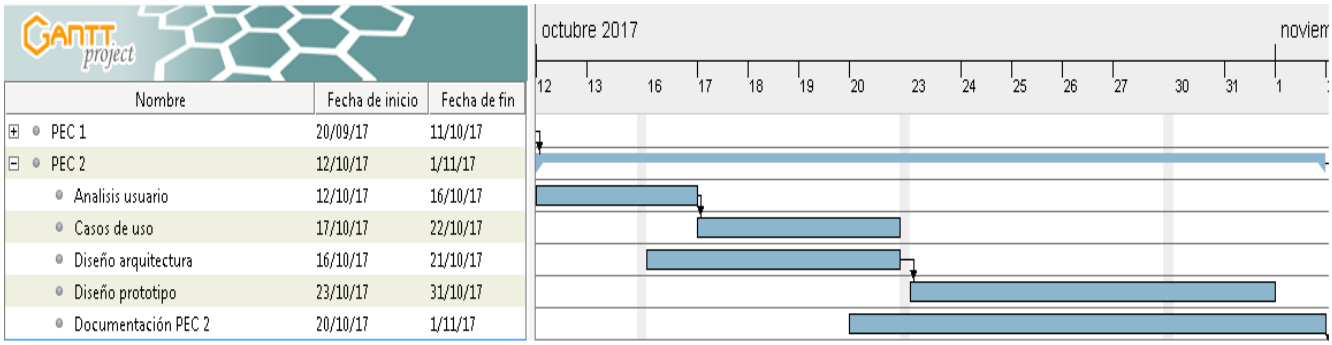

Ilustración 2 - Gantt PEC2

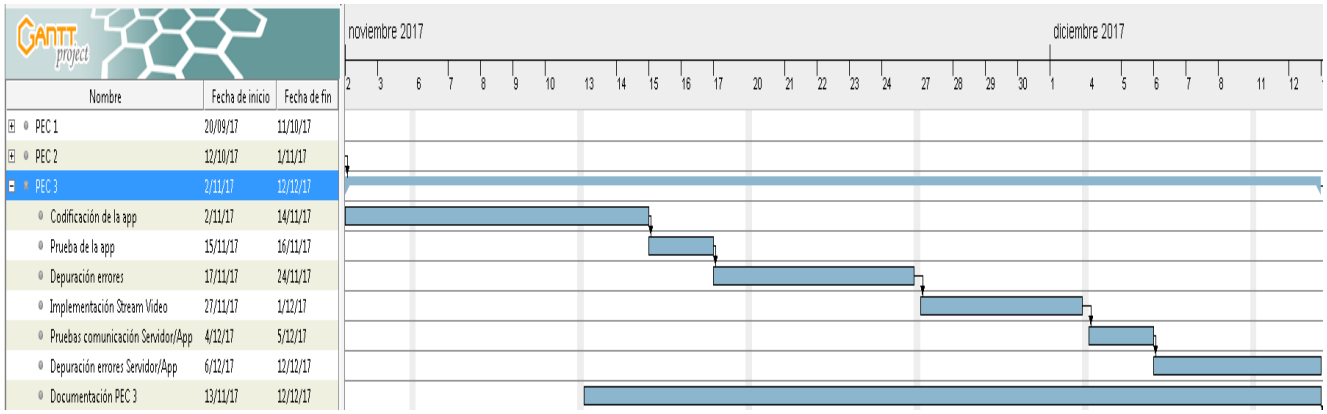

Ilustración 3 - Gantt PEC3

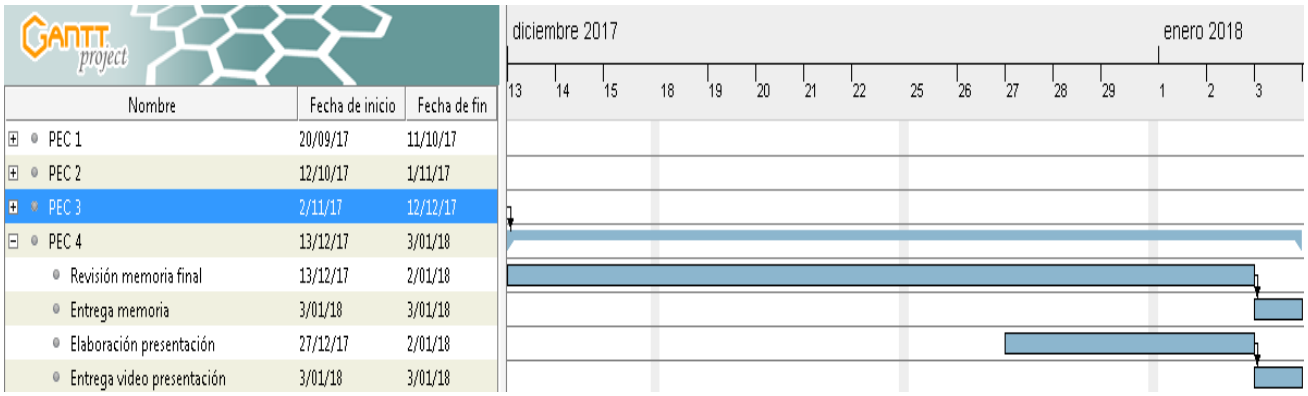

Ilustración 4 - Gantt PEC4

1.5 Breve sumario de productos obtenidos

A la finalización de este trabajo se espera tener una aplicación operativa cumpliendo con los objetivos acordados al inicio de esta memoria.

Al ser un proyecto que se desarrollará en paralelo al proyecto empresarial, se han marcado los objetivos acordes con lo establecido en el calendario de entregas del proyecto. En los próximos capítulos, sin entrar en detalle, se explicarán los conceptos usados para la realización

y ejecución de todo el proyecto, explicando por encima las partes que están desarrollando las empresas colaboradoras.

- Aplicación Android operativa en unas gafas AR. Comunicación con los servicios desarrollados por las empresas colaboradoras y visualización de un entorno AR para el posicionamiento de objetos y personas en interiores.

- Pequeño servicio web listo para integrarse en el sistema global del proyecto, para mostrar el funcionamiento del envío de video desde los dispositivos AR a un navegador web.

- Ficheros de código fuente y manual de compilación del proyecto

1.6 Breve descripción de los otros capítulos de la memoria

Este documento incluye los siguientes apartados:

- Capítulo 1, se explica el Trabajo a realizar mediante una introducción de la idea a desarrollar. A continuación, se exponen los objetivos del Trabajo a desarrollar. Posteriormente se realiza una explicación sobre la forma de enfocar el problema y termina con una planificación del Trabajo.

- Capítulo 2, se explicarán las distintas tecnologías existentes en el mercado y las razones para escoger la tecnología usada finalmente.

- Capítulo 3, este capítulo estará enfocado en comentar el diseño de la aplicación, la arquitectura del sistema y sus casos de uso. El capítulo anterior y este se completarán en el hito 2

- Capítulo 4, todo lo referente a la implementación de la solución propuesta y las pruebas realizadas estarán documentadas en este capítulo. Este capítulo se completará en el hito 3

- Capítulo 5 con las conclusiones del trabajo realizado y las posibles mejoras futuras que se pueden llevar a cabo. Este capítulo se completará en el hito 4.

2. Estado del arte

El objetivo del proyecto es la visualización de objetos virtuales superpuestos en objetos reales. Para poder llevar a cabo este objetivo se necesita de un dispositivo capaz de mostrar estos dos objetos.

Una primera aproximación sería usar una imagen de video. Capturando la imagen real y añadiendo mediante procesado un objeto virtual, podríamos alterar la realidad. Este método se descartó al añadir la necesidad de capturar mediante video la imagen y procesarla, produciendo un posible retardo en la visualización posterior del usuario. Esto provocaría que un usuario podría estar viendo una imagen de video anterior al estado actual.

Se pensó en usar la tecnología de Realidad Virtual, pero esto provoca que haya que generar todo el escenario en 3D de forma fiel, para que el usuario pueda interactuar con de forma segura con el mundo real. Esto se ha descartado ya que es imposible realizar una reconstrucción fiel del escenario donde se actuará, ya que pueden realizarse cambios de mobiliario o puede aparecer una persona en mitad del camino y no será posible contemplarlo en un mundo virtual.

Finalmente se ha decantado por un sistema de realidad aumentada que permita ver el mundo real y añadir una capa virtual encima. Para poder trabajar en condiciones de poca visibilidad se ha optado por generar un escenario en 3D que posicione en un sistema de coordenadas virtual los objetos de toda la escena. Mediante posicionamiento podremos saber en todo momento en que punto está el usuario y representar en el mundo virtual la escena que está viendo. Para ello se ha propuesto el uso de unas gafas de realidad aumentada. Esta tecnología permite proyectar en el cristal de las gafas información sin entorpecer la visión del mundo real.

Se ha realizado el estudio de distintos dispositivos de realidad aumentada. A continuación, se muestran las gafas estudiadas y la decisión final tomada.

2.1 Gafas AR

2.1.1 Meta 2 [3]

Ilustración 5 - Gafas Meta2

Todo y tener unas especificaciones muy buenas: resolución de 2550x1440, 4 altavoces, cámara HD, etc. Este dispositivo quedo descartado por la necesidad de tener un ordenador conectado para poder realizar el control y proyección de datos en las gafas.

2.1.2 Epson Moverio BT-200 [4]

Ilustración 6 - Gafas Moverio BT-200

Segunda versión de las gafas de Epson. Especificaciones un poco antiguas, pero con disponibilidad en el mercado. Se descartó por no tener Bluetooth 4.0, básico para comunicación con la electrónica externa.

2.1.3 ODG R-7 [5]

Ilustración 7 - Gafas ODG R-7

Este modelo cumplía con todos los requerimientos de hardware, aunque no estaba claro si funcionaria de forma autónoma o mediante conexión a un ordenador. Descartado por un precio muy elevado y todavía no disponible en el mercado en el momento de la compra.

2.1.4 Holo Lens [6]

Ilustración 8 - Holo Lens

Posiblemente las mejores gafas de realidad aumentada del mercado, pero su precio y la dificultad de comprarlas desde España hicieron que fueran descartadas. Aparte necesita un ordenador con unas versiones específicas de software y sistema para poder realizar el desarrollo, esto provoca un encarecimiento del producto al tener que comprar licencias de software.

2.1.5 Epson Moverio BT-300 [7]

Ilustración 9 - Gafas Epson BT-300

Este dispositivo ha sido el escogido finalmente para la implementación del proyecto. Cumple los requisitos necesarios: IMU de 9 ejes, cámara frontal, posibilidad de audio mediante cascos, Bluetooth 4.0. Se programa directamente con Android 5.1 y tiene la posibilidad de usar una librearía propia de Epson para controlar algunas de las funcionalidades que incorpora.

2.2 Escenario 3D

Para poder crear la escena en 3D y mostrar los objetos hace falta un motor gráfico para poder dibujarlos. Se han barajado distintas opciones.

2.2.1 Unity [8]

Este motor gráfico otorga una gran facilidad en el momento de generar nuestro entorno 3D. Se programa en C# y permite generar una aplicación para Android. Se descartó por la necesidad de tener una aplicación Android que permitiera aparte de recrear contenido 3D, trabajar con sensores, comunicaciones de red y Bluetooth. Ante la complejidad de aprender a usar un nuevo programa como es Unity y aprender un lenguaje nuevo como es el C# se terminó descartado.

2.2.2 Ogre3D [9]

Motor gráfico multiplataforma que permite desarrollar para Android. Se programa en C++. Al igual que el motor Unity se descartó por la complejidad de desarrollar en C++ para Android a la vez que se realizaba una aplicación en Java.

2.2.3 OpenGL

La mejor opción para desarrollar directamente en Java desde el IDE de Android. La API de Android ya incorpora OpenGL y es relativamente fácil de empezar a crear un escenario 3D. Aunque es una buena opción se terminó descantando.

2.2.4 JPCT [10]

Este motor gráfico ha sido el escogido para la realización del proyecto. Implementación de OpenGL en Java de alto nivel. Permite generar contenido OpenGL de forma muy sencilla.

2.3 Sistema de mensajería

El bus de mensajes está implementado usando el sistema RabbitMQ [11], es agente de mensajes de código abierto que implementa el estándar AMQP y permite la de negociación de mensajes. Este sistema permite que un cliente se conecte a una cola de mensajes, de esta forma será capaz de enviar mensajes a la cola y ver todos los mensajes que pasan por ella.

3. Diseño

3.1 Diseño centrado en el usuario

3.1.1 Usuarios y contexto de uso

La aplicación que se quiere desarrollar está enfocada a los servicios de emergencia. Al ser un sector muy específico limita los escenarios en los que se pueda utilizar. La aplicación al estar orientada a un conjunto muy pequeño y acotado de la población, se ha realizado un único perfil de usuario.

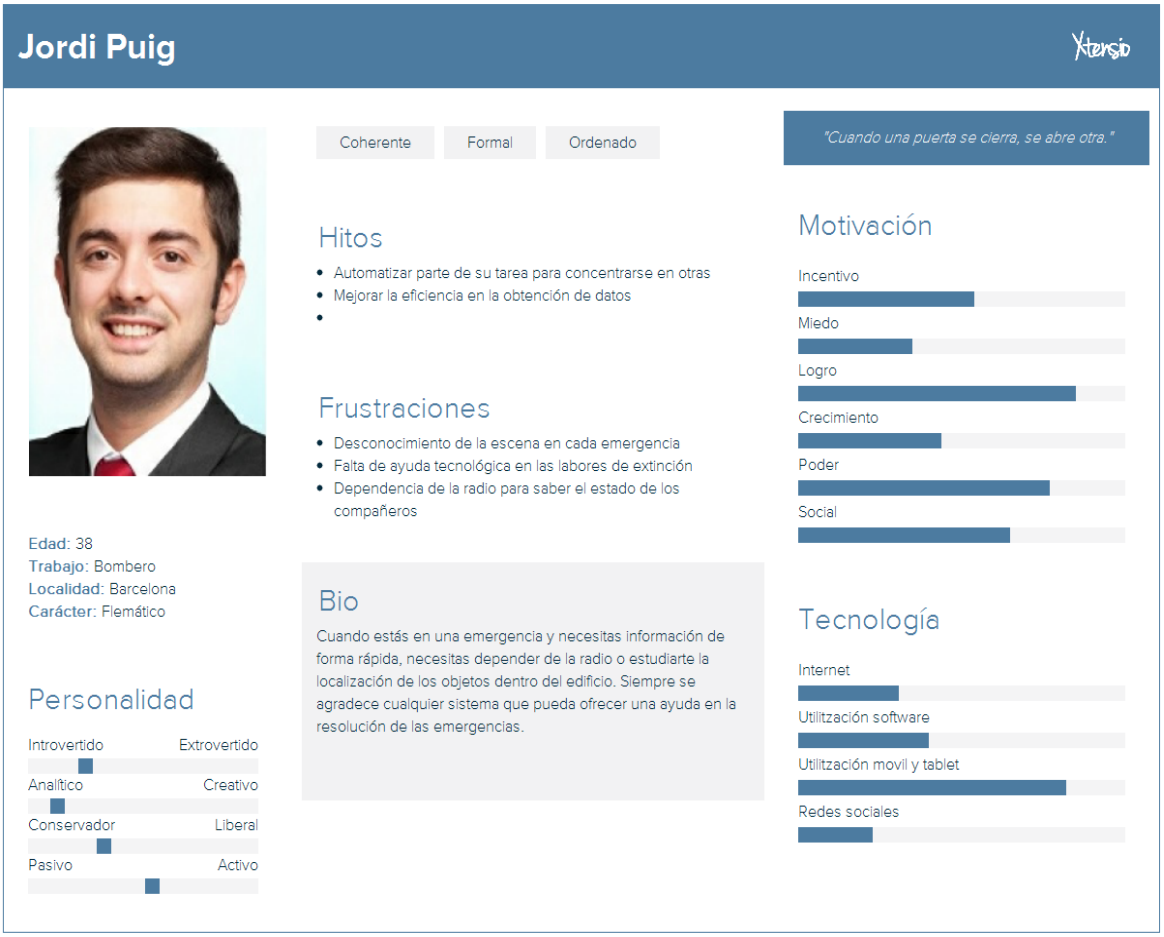

Ilustración 10 - Persona

3.1.2. Diseño conceptual

A continuación, puede verse un escenario para el perfil persona creado anteriormente.

Escenario 1

Llega una llamada de emergencia a la estación de bomberos, hay un fuego en un hotel. Como de costumbre se preparan lo más rápido posible, mientras los jefes de equipo que están en el centro de control preparan la documentación para poder planificar la mejor estrategia.
Tener a mano los planos del edificio, posibles rutas hacia las salidas de emergencia más próximas en los puntos donde está el fuego.

Desde hace unos días están en pruebas de un nuevo sistema de navegación en interiores. Este sistema les permite ver objetos dentro del edificio en condiciones de poca visibilidad como puede ser a oscuras o con mucho humo. Aparte permite generar avisos desde el centro de control, planificar rutas en tiempo real y permite mostrar al centro de control lo que está pasando en el interior del edificio.

3.1.3. Prototipo

El prototipo se basa en la realización de un modelo de la interfaz de la aplicación que se desarrollará. Este modelo permite evaluar las decisiones de diseño, permitiendo su modificación de forma sencilla.

Cuando se inicia la aplicación la primera pantalla que verá el usuario la configuración del dispositivo Bluetooth. Este paso es obligatorio para el correcto funcionamiento de la aplicación. El dispositivo bluetooth con el que nos emparejaremos transmite el identificador que tendrá el usuario, aparte da información de temperatura y humedad cada cierto tiempo y es el dispositivo que permitirá al usuario estar localizado en el interior del edificio.

La pantalla estará formada por una lista de ítems, cada uno de los ítems será un dispositivo bluetooth encontrado a nuestro alrededor. Cada ítem tendrá un botón para emparejarse con el dispositivo de realidad aumentada. En la parte superior derecha un botón permitirá volver a escanear los dispositivos a nuestro alrededor.

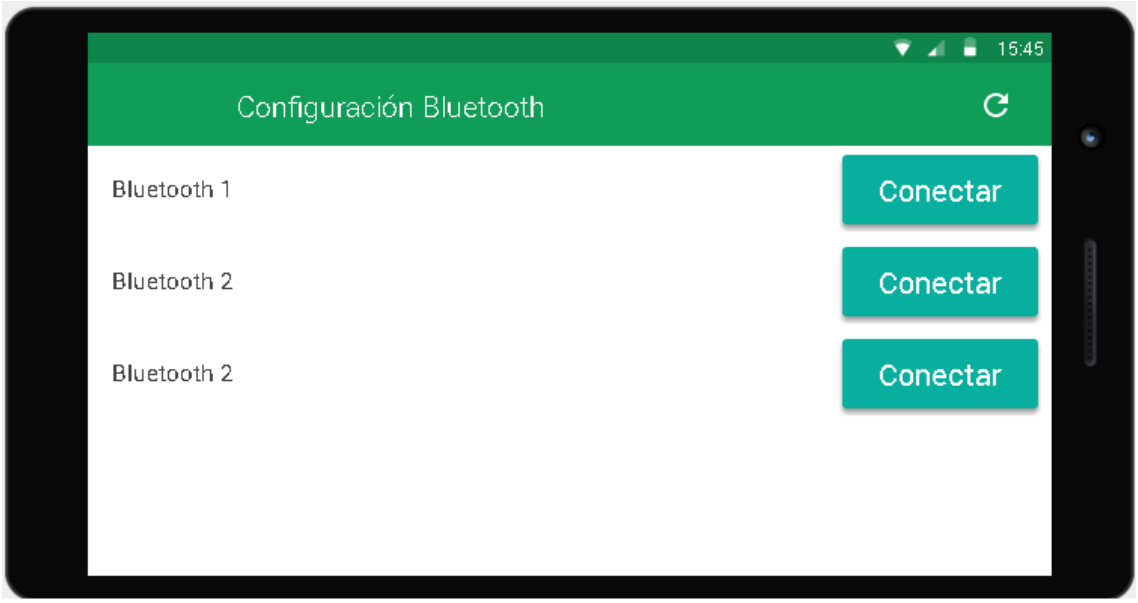

Ilustración 11 - Pantalla configuración Bluetooth

Una vez realizado el emparejamiento se cargará la pantalla de configuración. En esta pantalla el usuario podrá configurar la velocidad de refresco de la escena y los colores que aparecen en ella, para evitar problemas en la visualización de los colores.

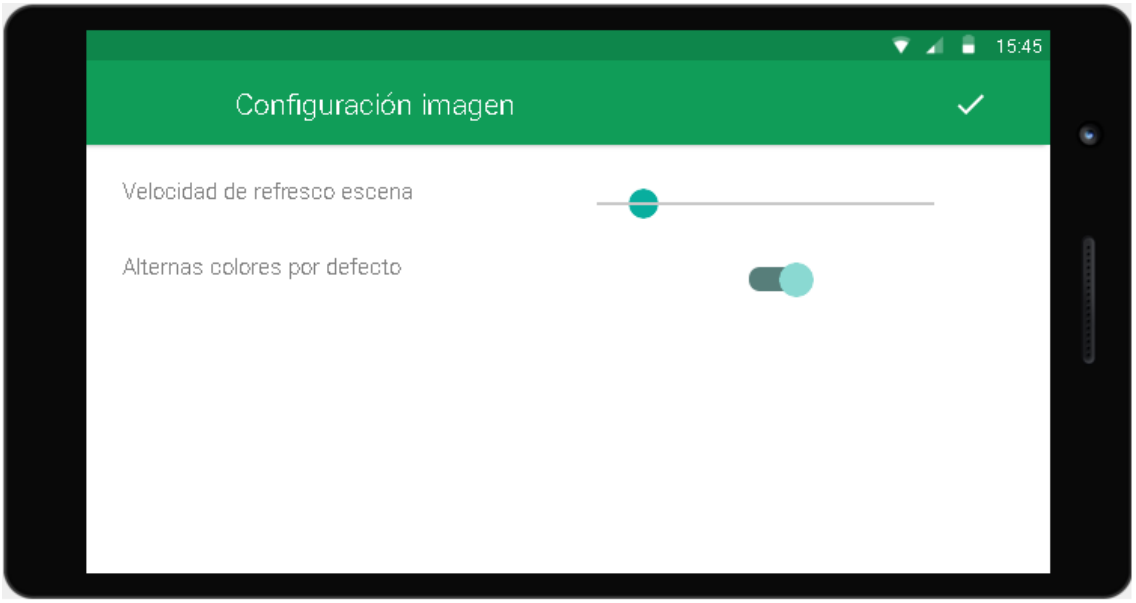

Ilustración 12 - Pantalla de configuración

La última pantalla es la de navegación. En esta pantalla el usuario podrá ver la escena 3D reconstruida. A medida que el usuario se desplace por el edificio se irá actualizando su punto de vista. Aparte, usando los sensores del dispositivo el usuario podrá ver la escena en 360º pudiendo rotar sobre si mismo para ver lo todos los objetos que hay a su alrededor.

En la parte superior de la pantalla el usuario podrá ver distinta información que puedan ser de utilidad, como puede ser la temperatura y humedad a su alrededor, si desde el centro de control han activado su cámara. Aparte, aparecerán mensajes de aviso sobreimpresos informando de alguna acción que deba llevar a cabo.

En la siguiente ilustración se puede ver cómo queda representada una escena 3D. En este caso la cámara está situada en el exterior y se pueden apreciar las 3 plantas del edificio y sus distintas habitaciones. Las paredes se recrean con una altura de medio metro para no obstaculizar la visión del usuario. Los objetos están situados en las posiciones reales dentro del edificio. Con el fondo en negro al usarlo en unas gafas de AR, se puede ver a través del cristal la realidad con lo que conseguimos que se proyecte la imagen de los objetos encima de las paredes reales facilitando la identificación y la distancia a la que se encuentran.

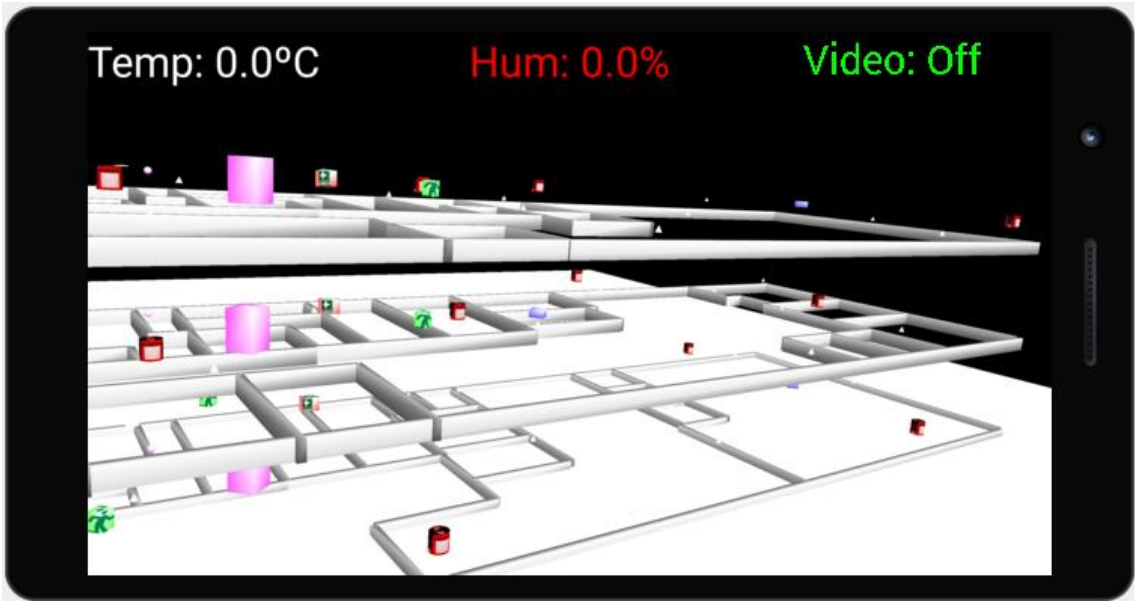

Ilustración 13 - Pantalla escena 3D

3.1.4. Evaluación

En el diseño del prototipo se ha intentado simplificar al máximo la interfaz, dejando una actividad para cada una de las funciones que se puedan realizar. El funcionamiento de la aplicación es sencillo y se ha focalizado en tener una buena usabilidad sin mostrar más información de la necesaria en cada pantalla.

Una vez se desarrolle la aplicación se podrá valorar de forma más certera si el prototipo diseñado cumple con los criterios de usabilidad.

3.2 Definición de los casos de uso

3.2.1 Diagrama de flujo

En la siguiente figura se puede ver el diagrama de flujo de ejecución de la aplicación. Se detallan los procesos y decisiones que tomará el sistema o el usuario interactuando con la interfaz

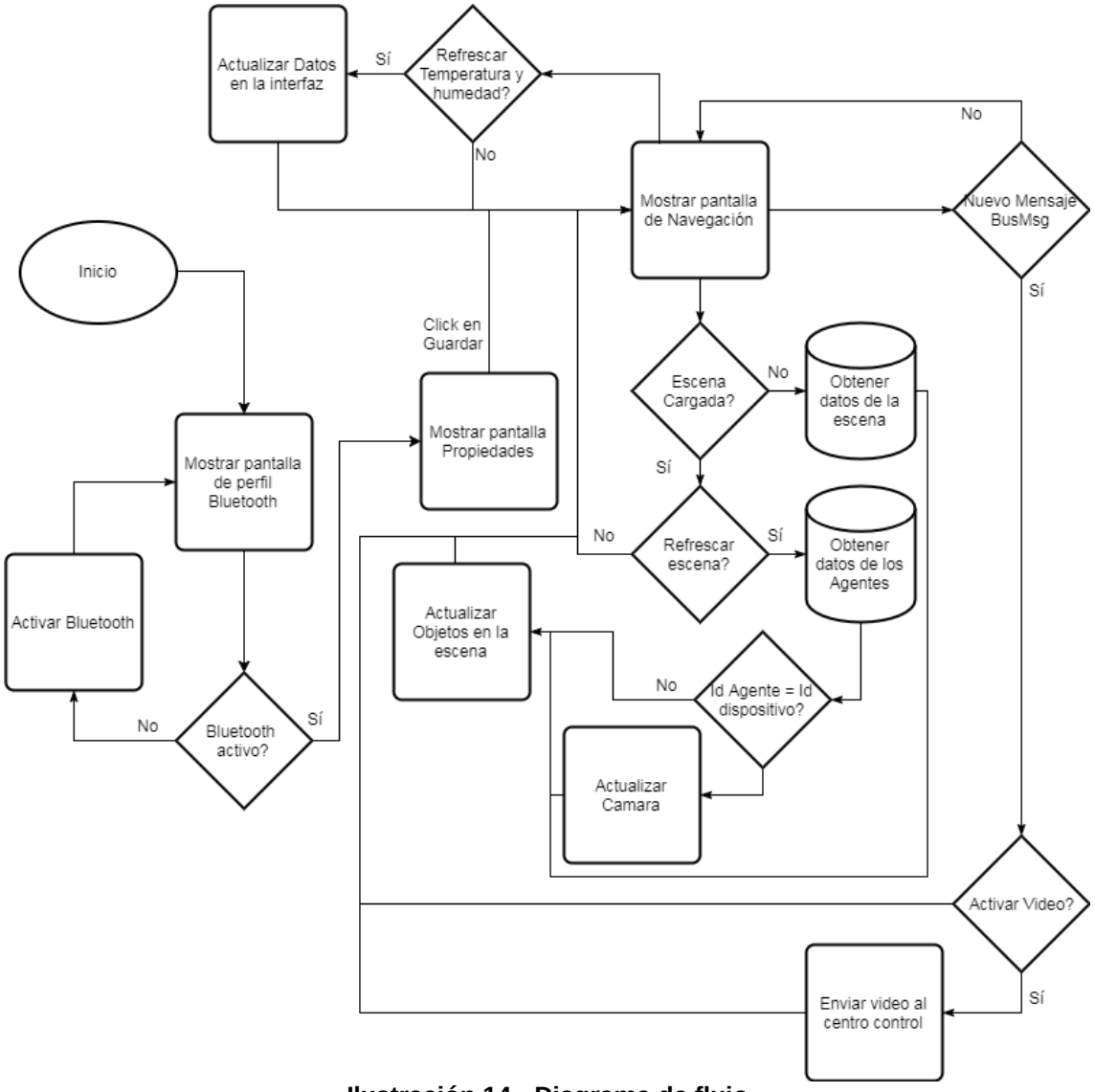

Ilustración 14 - Diagrama de flujo

En el diagrama, figura 14, pueden verse representadas las 3 pantallas que componen la aplicación. Perfil Bluetooth, Propiedades y Navegación. También puede verse el recorrido que realiza el usuario desde que inicia la aplicación. Primero debe pasar por la pantalla de perfil, para conectarse con la electrónica externa, que ofrece el identificador de usuario y la temperatura. En el caso de no tener el

Bluetooth activado hay que permitir activarlo para poder realizar la comunicación. A continuación, Mostramos la pantalla de propiedades, para realizar pequeños cambios en la forma de mostrar los datos de la aplicación.

La siguiente pantalla que se muestra es la de navegación. En este punto entramos en el bucle principal de la aplicación. Se generarán el escenario y los objetos 3D que se muestran en la pantalla de navegación. Se deberá de comprobar si llegan mensajes para el dispositivo. Estos mensajes realizarán una petición de video para iniciar una retransmisión hacia el centro de control. Por último, hay que comprobar si hay que realizar una actualización de los datos que nos llegan pro Bluetooth.

3.2.2 Diagrama de casos de uso

En la ilustración 15 se puede ver el diagrama de casos de uso de la aplicación a desarrollar.

En el diagrama puede verse a dos usuarios, el primero hace referencia a cuando se inicia la aplicación. Una vez completado el perfil ya seriamos un usuario definido en el sistema.

En el caso de usuario definido solo tenemos un único tipo de usuario. El resto de casos son exclusivos de este usuario definido. En un futuro se podría crear perfiles diferentes para los usuarios en el caso de que los agentes de emergencias realizaran tareas específicas diferenciadas en un incendio.

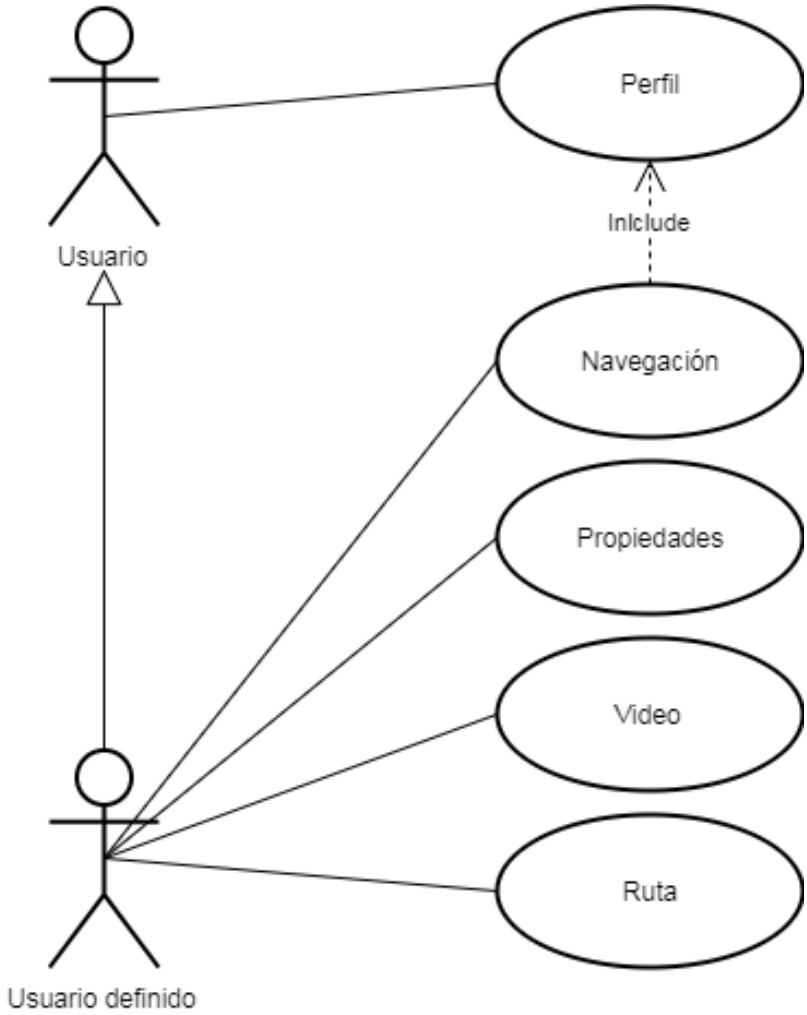

Ilustración 15 - Diagrama de casos de uso

3.2.3 Casos de uso

Caso de Uso: Perfil
Descripción: Permite asignar un identificador al usuario
Actores: Todos
Precondiciones: -
Flujo: 1. El usuario pulsa el botón para realizar una conexión a la electrónica externa 2. Una vez creada la conexión se pasa a la actividad de propiedades
Flujo alternativo: -
Post condición: Se muestra la pantalla de propiedades

Caso de Uso: Propiedades
Descripción: Esta pantalla permite realizar una pequeña configuración de la aplicación
Actores: Todos
Precondiciones: Perfil de usuario
Flujo: 1. El usuario puede seleccionar si es daltónico, para cambiar la composición de colores. 2. Cambiar velocidad de actualización de la vista
Flujo alternativo: -
Post condición: Se muestra la pantalla principal de navegación

Caso de Uso: Navegación
Descripción: Actividad que muestra la recreación 3D del edificio y objetos
Actores: Todos
Precondiciones: El usuario tiene una conexión activa con la electrónica externa
Flujo: 1. Mediante el movimiento del usuario se actualizará su punto de vista en la escena.

Flujo alternativo: 1. No hay conexión a la red de datos - Se muestra un mensaje de error informando del problema
Post condición: -

Caso de Uso: Video
Descripción: En este caso, el sistema activa la cámara del dispositivo e inicia un envío hacia el centro de control
Actores: Todos
Precondiciones: Estar en la pantalla de navegación, para poder recibir las peticiones del centro de control.
Flujo: 1. Se realiza una petición desde el centro de control para activar la cámara del usuario. 2. La cámara inicia el envío de forma automática.
Flujo alternativo: -
Post condición: Se realiza una conexión al centro de control para enviar el video

Caso de Uso: Ruta
Descripción: Este proceso crea una ruta virtual sobre la escena, indicando el punto al que debe dirigirse el agente
Actores: Todos
Precondiciones: Estar en la pantalla de navegación, para poder recibir las peticiones del centro de control.
Flujo: 1. Se realiza una petición desde el centro de control para iniciar una nueva ruta. 2. El sistema genera en la escena los puntos por los que tiene que pasar el usuario para poder llegar a destino.
Flujo alternativo: -
Post condición: -

3.3 Diseño de la arquitectura

El diseño de la arquitectura sirve para identificar las clases y objetos que se utilizarán para gestionar los distintos procesos y la estructura de la API que servirá para realizar las peticiones al servidor.

En este proyecto, al formar parte de un proyecto empresarial, solo se implementa parte de la arquitectura. En la siguiente ilustración puede verse como es la arquitectura del sistema en el proyecto empresarial.

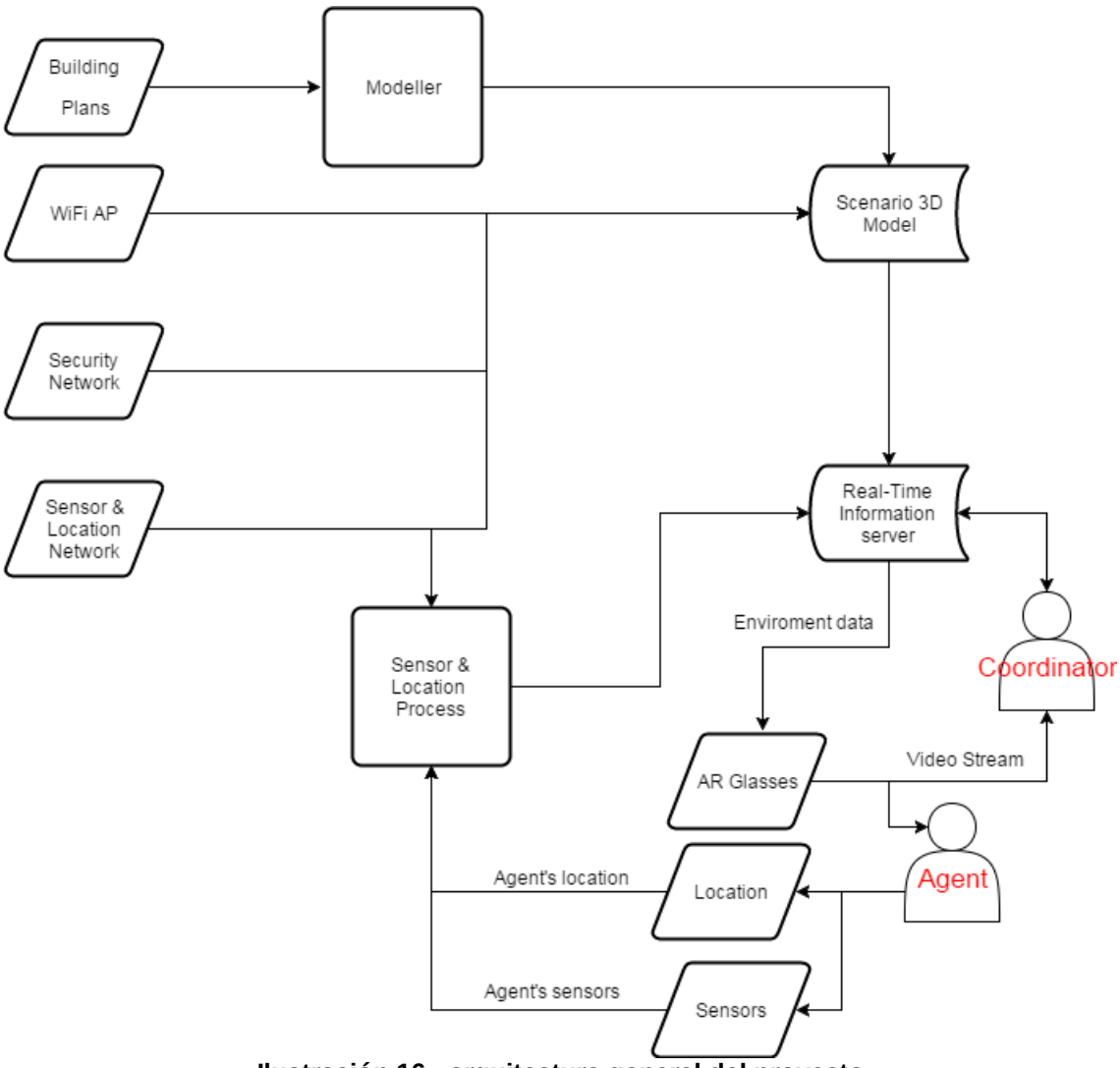

Ilustración 16 - arquitectura general del proyecto

Como se puede apreciar, hay una serie de puntos que no están tratados en este documento y que forman parte del trabajo realizado por las otras empresas dentro del proyecto empresarial.

El proceso Modeller extrae de los planos del edificio todos los puntos necesarios para generar un modelo en 3D. Estos puntos serán los

usados posteriormente por la aplicación desarrollada en este documento.

El siguiente proceso es el de Sensors & Location. Este se encarga de recoger la localización de las electrónicas que acompañan a los agentes de emergencias. Estas electrónicas son las que mandan los datos de temperatura y humedad al dispositivo del agente mediante Bluetooth.

Para poder realizar la localización se distribuyen en los alrededores del edificio unas balizas portátiles. En caso que el edificio que tenga instalado un sistema de balizas no será necesario desplegar el sistema portátil. Mediante el sistema de notificación del Bluetooth se triangula la señal que reciben las balizas. De esta forma se puede saber a qué distancia de cada antena está un dispositivo y se puede realizar una localización de forma bastante precisa.

Con los datos de posición de las balizas, la información de los puntos de acceso wifi, los distintos sensores de alarma que pueda tener el edificio y los datos generados por el Modeller, se crea un escenario 3D.

Los datos de los sensores, del escenario 3D y los datos generados por la electrónica del agente se inyectan en un sistema de mensajes en tiempo real. Este sistema permite ir generando en tiempo real los avisos que podrán ver los agentes en sus dispositivos de realidad aumentada y permite al centro de control monitorizar todo lo que sucede dentro del edificio.

En este proyecto no interviene una base de datos directamente ya que la parte que se está implementando es la de visualización y generación de datos. En el siguiente grafico puede observarse con más detalle los procesos que intervienen en las gafas.

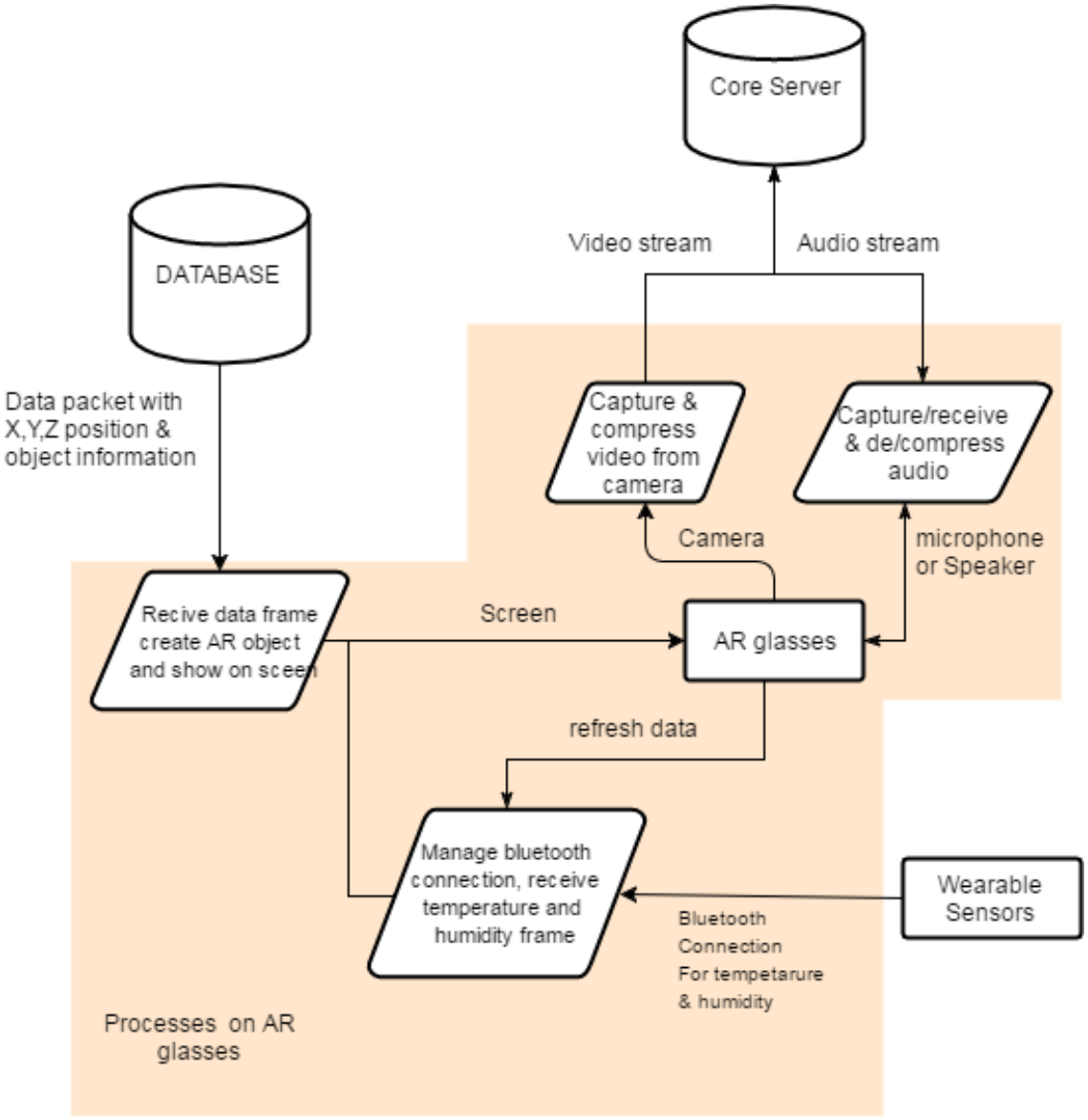

Ilustración 17 - Procesos del dispositivo de Realidad Aumentada

En la ilustración anterior se puede observar como el dispositivo de realidad aumentada realiza reconstrucción de la escena y la visualización bajo petición a la base de datos, muestra los datos recibidos a través de la comunicación Bluetooth y realiza una retransmisión de video y audio hacia un servidor bajo petición.

3.3.1 Diagrama UML correspondiente a las entidades y clases

En el siguiente diagrama se muestra las principales clases de la aplicación.

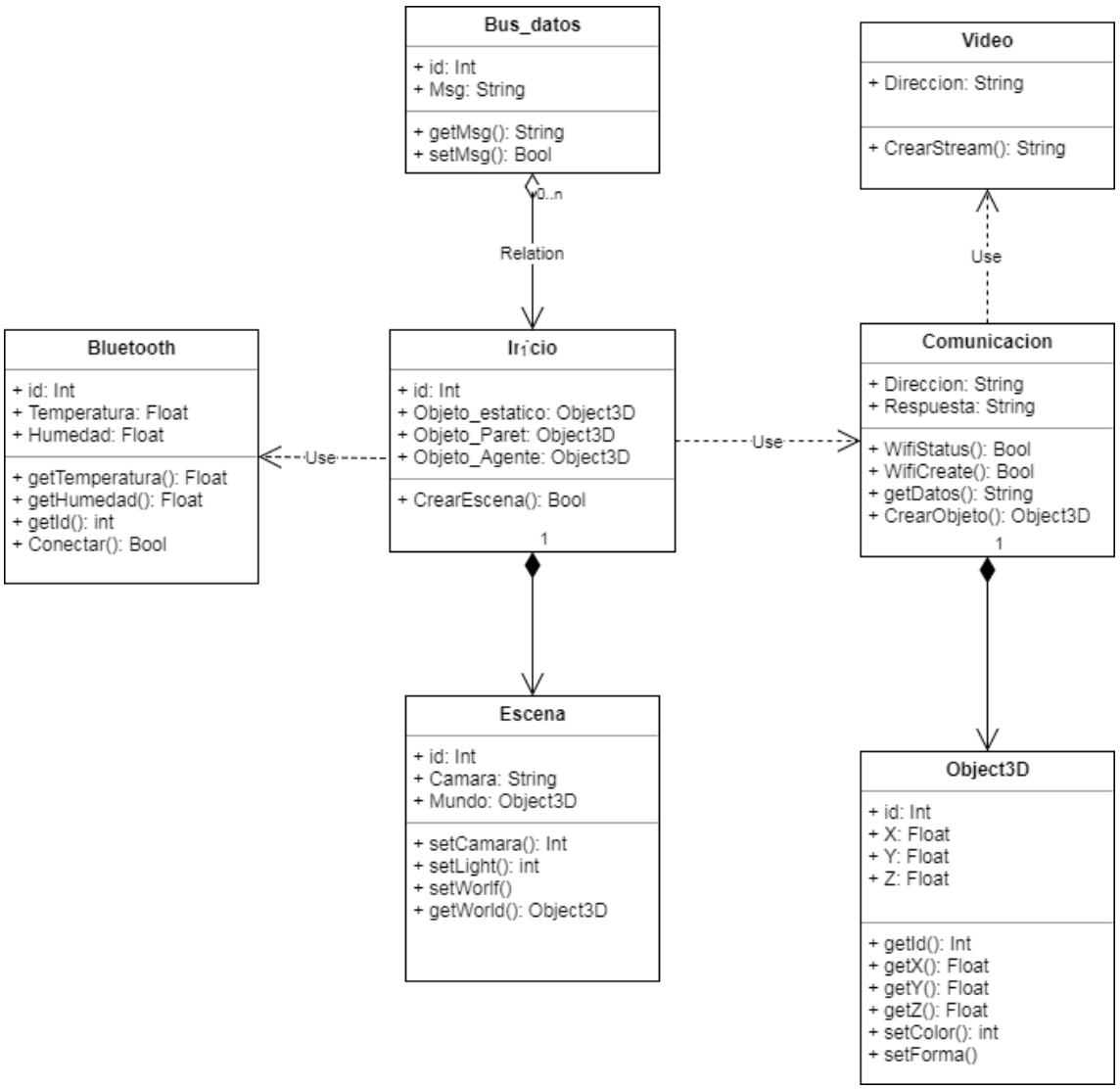

Ilustración 18 - Diagrama de clases

3.3.2 Diagrama de la arquitectura del sistema

El diagrama de la arquitectura servirá para tener una visión simplificada del sistema. Se ha optado por la arquitectura Modelo-Vista-Controlador(MVC). En la ilustración 19 puede verse un diagrama del sistema.

Modelo: Está parte es la encargada de obtener los datos requeridos por la aplicación. Es el encargado de comunicarse con la base de datos y retornar la información requerida. Aunque en este proyecto no se realiza una implementación de una base de datos, tal como se ha visto en la ilustración 17, el proyecto empresarial sí que la implementa y se realizan peticiones para obtener los datos a visualizar.

Aparte de comunicarse con la base de datos, también recibirá datos del bus de mensajes, por ejemplo, mensajes de alerta o activación de video. Y será el encargado de recibir los datos de temperatura y humedad procedentes del dispositivo Bluetooth.

Vista: Esta parte es la encargada de mostrar la interfaz al usuario. Mostrará los datos obtenidos de la base de datos, el dispositivo Bluetooth o el bus de mensajes. Mostrará la escena 3D, la temperatura y humedad e información que pueda llegar desde el centro de control.

Controlador: Es la parte encargada de comunicar la interfaz de usuario con la base de datos. En esta parte se realizará la lógica de la aplicación y la reconstrucción de la escena 3D.

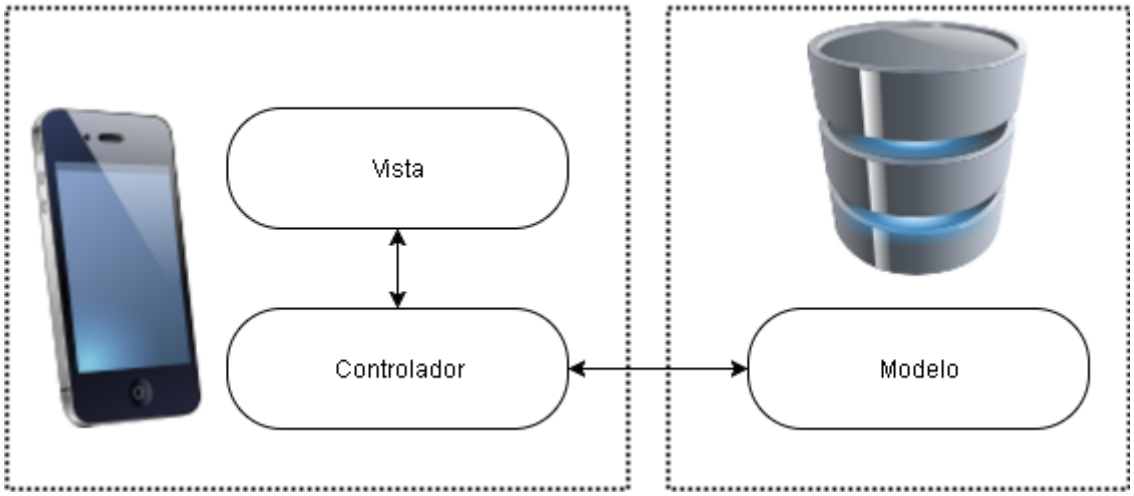

Ilustración 19 - Arquitectura del sistema

4. Implementación

En este apartado se comentará el proceso llevado a cabo para realizar la implementación de la aplicación. Como se ha comentado en distintas ocasiones a lo largo de la memoria, la aplicación que se está desarrollando en este trabajo forma parte de un proyecto de empresa en el que distintos elementos externos interactúan para enviar o recibir datos de la aplicación. Puesto que hay partes del proyecto que están bajo un acuerdo de confidencialidad y la funcionalidad de todo el proyecto es dependiente del trabajo de otras empresas, en esta memoria se explicarán las implementaciones llevadas a cabo en la aplicación y los pasos que se han seguido para completar el trabajo.

En la versión de la aplicación compilada para este proyecto de máster se realizarán unas pequeñas modificaciones para poder ejecutarla sin ser dependiente de los recursos externos. No se podrá ver un funcionamiento 100% real de la aplicación, pero se intentará emular parte del proceso.

Para la realización de la aplicación se han implementado 3 actividades, la primera nos permite escanear los dispositivos BLE a nuestro alrededor, la segunda actividad nos permite seleccionar la velocidad de refresco de la vista en la escena 3D y la tercera actividad nos muestra la reconstrucción de la escena 3D. En lugar de explicar las actividades implementadas, se explicará la implementación según los módulos usados. Para la primera actividad, se ha usado el módulo de Bluetooth, la segunda actividad no implementa ningún módulo, únicamente se usa como configuración inicial. La tercera actividad es la que implementa el módulo de vídeo, de mensajes y la creación 3D.

A continuación, pueden verse como han quedado las 3 actividades implementadas. En la imagen 20 puede verse la pantalla de selección de dispositivos BLE. Permite escanear los dispositivos, pero no conectarse a ellos. En la imagen 21 puede verse la actividad de propiedades. Una SeekBar permite seleccionar a queremos refrescar el movimiento de la imagen en la escena 3D y un checkbox nos permite realizar un cambio de colores en caso de tener dificultad para ver los colores de los objetos. Por último, la imagen 22 nos permite ver la escena 3D, con las paredes de los distintos despachos y los objetos situados en la escena.

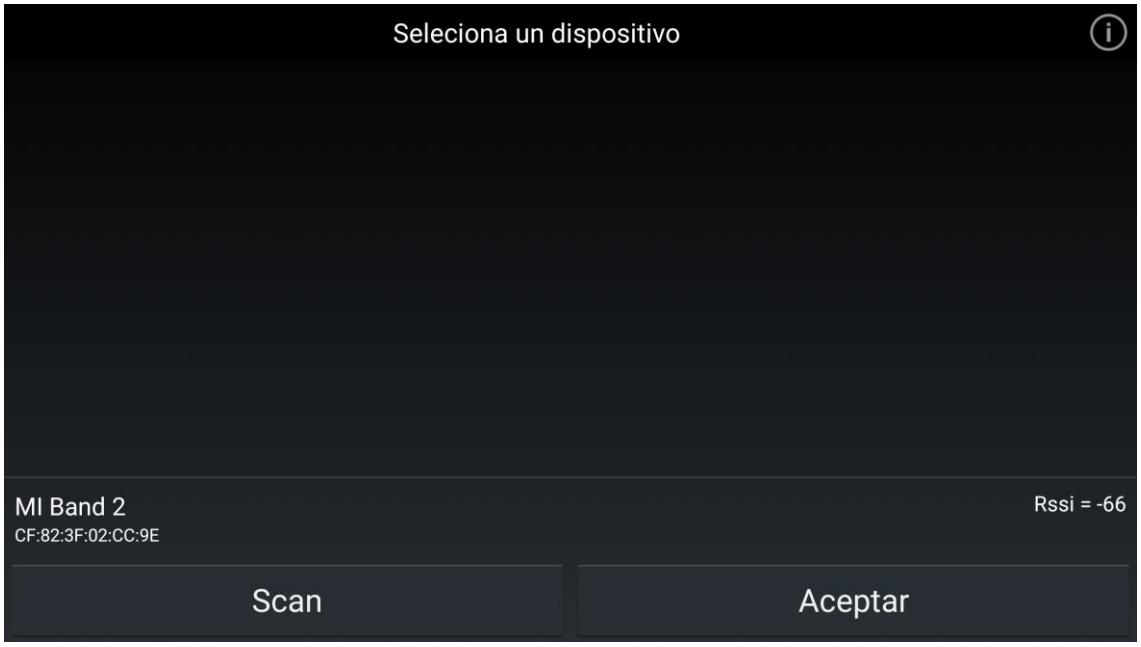

Ilustración 20 - Actividad 1 - BLE

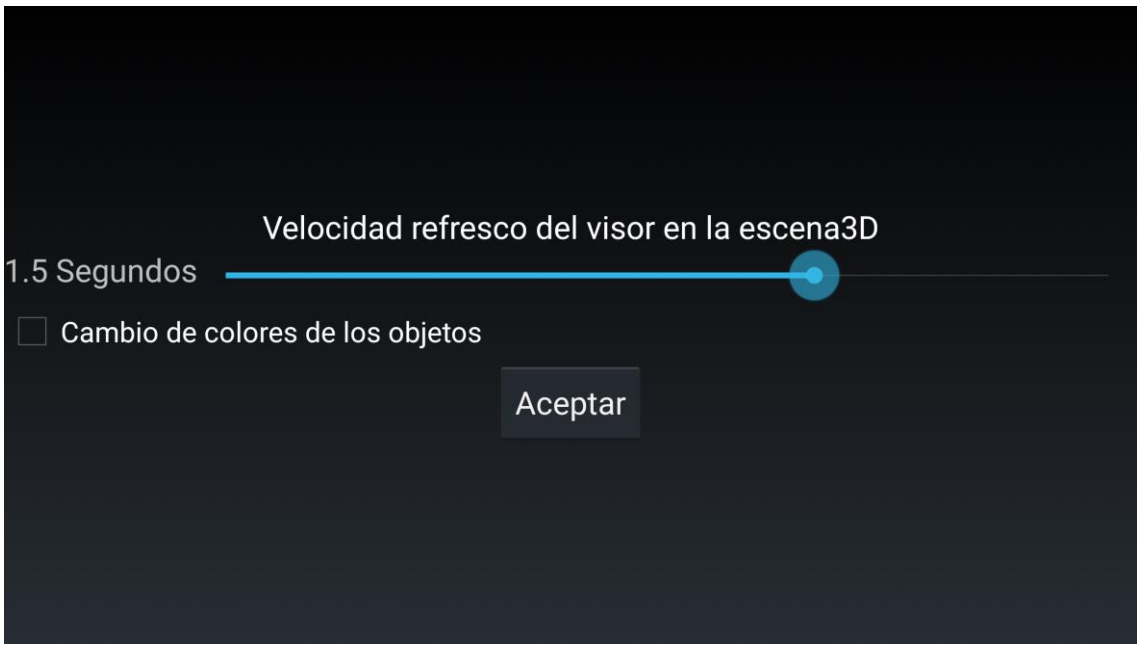

Ilustración 21 - Actividad 2 - Propiedades

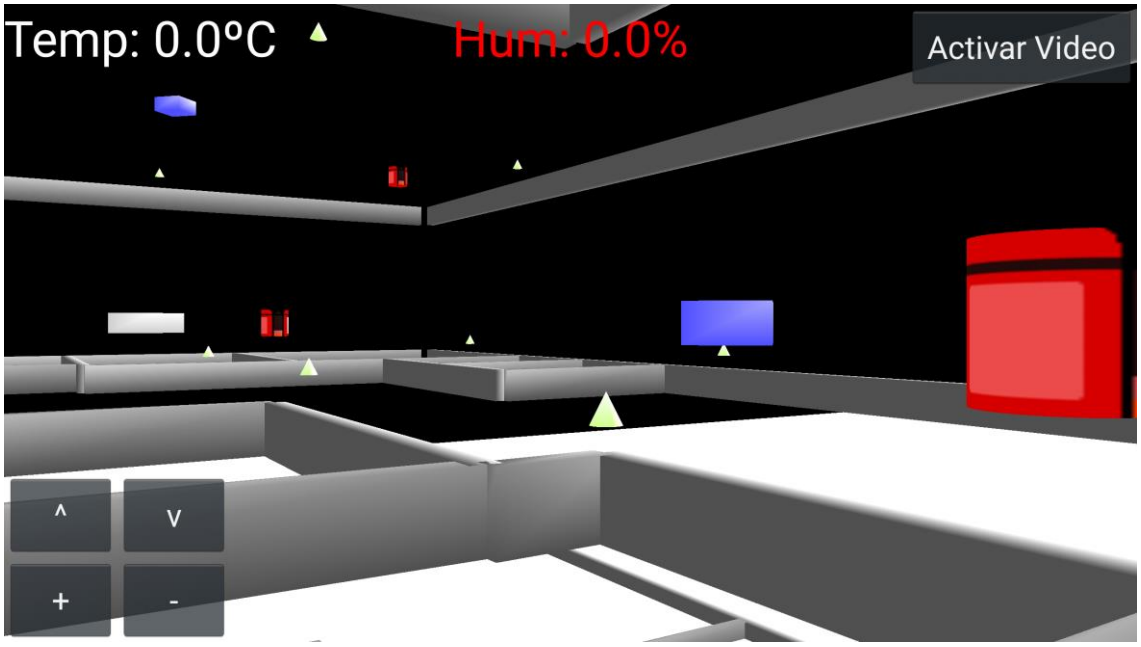

Ilustración 22 - Actividad 3 - Escena3D

4.1 Módulos

La aplicación se ha estructurado en 4 módulos independientes para poder realizar la implementación de cada uno de los módulos por separado y posteriormente integrarlo todo en una sola aplicación.

En el diagrama de clases de la figura 18 se pueden ver los distintos módulos de la aplicación:

- Bluetooth: este módulo es el encargado de realizar la comunicación con los dispositivos externos. Tanto la electrónica creada para este proyecto, con sensor de temperatura y humedad, como una pulsera con sensor de pulso cardiaco.

- 3D: este módulo es el encargado de generar toda la escena 3D de la aplicación.

- Bus mensajes: módulo encargado de escuchar el canal de comunicaciones con el servidor. Todos los mensajes son recibidos por este módulo y generan la acción correspondiente según el mensaje recibido.

- Video: servidor de video que mostrará la cámara del dispositivo en un PC.

Para comprobar el funcionamiento de cada módulo, primero se ha generado una aplicación exclusiva que realice únicamente la función deseada. Se ha testeado su funcionamiento y se ha preparado su integración con el resto de módulos. En total se han generado 4 pequeñas aplicaciones para poder tener la aplicación completa.

La figura 23 muestra el árbol del proyecto, mostrando las clases y recursos que lo componen

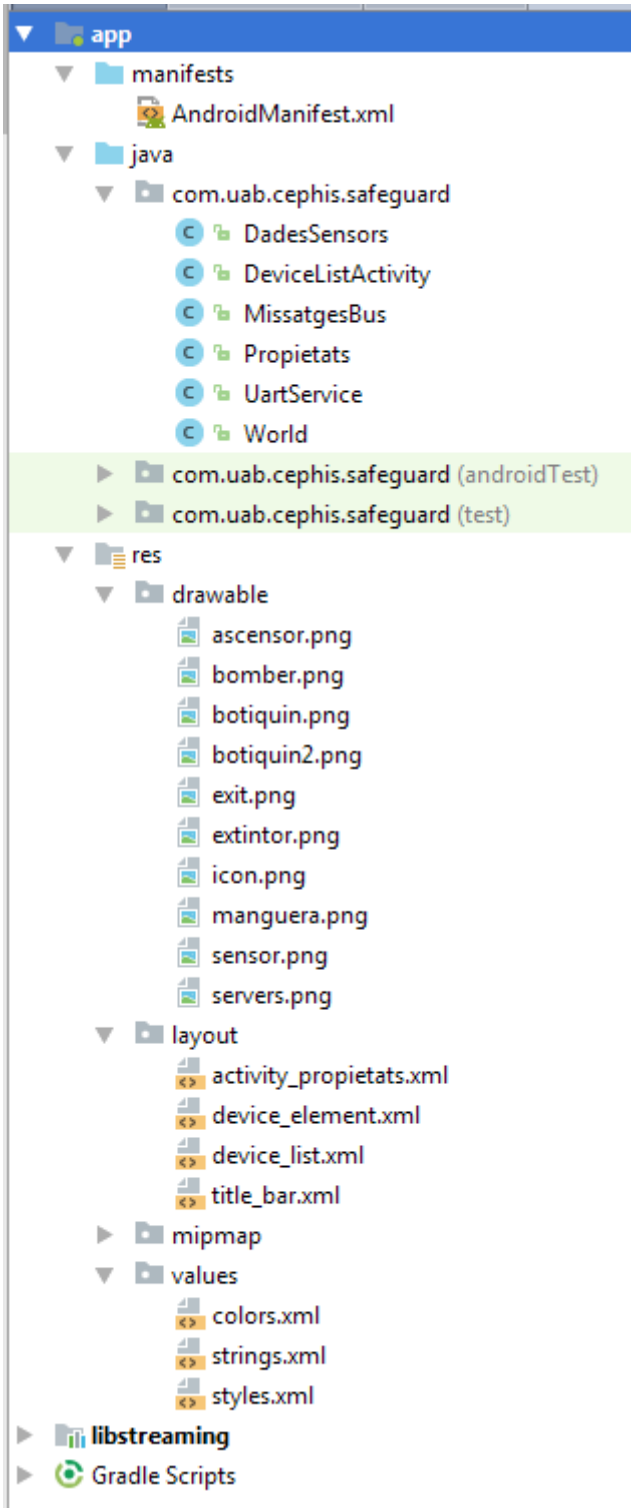

Ilustración 23 - Estructura del proyecto

4.1.1 Bluetooth

Esta parte de la aplicación controla la comunicación con los dispositivos Bluetooth Low Energy (BLE) conectados a nuestro dispositivo Android.

Se ha generado una aplicación que escanea los dispositivos BLE disponibles en nuestro alrededor, crear una comunicación con ellos y mediante un Android s*ervice* mantener la conexión y la comunicación

Este módulo sirve para conectarse a una electrónica desarrollada específicamente para este proyecto. Esta electrónica tiene:

- sensor de temperatura y humedad, estos datos se enviarán al dispositivo Android para poder mostrar los datos en la pantalla del usuario. Aparte se usarán para generar una alarma de choque térmico para evitar problemas de salud al usuario. [12]

- Sistema de comunicación Lora, este sistema permite comunicación a largas distancias de forma segura. Con este sistema podemos enviar un mensaje al servidor informando de los valores que estamos leyendo de los sensores.

- IMU de 9 ejes, sensor formado por un acelerómetro, giroscopio y magnetómetro. Este sensor nos permite detectar una caída de la persona que lleva la electrónica, pudiendo informar al sistema que se ha producido un evento de caída.

- Bluetooth, para poder realizar la comunicación entre el dispositivo Android y la electrónica, esta lleva un chip BLE.

Este módulo también nos permite conectarnos a una pulsera con sensor de pulso cardiaco. Esto significa que tendremos dos procesos de Bluetooth que controlar que nos enviarán datos de forma periódica. Con los datos del sensor de temperatura y humedad y los datos del sensor de pulso podremos calcular cuando se está llegando a la zona critica de choque térmico. Esta zona es peligrosa ya que puede generar pérdida del conocimiento y la muerte de la persona en caso de producirse en condiciones extremas. En la figura 24 puede observarse una tabla con los valores de sensación de temperatura dada una temperatura y humedad. En rojo puede apreciarse los puntos en los que es peligroso trabajar.

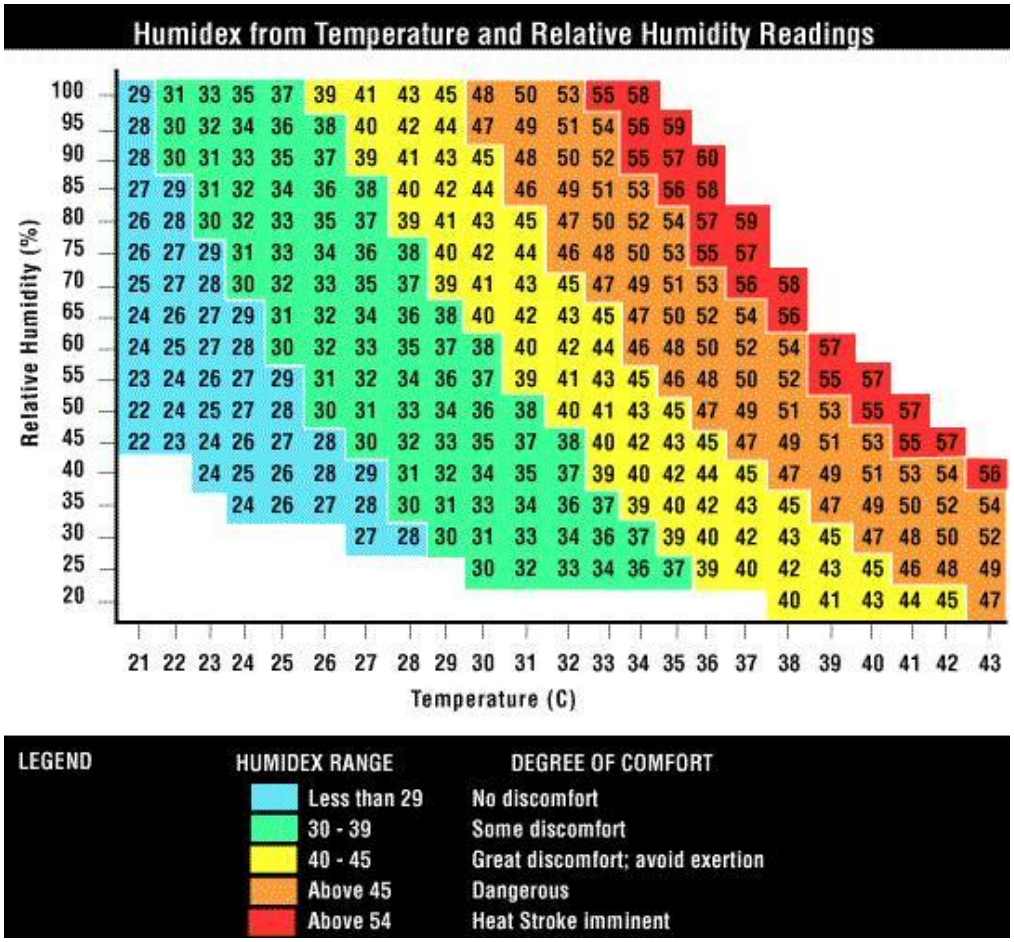

Ilustración 24 - Tabla choque térmico

La implementación del módulo Bluetooth está formada por la clase escáner y un servicio. La clase escáner nos buscará los dispositivos BLE que haya a nuestro alrededor y nos permitirá realizar una conexión con ellos. Una vez creada la conexión, el servicio se encargará de mantener la conexión entre los dispositivos y de realizar la comunicación.

Primero de todo hay que especificar en el Manifest.xml los permisos necesarios para usar el Bluetooth

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

A continuación, deberemos comprobar que temenos BLE en nuestro dispositivo, sino no podremos usar la aplicación. En caso de tener BLE, debermos comprobar que este activado o mostrar un aviso para realizar la activación.

Lo siguiente será realizar la búsqueda de dispositivos usando el método *startLeScan().* Este método nos devolverá todos los dispositivos localizados a nuestro alrededor. Para nuestra aplicación la búsqueda se restringe a los tipos de dispositivos que permitan realizar comunicación UART, usada para la comunicación entre la

electrónica y el sistema Android, o dispositivo con HRS (hearth rate sensor) que nos dará datos de un sensor de pulsaciones.

Una vez hemos realizado la comunicación con el dispositivo se lanza el servicio que realizará la comunicación BLE. Cuando un callback se active, lanzará un broadcastUppate() de la acción solicitada. EN nuestra aplicación un broadcastReceiver() recibirá el dato enviado por bluetooth.

4.1.2 Bus de mensajes

Este módulo realiza una comunicación con un servidor de mensajes. Todos los datos que se generan en los distintos procesos del proyecto se envían a este servidor de mensajes. Todo el que esté escuchando el canal podrá leer los mensajes que van llegando y activar las acciones necesarias en caso de ser un mensaje dirigido a su parte de aplicación. En la aplicación Android el sistema de mensajes se usa para recibir peticiones al dispositivo Android.

Por ese sistema de mensajes el usuario Android recibirá datos de caídas de sus compañeros, peticiones de activación (o desactivación) de la cámara del dispositivo, mensajes de alarma que se mostrarán en la pantalla del usuario, así como también enviará datos de su punto de vista para generar una visión 3D de lo que está viendo el usuario, y también enviará la IP de su dispositivo para poder realizar la conexión de video.

Los métodos principales de esta clase son:

- Suscribe: este método realiza la conexión con el servidor de mensajes y a medida que recibe mensajes los va analizando. Mediante este thread recibiremos los mensajes para activación del video.

- Main: este método es el usado para enviar un mensaje al servidor de mensajes. Es llamado desde el método Enviar()

- Enviar(type, argv): Construye la trama del mensaje a enviar. Dependiendo del type especificado tendrá más o menos argumentos. Type=0 se usa para enviar en qué dirección estamos mirando. Type=1 se usa para enviar la IP del servidor de video.

- GetTemps/GetHum: estos métodos retornan la temperatura o humedad recibidos por el bus de mensajes. Implementado en el supuesto que no tengamos conexión por Bluetooth para recibirlo directamente desde la electrónica.

- SetTemp/SetHum: métodos usados al recibir un mensaje con esta información. Los datos son almacenados en una variable y pueden ser recuperados posteriormente por GetTemp/GetHum

los metodos Enviar, Subscribe, GetTemp i GetHum los usamos en la clase World, donde tenemos la escena en 3D y la información para el usuario.

En la siguiente imagen, ilustración 25, puede verse las tramas enviadas por el canal de comunicaciones. En este caso recibimos la misma trama puesto que es el único mensaje que se está enviando

```
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:3,pitch:0,roll:274}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:3,pitch:0,roll:274}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:3,pitch:0,roll:274}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:3,pitch:0,roll:274}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:3,pitch:0,roll:274}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:3,pitch:0,roll:274}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:274}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:274}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:274}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:274}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:273}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:273}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:273}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:4,pitch:1,roll:273}
/com.uab.cephis.safeguard D/3dRabbit:  [x] Sent '{msgtype:camera_orientation,id:2525,building:EE110,acimut:353,pitch:357,roll:292}'
com.uab.cephis.safeguard D/3dRabbit: [recived] {msgtype:camera_orientation,id:2525,building:EE110,acimut:353,pitch:357,roll:292}
```

Ilustración 25 - Trama bus de mensaje

En los métodos suscribe y main donde se realiza la conexión se han dejado con valores dummy para no conectarse a los servidores, puesto que son los servidores de producción.

4.1.3 Vídeo

Para la implementación de este módulo se ha usado la librería libstreaming[13]. Esta librería desarrollada por el usuario Simon (fyhertz) y compartida en github usando la licencia Apache 2.0, permite la creación de un servidor o cliente de video RTSP, usando codificación H263 o H264.

Se ha optado por esta solución ya que permite de forma rápida tener un servidor de video en nuestro dispositivo Android y será el cliente de video el que se conecte a nuestro dispositivo para poder visualizar las imágenes capturadas por nuestro dispositivo.

La clase SessionBuilder es la usada para crear nuestro servidor de video. Este servidor será un servicio que estará corriendo cuando recibamos una petición de video mediante el servidor de mensajes. Los distintos métodos que implementa está clase permiten configurar la calidad del video y el códec usado para el video y el audio.

Uno de los métodos requeridos para poder usar la cámara de video es pasarle una SurfaceView dentro de nuestra aplicación para insertar la vista previa de la cámara. Esto limita la funcionalidad del servidor

ya que obliga a tener una imagen de la cámara mientras se realiza la retransmisión de video.

Para solucionar esto se procedió a modificar la librería para permitir la retrasmisión sin tener una vista de la cámara en pantalla. Para ello se creó un nuevo método que en lugar de usar un SurfaceView use un SurfaceTexture. De esta forma si en lugar de asignar la vista previa a una vista lo asignamos a un objeto OpenGL y no mostramos ese objeto en nuestra vista, podemos estar transmitiendo video sin tener que ver la imagen de nuestra cámara.

La imagen de video tiene un retardo de 2 segundos respecto a lo que ve el usuario del dispositivo Android. Este retardo es aceptable ya que el video se quiere usar para poder dar más información al centro de control de las condiciones que encuentra el usuario dentro del edificio.

A continuación, hay una explicación de la clase y los métodos añadidos a la librería para poder trabajar con una SurfaceTexture:

- SessionBuilder, esta clase nos permite configurar nuestro servicio de video. En esta clase se ha añadido el siguiente método.

  ```
  public SessionBuilder setSurfaceTexture (int valor ) {
          mSurfaceTexture = new SurfaceTexture(valor) ;
          return this;}
  ```

  En el método build() se ha añadido una comprobación para escoger entre el método inicial (SurfaceView) y el nuevo método añadido (SurfaceTexture)

  ```
  if(mSurfaceView!=null){
          video.setSurfaceView(mSurfaceView);
  }
  else{
          video.setSurfaceTexture(10);
  }
  ```

- Session: También se ha añadido el método SurfaceTexture en esta clase, aunque no ha sido usada. Esta clase permite realizar stream de video como cliente en lugar de como servidor.

En le Manifest.xml hay que añadir la línea siguiente en el apartado de *<aplication>* para indicar que tenemos un servicio en nuestra aplicación y permitir su ejecución.

```
<service android:name="net.majorkernelpanic.streaming.rtsp.RtspServer" />
```

4.1.4 3D

Este módulo genera todo el entorno 3D de la escena y es la actividad principal de la aplicación. En esta actividad es donde se generarán o se recibirán todos los datos del resto de módulos.

Lo primero que se realiza en esta actividad es generar un mundo OpenGl para poder dibujar en él. Como se ha comentado en el apartado 2.2, se ha usado la API JPCT para la creación del entorno 3D. Esta API nos simplifica bastante la creación de elementos OpenGL.

Una vez tenemos la glSurface creada, se procede a la creación del mundo y añadir el foco de luz que iluminará la escena.

Toda la escena se genera mediante los datos recibidos de un servidor. Estos datos se piden al inició de la actividad y son:

- Objetos Estáticos: son los objetos dentro de a escena que pueden ser útiles para los equipos de emergencia, como, por ejemplo, salidas de emergencia, extintores, mangueras, ascensores, botiquines.

- Objetos Pared: estos objetos son las paredes de todo el edificio, de esta forma generamos una escena con todas las paredes para que sea más sencillo situar los objetos.

- Objetos Agente: este tipo de objeto son los agentes que están dentro del edifico. Nos permite ver la posición de los compañeros que estén en otra parte de la planta o en otras plantas.

Primero se piden los objetos estáticos y se posicionan en el mundo virtual, a continuación, se realiza la petición de las paredes y se termia de generar la escena. Las paredes están configuradas con un tamaño de 0.5 metros de altura para no obstaculizar la visión tanto de los objetos que hay detrás como para facilitar poder ver la imagen real con los datos de la escena virtual.

Por último, cada 5 segundos se realiza una nueva petición de Objetos Agente, de esta forma tenemos la posición de cada agente actualizada cada 5 segundos. No se puede conseguir una actualización de la posición en tiempo real puesto que las electrónicas que informan de la posición de los agentes tienen ese tiempo mínimo de notificación.

La petición al servidor es asíncrona, eso permite que la aplicación se siga ejecutando y en paralelo cuando se recibe la respuesta del servidor en formato JSON se procede a su análisis.

Los datos recibidos del servidor nos indican el tipo de objeto que estamos analizando y su posición dentro del edificio. Con esos datos procedemos a añadirlos a la escena virtual.

En caso de ser datos de agentes, se comprueba si el identificador del agente ya está incorporado a nuestro mundo virtual y se modifica los datos de su posición. En caso de que el agente aún no esté en el mundo virtual, se añade un nuevo objeto con los datos recibidos.

Aparte de recibir datos de los agentes cada 5 segundos, también se realiza una lectura del servicio de mensajes para comprobar si hemos recibido alguna petición. En este caso puede ser una alarma de caída de un compañero, petición de activación o desactivación de video o mostrar un mensaje de aviso por pantalla. En caso de ser una petición de aviso se procede a mostrarlo por pantalla. Si es una petición de video, se crea el servidor de video y se notifica por mensaje al bus de mensajes de la IP y puerto en el que está escuchando el servidor.

Cada segundo se realiza una actualización del punto de vista del dispositivo, de esta forma posicionamos la cámara del mundo virtual según hacia donde esté mirando el usuario de las gafas. Para realizar este posicionamiento usamos los sensores del dispositivo Android y usamos el *Rotation Vector Sensor.* [14]

El *Rotation Vector Sensor* es un tipo de sensor que usa los datos de la IMU del dispositivo Android para generar un posicionamiento dentro de una esfera. En nuestro caso la esfera es el planeta tierra y los datos que recibiremos del sensor en la coordenada X es la posición del norte magnético. Esto nos permite poder orientar el punto de vista respecto al norte terrestre. Si el usuario está mirando a 0º tendremos el Norte, en 90ª el Este, en 180ª el Sur i en 270ª el Oeste.

Estos datos de posición de punto de vista se mandan cada según mediante el sistema de mensajes al servidor de datos para poder generar una escena en 3D en el centro de control.

Se ha añadido a la pantalla de la actividad la información de temperatura, humedad y estado del vídeo. Cuando se recibe un mensaje para activar/desactivar el vídeo o cuando se recibe nuevos valores de temperatura y humedad se actualizan en la vista. Para poder hacer esto se ha tenido que hacer un thread para la parte de interface de usuario, que permite actualizar de forma separada los datos de OpenGL con los TextView que se usan para mostrar los datos de temperatura, humedad y vídeo.

En el caso de seleccionar la opción de cambio de colores en la actividad 2, ilustración 21, se han realizado modificaciones en la construcción de los objetos de la escena 3D, como puede verse en las siguientes imágenes.

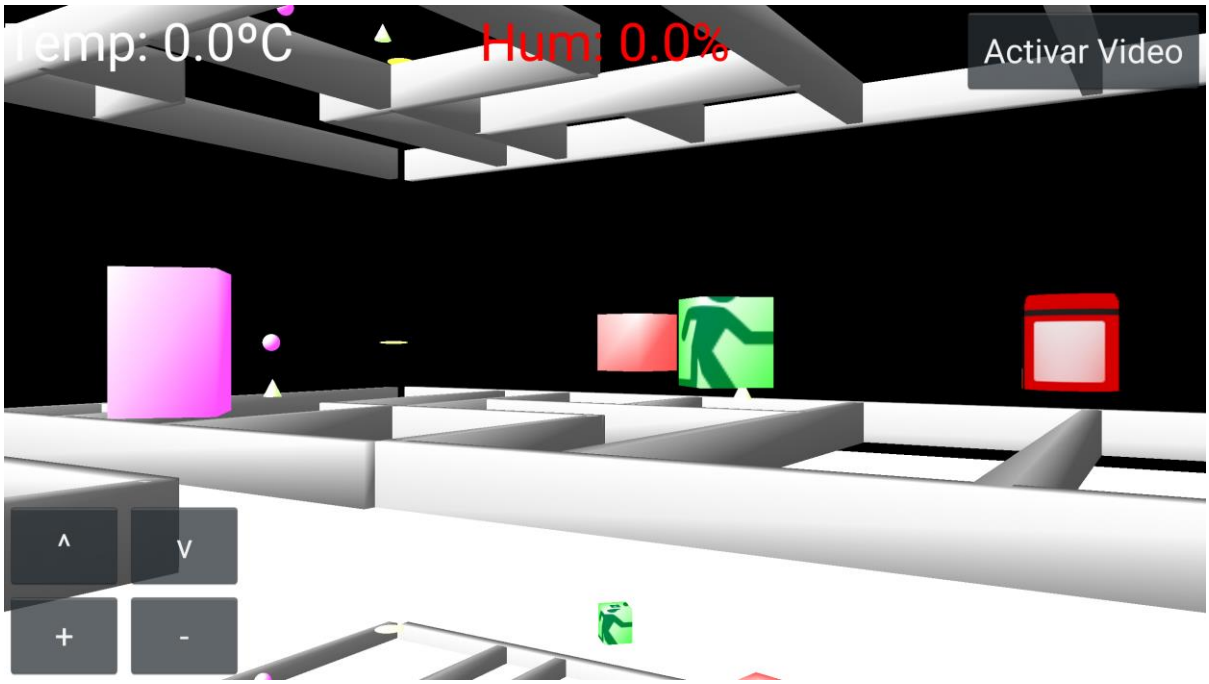

Ilustración 26 - Imagen colores original

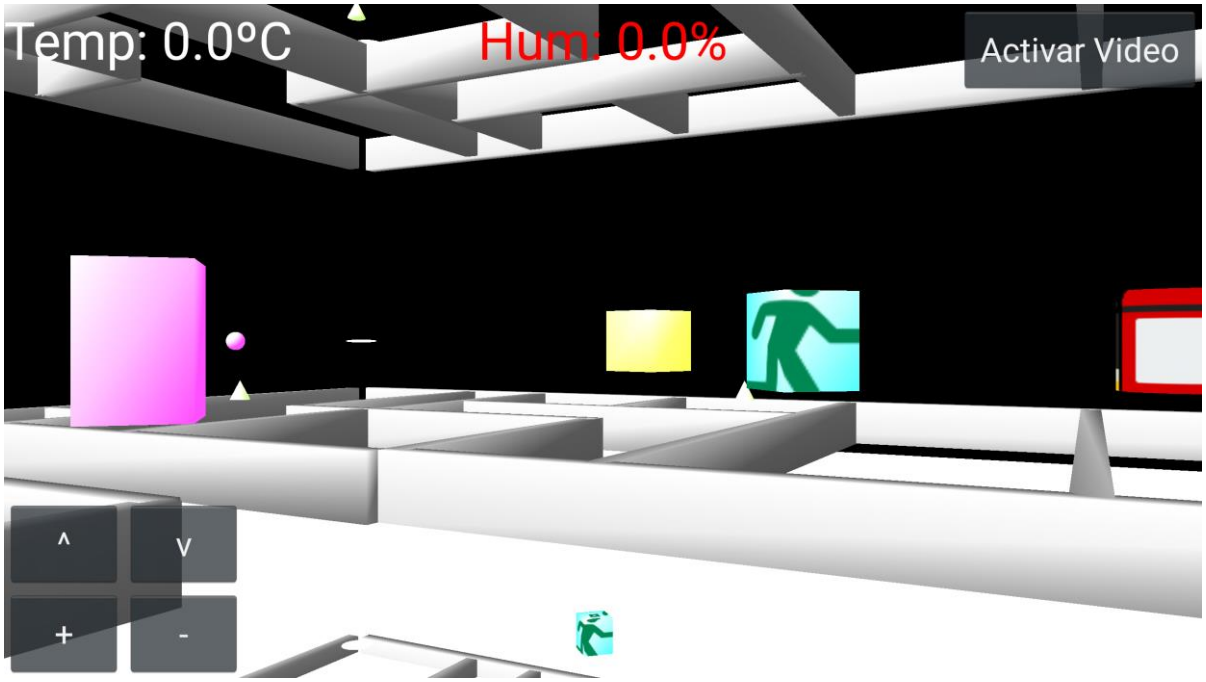

Ilustración 27 - Imagen colores modificados

Si nos fijamos en la imagen 26, puede verse en el centro de la imagen un objeto de color rojo y otro de color verde. En la imagen 27 se ha modificado el objeto de color rojo y se le ha asignado el color amarillo, y el objeto verde se ha modificado ligeramente para darle una tonalidad de verde algo distinta. Al no disponer de alguien con esta alteración en la percepción de los colores no se ha podido evaluar si las tonalidades escogidas con las correctas.

Procesos de la clase World

onCreate

En este método se inicializa la actividad, comprobando inicialmente si tenemos conexión Wifi. En caso de tenerlo, se procede a recuperar los valores pasados por la actividad anterior, velocidad de refresco y colores.

Se inicializa el bus de mensajes y se crean las texturas para cada uno de los objetos de la escena. (En la implementación solo se han insertado texturas para 3 elementos distintos).

Asignamos la aplicación a una OepnGLSurfaceView. Esta vista será la que usaremos para generar todo el contenido OpenGL de la escena3D.

A continuación, se crean los elementos de la interface de usuario, como son los textos y los botones. Estos elementos se crean mediante código y se insertan a la vista usando el método *addContentView().*

Una vez generados los botones, se inicializa la lectura de los sensores y nos conectamos al servidor de mensajes.

onPause, onResume, onDestroy, onStop

Estos métodos llaman a las implementaciones en cada una de las clases para detener o reiniciar los servicios. Se llama a los métodos de las clases MissatgesBus y DadesSensors para desconectarse del servidor de mensajes y detener la captura de valores de los sensores del dispositivo. También se realiza una detención del servicio de Bluetooth y del servidor de vídeo.

onBackPressed

Se ha realizado un override de este método para evitar que vuelva a la actividad anterior. Cuando se pulsa aparece un mensaje informando si queremos detener la aplicación.

Dentro de la clase World, tenemos 2 clases donde se realizan la mayor parte de los procesos

MyRenderer

En esta clase se implementan los métodos para dibujar en la GLSurfaceView. Inicialmente se crea la escena vacía con un suelo y un punto de luz, a continuación, se posiciona la cámara.

A partir de este momento cada vez que se dibuja un frame se llama al método onDrawFrame. En este método es donde dibujamos los elementos que hay en el framebuffer. El valor de refresco se usa en este punto para realizar la rotación de la cámara. Si movemos nuestro dispositivo en el eje X realizaremos una rotación de la cámara en ese eje.

Aparte, cada 5 segundos se realiza una consulta con el servidor para pedir la nueva posición de los agentes, y dibujarlos en la escena. Se usa este mismo proceso para realizar el dibujo de las paredes de la escena, es por ello los 5 primeros segundos de ejecución de la escena se usan para generar el contenido de objetos estáticos y paredes. Es en este mismo intervalo que se realiza la comprobación del estado del vídeo. Si se ha pulsado el botón para activar vídeo puede tardar hasta 5 segundos en activarse el servidor. Sabremos que se ha activado porque veremos que le texto de “Activar Video” cambia a “Desactivar Video”.

Para poder realizar cambios a nivel de interface de usuario, se crea un UIthread, lo que nos permite modificar el texto de los elementos como son los TextView y los Botones.

ObtenerJsonData

Esta clase es la encargada de realizar la consulta al servidor y recibir y analizar el Json de respuesta. Esta tarea se realiza de forma asíncrona para permitir la correcta ejecución del sistema sin bloquearlo. Una vez realizada la petición de cada uno de los elementos al servidor, se recibe la respuesta en el método onPostExecute, donde se analiza si el Json recibido es correcto.

A continuación, se comprueba en qué estado de petición estamos. Podemos estar en la primera petición, donde recibimos y creamos los objetos estáticos de la escena como son señales de salida de emergencia, botiquines, mangueras, extintores, etc. A cada uno de estos elementos se les asigna una figura geométrica para su representación en la escena. Aparte se ha añadido textura a alguna de las figuras para intentar ubicarlas de forma más rápida.

En la ilustración 28 puede observarse la petición realizada y la respuesta Json del servidor. Puede verse que cada elemento nos indica de que tipo es y su posición en latitud y longitud dentro de la escena.

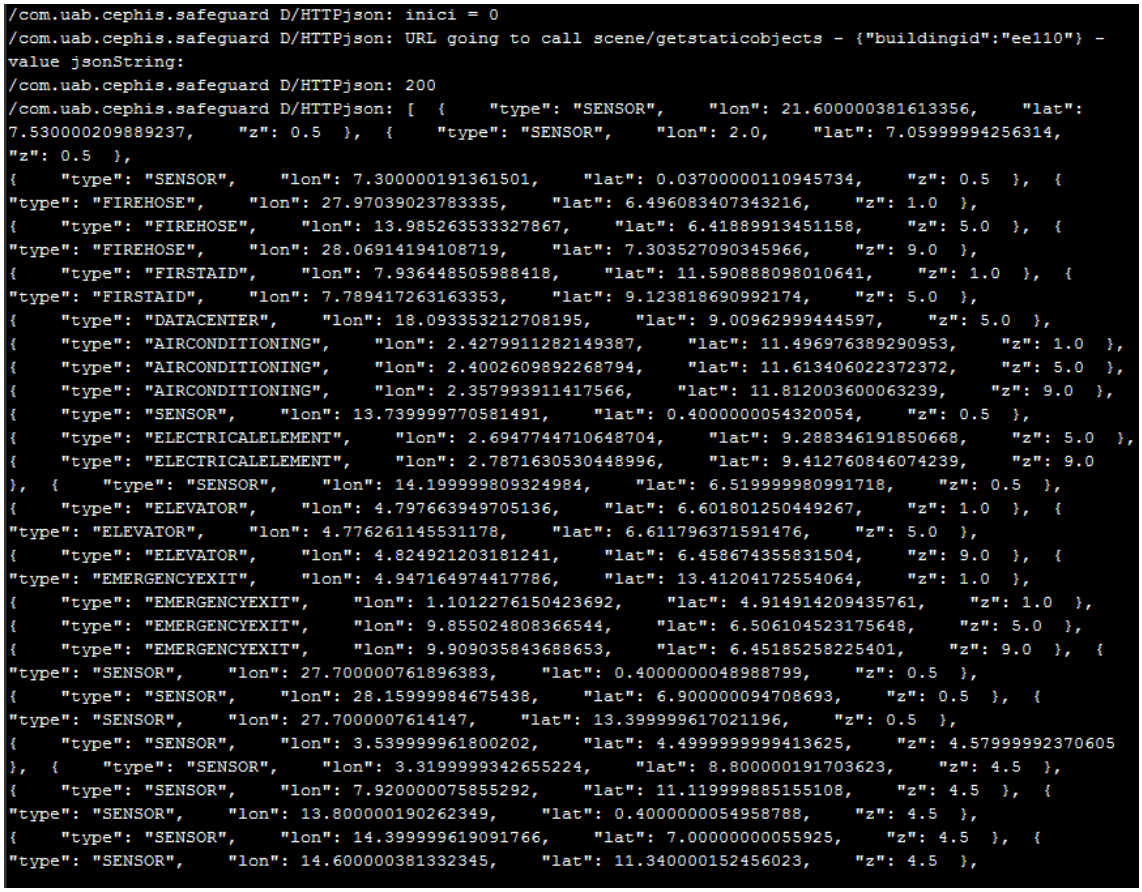

Ilustración 28 - Json recibido en la primera petición

Si estamos en la segunda petición Http, recibiremos un Json con los valores de las paredes. Se crea la estructura de cada habitación según sus puntos de coordenadas. En la figura 29 se muestra la estructura del Json recibido. Cada dos conjuntos de puntos forman una pared.

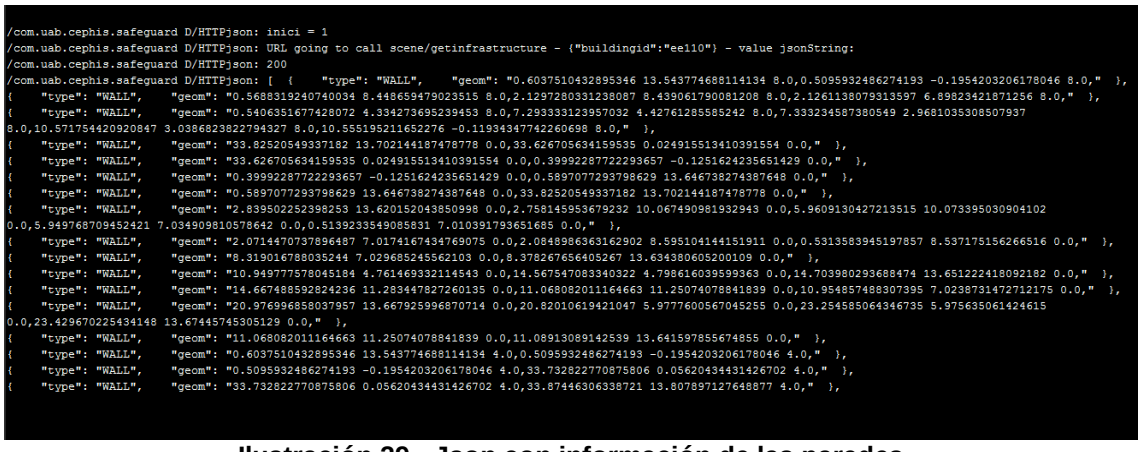

Ilustración 29 - Json con información de las paredes

Para la tercera petición estamos en el estado de recibir la posición de los agentes. A partir de este momento, esta petición se realizará cada 5 segundos para ir actualizando su posición.

En la imagen siguiente puede verse la estructura de Json recibida para la tercera petición. Para cada agente recibimos su identificador y su posición dentro de la escena.

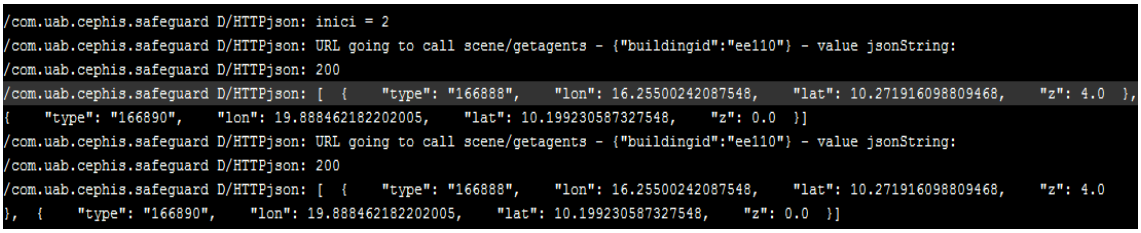

Ilustración 30 - Json con información de agentes

4.2 Test realizados

En el proceso de creación de la aplicación se han realizado distintos test para cada una de las partes que se han implementado y posteriormente se han unido en una sola aplicación.

4.2.1 Bluetooth

Como se ha comentado ene l punto 4.1 se ha realizado una aplicación para cada uno de los módulos implementados, el objetivo de estas aplicaciones era poder probar el funcionamiento de cada una de las partes del sistema para poder posteriormente integrarlo todo en una única aplicación.

Para la parte de bluetooth se ha realizado una aplicación que escaneaba los dispositivos Bluetooth de las cercanías y realizaba una conexión a la electrónica. Una vez establecida la comunicación teníamos que recibir datos de los sensores mediante comunicación UART. Si la trama de datos era la esperada quería decir que la comunicación era exitosa y por tanto podíamos implementar la parte de Bluetooth en la aplicación principal.

4.2.2 Bus de mensajes

Aplicación destinada a probar el funcionamiento del sistema de mensajes, permitir una conexión con el servidor de mensajes, poder leer los mensajes y poder realizar una publicación de mensaje.

El sistema de mensajes puede recibir gran cantidad de mensajes y todos con una estructura parecida, se han implementado filtros para solo aceptar los mensajes con la estructura deseada por nuestra aplicación, de esta forma descartamos los mensajes que no están destinados a ser interpretados por nuestra parte del sistema.

Una vez testeado el sistema de mensajes se ha procedido a la implementación en la aplicación principal.

4.2.3 Vídeo

Realizar una aplicación integrando la librería LibStreaming y crear un servidor de vídeo. Se ha realizado la visualización del vídeo mediante la aplicación para ordenador VLC.

Inicialmente se pretendía crear una aplicación web que permitiera la visualización del vídeo de forma independiente del sistema, pero no ha sido posible. El protocolo RTSP no está soportado de forma nativa en web y por tanto no es de fácil implementación. Se han buscado soluciones como la creación de un servidor NodeJS que realice una

conversión de RTSP a otro formato de video que sea fácilmente reproducible en web, pero la combinación de NodeJS, el paquete ***node-rtsp-stream*** [15] y el reproductor de video **jsmepg** [16] no ha permitido la conversión del video RTSP. Este paquete hace uso del programa ffmpeg para realizar la conversión de RTSP al formato deseado, pero por lo que se ha visto en el proceso de pruebas, el programa ffmpeg no consigue mantener el canal de comunicación con el servidor RTSP cuando se pide la conversión y provoca que no se realice ninguna emisión de video.

Se han realizado pruebas con otros sistemas web para poder reproducir video RTSP en web, como, por ejemplo, usando una extensión de Chrome que permite la reproducción de RTSP. Este sistema también ha sido descartado ya que no se ha conseguido su funciona miento.

Por último, se ha estudiado la realización del servidor de video usando otro protocolo que no sea RTSP. Por falta de tiempo se ha descartado la opción y en un futuro se volverá a estudiar la opción de intentar usar websocket directamente desde el dispositivo Android.

4.2.4 Escena 3D

Está aplicación es al usada para integrar todos los módulos testeados, por lo tanto, es la que hará de aplicación principal una vez esté todo integrado.

Se han realizado distintas pruebas en esta aplicación.

1. Se ha realizado la medición de los valores de sensores de al IMU del dispositivo Android.
2. Pruebas de comunicación mediante HTTP, para poder descargar los datos del servidor se realiza una petición HTTP
3. Obtención de los datos JSON y posterior análisis.
4. Creación de los elementos en la escena según los datos obtenidos del JSON
5. Actualización de datos en la escena según los datos de JSON
6. Implementación del sistema de mensajes, y envío de datos del punto de vista de la escena virtual al servidor.
7. Obtención de datos del dispositivo Bluetooth y actualización en la vista de la actividad
8. Creación del servidor de vídeo una vez integrado. Para emular la recepción de mensajes se ha sustituido el TextView con el estado del servidor de vídeo por un botón para poder activar o desactivar el vídeo a requerimiento del usuario.

4.3 Emulación de funcionamiento

Como se ha comentado, la versión entregada algunas de las funcionalidades están desactivadas.

1. Se realiza la búsqueda de dispositivos Bluetooth, pero no permite realizar una conexión y la aplicación permite pasar a la siguiente actividad. En la figura 30 puede verse la actividad que escanea los dispositivos Bluetooth Low energy. Los ítems de lista son seleccionables y nos aparecerá un Toast en lugar de realizarse la conexión. Usando el botón Aceptar pasaremos a la siguiente actividad.

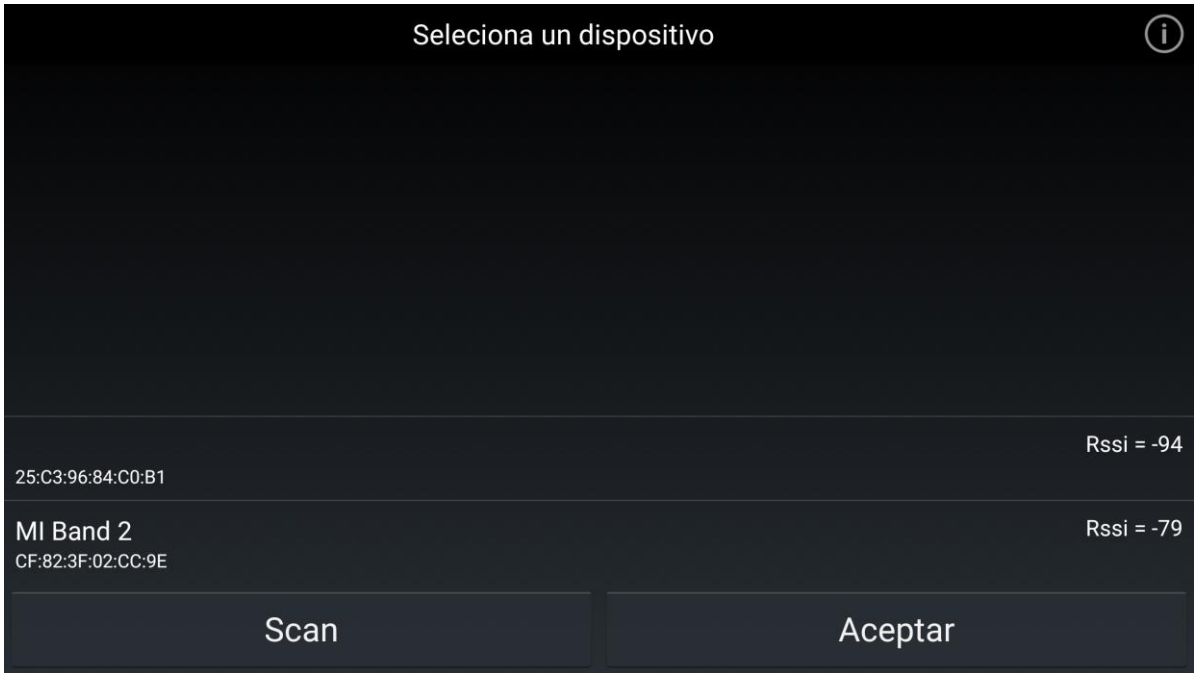

Ilustración 31 - Actividad BLE mostrando dos dispositivos

2. El sistema de mensajes está desactivado, se ha dejado la estructura intacta pero no se realiza la conexión con el servidor de mensajes, por el contrario se muestra en el LogCat un mensaje de log usando *Log.d(TAG,msg),* en este mensaje se muestra la información que se enviaría al servidor de mensajes. En la figura 25 puede verse la trama de mensajes enviados y recibidos.

3. Como tampoco se puede recibir ningún mensaje, se ha substituido el texto del estado de video de la interfaz por un botón para permitir activar o desactivar el vídeo. Si se está conectado a un wifi con el móvil y se tiene un Pc en la misma red, cuando se apriete el botón nos mostrará un mensaje Toast por pantalla con la IP:Puerto.

   Usando el reproductor VLC se puede ver el video que transmite nuestro dispositivo. Como puede verse en la siguiente imagen, aparece un mensaje en pantalla informando de la IP y el puerto al que debemos conectarnos. El botón de la parte superior derecha nos

permite activar y desactivar el vídeo, cambiando su texto para indicarnos si el vídeo se está emitiendo o no.

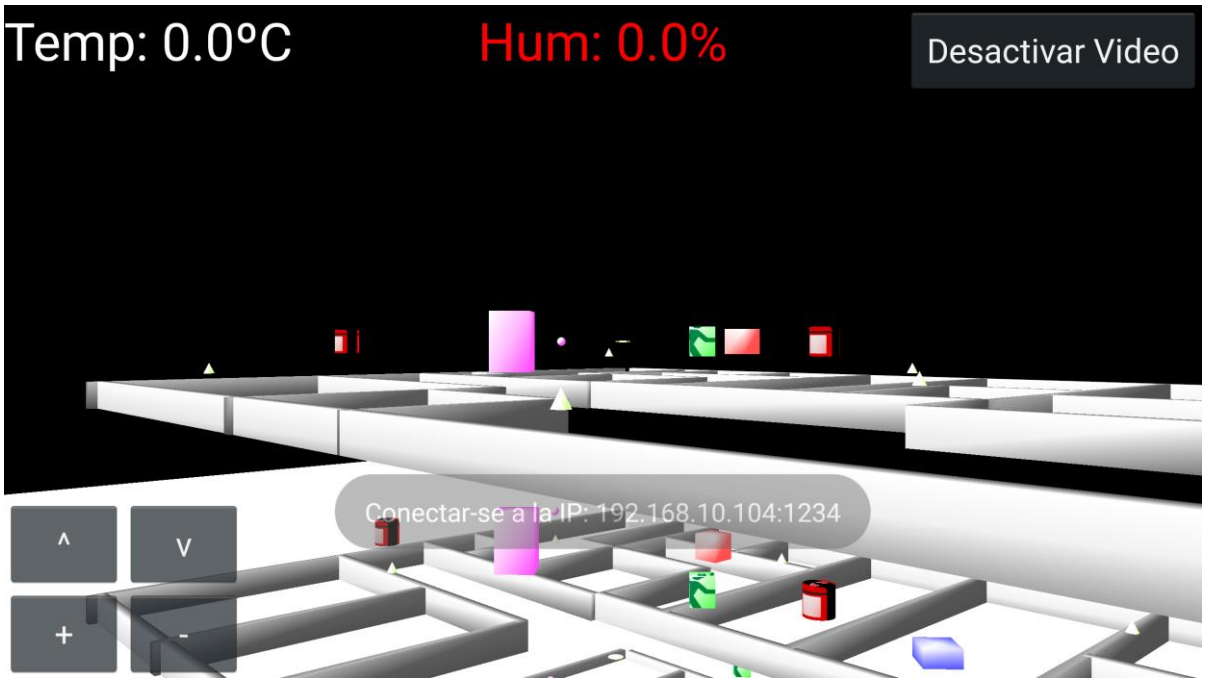

Ilustración 32 - Escena3D con navegación y botón Vídeo

4. Como es posible que no se reciban actualizaciones de los agentes, se ha añadido 4 botones para poder moverse por la escena, permitiendo ir hacia adelante o hacia atrás y ascender o descender un piso. Como se puede apreciar en la figura anterior, en la parte inferior izquierda se pueden ver los 4 botones. Los que tiene los símbolos “^” y “v” nos permiten movernos hacia adelante y hacia atrás en la escena. Los que tienen el símbolo “-” y “+“ permiten subir o bajar de piso, informando por pantalla cuando se alcanza el piso máximo y el mínimo.

5. Conclusiones

Una vez realizada la implementación del trabajo solo queda comentar y valorar el trabajo realizado. Primero de todo, dar las gracias a los profesores del master por el seguimiento que han realizado de las actividades planteadas a lo largo de las 9 asignaturas cursadas y de las indicaciones para solucionar los problemas que pudieran ir apareciendo. En este trabajo, al ser específicamente una aplicación para Android, he usado como base los conocimientos adquiridos en las asignaturas de *Desarrollo de aplicaciones para dispositivos Android*, Desarrollo *avanzado de aplicaciones para dispositivos Android* y *Diseño de productos interactivos multidispositivo.*

En este trabajo aparte de trabajar con el entorno de desarrollo Android Studio para realizar la programación de la aplicación, he tenido que crear un mundo OpenGL usando un motor gráfico y realizar modificaciones a una librería externa al sistema Android para poder retransmitir video de forma sencilla.

De los objetivos planteados inicialmente se han implementado en su totalidad a excepción de la creación de un sistema web para visualizar el vídeo retransmitido por el móvil. Esto ha sido ocasionado por el protocolo usado para realizar la transmisión, que no es compatible con los formatos web y tampoco ha sido posible conseguir una conversión del video como ya se ha explicado en el apartado 4.2.3.

Respecto a los objetivos opcionales, ha sido imposible realizarlos puesto que, como se ha comentado durante la memoria, este proyecto forma parte de un proyecto más extenso y finalmente esos apartados no están previstos que se implementen hasta el primer cuarto de 2018.

Se ha seguido la planificación propuesta al inicio del proyecto y se han cumplido con todas las fechas de entrega. En cuanto a la forma de realizar la implementación, se ha intentado implementar de forma modular todos los aspectos de la aplicación y testeándola de forma independiente para poder cumplir con los objetivos propuesto. El único cambio que se ha realizado respecto a la propuesta inicial ha sido la visualización del vídeo, que se realiza mediante un reproductor de PC en lugar de un visor web.

Como trabajo futuro se trabajará en encontrar un protocolo de vídeo que permita su reproducción web y conseguir un sistema no dependiente de aplicaciones instaladas por el usuario. También se trabajará en realizar la implementación de los dos objetivos opcionales no cumplidos.

6. Glosario

AMQP *Advanced Message Queuing Protocol* (Protocolo de Cola de Mensajes Avanzada)

API *Application Programming* Interface (interfaz de programación de aplicaciones)

AR *Augmented Reality* (Realidad Aumentada)

BLE *Bluetooth Low Energy* (Bluetooth de Baja Energía)

IDE *Integrated Development Environment* (Entorno de Desarrollo Integrado)

IMU *Inertial Measurement Unit* (Unidad de Medición Inercial)

HRS *Heart Rate Sensor* (sensor de pulso cardíaco)

UART *Universal Asynchronous Receiver-Transmitter*

7. Bibliografía

[1] 3dsafeguard.uab.cat – Web del proyecto con información de las empresas colaboradoras - 3 de octubre de 2017.

[2] www.opengl.org – Web de las librerías gráficas OpenGL – 8 de octubre de 2017

[3] https://www.metavision.com/ - Web de las gafas Meta2 – 20 de octubre de 2017

[4] https://www.epson.es/products/see-through-mobile-viewer/moverio-bt-200 - Página del fabricante Epson – 20 de octubre de 2017

[5] https://www.osterhoutgroup.com/r-7-glasses-system.html - Página del fabricante de las gafas ODG R-7 – 20 de octubre de 2017

[6] https://www.microsoft.com/en-us/hololens - Página de producto de Microsoft Holo Lens – 20 de octubre de 2017

[7] https://www.epson.es/products/see-through-mobile-viewer/gafas-moverio-bt-300 - Web del producto Epson BT-300 – 20 de octubre de 2017

[8] https://unity3d.com/es - Web del motor gráfico Unity3D – 21 de octubre de 2017

[9] http://www.ogre3d.org/ - Web del motor gráfico Ogre3D – 21 de octubre de 2017

[10] http://www.jpct.net/jpct-ae/index.html - Web del motor gráfico específico para Android – 21 de octubre de 2017

[11] http://www.rabbitmq.com/ - Web del sistema de mensajes implementado – 23 de octubre de 2017

[12] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3076691/ - Effect of Thermal Stress on Cardiac Function, Thad E. Wilson and Craig G. Crandall – 15 de noviembre de 1027

[13] https://github.com/fyhertz/libstreaming - Github de la librería – 15 de noviembre de 2017

[14] https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-rotate – web Android developers acerca del vector de rotación – 3 de noviembre de 2017

[15] https://www.npmjs.com/package/node-rtsp-stream - paquete conversión RTSP - 20 noviembre de 2017

[16] https://github.com/phoboslab/jsmpeg - Web del videoplayer – 21 de noviembre de 2017

8. Anexos

8.1 Información sobre la compilación

La compilación del proyecto no requiere de ningún requisito previo. El propio Android Studio informará de los paquetes del SDK de Android que faltan por instalar.

- Se ha comprobado la compilación en 1 máquina distinta a la del desarrollo.

- Se ha requerido instalar Android-25, Android-21, Build-tools-26.0.2 y Constraints-layout-2.0.2. El propio IDE Android Studio ha realizado la instalación de cada uno de los paquetes.

- Se ha ejecutado la compilación en 3 dispositivos distintos con Android 5.1, 6.0 y 7.0 para comprobar el correcto funcionamiento

8.2 Información de uso

- Actividad 1:

Permite escanear los dispositivos Bluetooth >4.0. No permite realizar conexión con ningún dispositivo. Pulsando Aceptar pasaremos a la segunda actividad

- Actividad 2:

Esta actividad permite seleccionar la velocidad de refresco en la escena 3D y modificar colores. Pulsando Aceptar pasamos la tercera actividad

- Actividad 3:

Esta actividad crea una escena 3D, si se dispone de unas gafas, permitirá ver a través de la escena, puesto que el fondo es negro.
Para navegar por la escena pulsar los botones de la parte inferior izquierda para avanzar, retroceder o subir/bajar piso.
Se puede navegar por la escena 360º girando el dispositivo Android en el eje X o en caso de tener unas gafas, moviendo la cabeza y girando sobre uno mismo.
Si se pulsa el botón “Activar vídeo” un mensaje mostrará la IP y puerto para poder conectarse mediante VLC. En el reproductor VLC, seleccionar Medio->Ubicación de red e introducir: **rtsp://IP:puerto**