CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Generación de Indicaciones para Localización de Objetos en Realidad Aumentada

Autor: Pablo Aceituno Ferro

Directora: Angélica de Antonio Jiménez

MADRID, JUNIO 2017

ÍNDICE

RESUMEN

"La realidad aumentada abarca más que la realidad virtual, probablemente con diferencia, porque nos da la posibilidad de estar presentes y de comunicarnos, pero también de que disfrutemos a otros niveles visuales", Tim Cook, director ejecutivo de Apple, en una entrevista a la cadena estadounidense ABC News.

El comentario que acabamos de leer resume perfectamente el propósito de este proyecto. Disponemos de un sistema diseñado para Realidad Virtual en el que se pretende dar indicaciones en lenguaje natural para ayudar a los usuarios a localizar objetos dentro del entorno. Para ello, las indicaciones se pueden apoyar en otros objetos que sirven como referencia (“el cenicero está sobre la mesa”, o “las llaves están junto al jarrón amarillo” serían ejemplos). A la hora de seleccionar los mejores objetos a utilizar como referencia, el sistema es capaz de calcular la “saliencia” de un objeto, esto es, el conjunto de cualidades de cada objeto en particular que hace que un ser humano se fije en él antes que en cualquier otro, como pueden ser el color, la forma, etc. La principal idea de este trabajo de fin de grado es dar los primeros pasos para llevar dicho sistema al ámbito de la realidad aumentada, elevando el grado de utilidad de la herramienta al sumergirnos en escenas del mundo real.

Durante el desarrollo del mismo, se han adquirido conocimientos acerca del funcionamiento y características de tecnologías de realidad aumentada que ofrece el mercado, modelos para obtener con precisión la posición de un usuario en un entorno real, incluyendo interiores. También se han obtenido conocimientos en lo que se refiere al desarrollo de aplicaciones para Android usando Unity 3D con el lenguaje de programación C#, incluyendo el manejo de tecnologías GPS y diseño de modelos Cliente-Servidor.

Esta memoria está diseñada también para que quienes recojan el testigo en nuevos proyectos que se lleven a cabo en el grupo Madrid HCI Lab, encuentren en ella una guía de referencia y preparación para continuar o abordar otros proyectos a partir del mismo.

El trabajo se ha llevado a cabo contando con los recursos del Laboratorio Decoroso Crespo, perteneciente al Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software de la Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid.

ABSTRACT

"My own view is that augmented reality is the larger of the two (AR and VR), probably by far, because this gives the capability for both of us to sit and be very present talking to each other, but also have other things visually for both of us to see," Tim Cook, Apple CEO.

The comment we have just read perfectly summarizes the main purpose of this project. We have a system designed for Virtual Reality in which we seek to calculate the "salience" of an object, that is, the set of characteristics that make humans to pay attention in it before in any other, such as color, shape, etc. The main idea of this project is to take the first steps to bring the system into the realm of augmented reality, raising the degree of reliability of the tool by immersing us in scenes with a deeper immersion.

During the development of the same, I have been acquired knowledge about the operation and characteristics of augmented reality technologies offered by the current market, models to accurately obtain the position of a user in a real environment, including interiors. Knowledge has also been gained in the development of Android applications using Unity 3D with the C # programming language, including GPS technology management and Client-Server model design.

This report is also designed for those who collect the witness of the projects that are carried out in the Madrid HCI Lab group, find in it a reference guide and preparation to continue or approach other projects from the same.

The work has been carried out with the resources of the laboratory Decoroso Crespo, belonging to the Department of Languages and Information Systems and Software Engineering of the Faculty of Technical Higher Computer Engineering, Universidad Politécnica de Madrid.

1.- INTRODUCCIÓN

El punto de partida de este proyecto es un sistema construido principalmente en Unity 3D, con el objetivo de dar indicaciones en lenguaje natural para ayudar a los usuarios a localizar objetos dentro de entornos virtuales. Para ello, las indicaciones se pueden apoyar en otros objetos que sirven como referencia ("el cenicero está sobre la mesa", o "las llaves están junto al jarrón amarillo" serían ejemplos). A la hora de seleccionar los mejores objetos a utilizar como referencia, el sistema es capaz de para calcular la saliencia de los objetos. Este sistema ha sido desarrollado por diferentes alumnos en el Laboratorio Decoroso Crespo como apoyo para la tesis doctoral de Graciela Lara López bajo la dirección de Angélica de Antonio Jiménez. En él encontramos una serie de escenarios virtuales que diseñé con la herramienta Blender en la asignatura Prácticum, además de un conjunto de funcionalidades implementadas por antiguos alumnos en sus respectivos trabajos de fin de grado como la voxelización de los objetos, un editor-exportador que calcula aplicando unas métricas la saliencia individual de los objetos, así como un sistema generador de indicaciones, incluyendo indicaciones por voz, que ayudan al usuario a identificar y localizar el objeto a buscar.

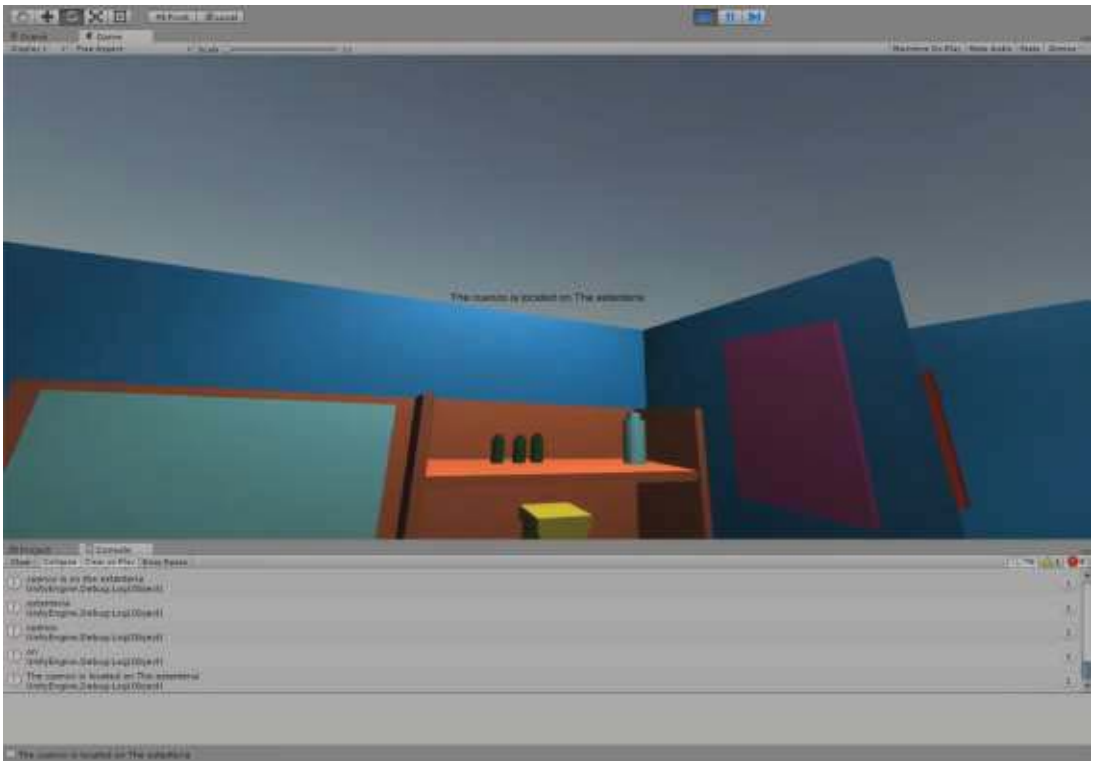


Figura 1.1 Sistema generador de indicaciones

Como ya se ha explicado en el resumen, la idea es llevar este sistema, basado en la Realidad Virtual, a un entorno de Realidad Aumentada.

Para ello, debemos hacer un estudio de las tecnologías disponibles en el mercado actual en lo referente a Realidad Aumentada. En el Capítulo II encontramos una investigación de dos de las principales alternativas de que disponemos, analizando las ventajas e inconvenientes asociadas a cada una. También, se hará un estudio para solventar la problemática de obtener con precisión la posición GPS de un usuario, incluyendo en interiores.

En el capítulo III desarrollaremos un mecanismo para localizar al usuario dentro del entorno real y poder trasladar esta posición al sistema que determina las relaciones espaciales que se dan entre los objetos y el usuario, con el fin de poder realizar pruebas. Serán analizados todos los detalles de la implementación, junto a los objetivos, dificultades encontradas y pruebas relacionadas.

Lo mismo será para el siguiente capítulo, el IV, en el que se describe a la adaptación del programa que tenemos a un modelo Cliente-Servidor con una *app* para Android desarrollada en Unity como cliente.

Veremos también cómo encajan estos primeros pasos dados con el proyecto vigente, a la par que una serie de pruebas diseñadas para comprobar el correcto funcionamiento de las partes diseñadas.

Herramientas

La totalidad del proyecto ha sido desarrollado en el entorno Unity 3D para asegurar la integración de la variante para realidad aumentada con el sistema ya existente. Es una potente herramienta pensada para el desarrollo de videojuegos, lo cual, como veremos más adelante, nos puede generar algún problema por no estar específicamente pensada para desarrollar sistemas como el nuestro, pero lo compensa con las facilidades que otorga en tareas como, por ejemplo, exportar un proyecto al formato SDK para el ecosistema Android.

Es importante destacar que, para evitar problemas de compatibilidad con el sistema disponible en Realidad Virtual, se utilizará Unity en su versión 5.1.f, en vez de la actual, la 5.7.

Figura 1.2 Logotipo Unity 3D

Para desarrollar en Unity se usará el lenguaje de programación C# junto librerías aportadas y diseñadas para Unity, lo cual nos da acceso a una amplia gama de funcionalidades que nos serán útiles para llevar a cabo el proyecto.

Para finalizar, en lo que se refiere a la escritura de esta memoria, los diagramas UML, de flujo, y de paso de mensajes han sido creados con la herramienta de código abierto Día.

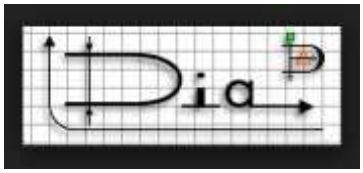


Figura 1.3 Logotipo Día

2.- OBTENCIÓN DE UN MODELO GEOMÉTRICO PARA EL MUNDO REAL

En la investigación de las alternativas disponibles para obtener un modelo geométrico del Mundo Real que nos permita aplicar los algoritmos ya existentes para el cálculo de saliencia, relaciones espaciales, etc., vamos a contemplar las siguientes opciones: Project Tango de Google y Kinect Fusion de Microsoft. Se complementará con un estudio de las diferentes soluciones para el posicionamiento en interiores. Se esclarecerán los usos y se listarán las ventajas e inconvenientes de estas tecnologías y métodos.

2.1.-Project Tango Google

Tecnología desarrollada por Google basada en la realidad aumentada, aún en desarrollo. Fue lanzada en 2014, y el primer y, de momento, único Smartphone o Tablet compatible es el Lenovo Phab 2 Pro. El propósito de este proyecto es imitar la percepción humana de la realidad dentro de nuestro dispositivo.

El dispositivo compatible con esta tecnología debe tener unos sensores en su(s) cámara(s) específicos, como un sensor de infrarrojos de movimiento, otro para medir la profundidad, y un sensor con gran ángulo de visión de 170 grados (ojo de pez). Dichos sensores recrean un modelo 3D dentro del terminal. La inclusión de todo esto hace que la resolución de la cámara sea menor en comparación con los estándares del mercado, pero es aceptable para el objetivo a buscar. [1], [2], [5], [7].

Esta tecnología se sirve también del acelerómetro y el giroscopio (herramientas que la mayoría de los terminales de hoy en día tienen, a diferencia de los sensores dichos anteriormente).

El procesador del hasta ahora único dispositivo compatible con esta tecnología tiene un procesador Snapdragon 652 de Qualcomm, modificado y adaptado para rendir más eficientemente para el ámbito de la realidad aumentada. [3], [4].

La capacidad de reconstruir un escenario es debido a la cámara IR y el proyector IR. Por un lado, la cámara infrarroja RGB puede absorber la luz del espectro de infrarrojos. Para interpretar la información del entorno, son claves las siguientes tecnologías: el aprendizaje en área, el seguimiento del movimiento y la percepción de profundidad.

Seguimiento del Movimiento (Motion tracking)

Permite determinar la posición y la orientación del dispositivo en la habitación. La posición determinada del dispositivo está compuesta por una coordenada xyz y de la orientación.

Para determinar un objeto en el entorno se usa la odometría (determinación y posicionamiento de un objeto móvil en una escena), una tecnología que hace uso de los ya mencionados acelerómetro y giroscopio, presentes en la gran mayoría de los smartphones de la actualidad.

Google, en Project Tango utiliza el concepto de Odometría Visual. En este procedimiento, el movimiento se deriva de la imagen de cámara de la cámara de gran angular. Con la detección de funciones en la imagen de la cámara, se puede calcular la ubicación del dispositivo. [10], [11].

Fuente: https://www.asus.com/Phone/ZenFone-AR-ZS571KL/

Aprendizaje del Área (Area Learning)

Almacenamiento de datos del entorno en un mapa que se puede volver a utilizar más adelante, compartida con otros dispositivos de Tango y mejorada con metadatos, tales como notas, instrucciones o puntos de interés. Esta tecnología, apoyándose en la cámara de gran angular u ojo de pez, permite al dispositivo reconocer su propia posición dentro de un área ya reconocida anteriormente.

La captura del área se hace grabando de manera lenta pero constante, permitiendo la detección y salvaguarda de las características observadas. Esta información es guardada en el archivo denominado ADF.

El aprendizaje del área está relacionado con el concepto anterior, *Motion Tracking,* debido a que mejora la precisión del seguimiento del movimiento. Ayuda a corregir la "deriva", es decir, el error de posicionamiento que va acumulándose con el paso del tiempo. Reconociendo las figuras y elementos del entorno, ayuda a recalcular y aproximar mejor la posición real del individuo. [1], [11].

Percepción de la Profundidad (Depth Perception)

Ésta tecnología es el modo del que disponemos para acercarnos aún más a la percepción del entorno que una persona es capaz de realizar. Para detectar la profundidad de los elementos del área que estamos estudiando, Project Tango usa los ya mencionados sensores de luz infrarroja RGB y el proyector IR.

La clave en los cálculos realizados para lograr este objetivo es la cantidad de luz sobre cada ítem de la escena, y el *Time-of-Flight,* o tiempo que tarda en ir y volver el rayo infrarrojo lanzado hacia un elemento. [1], [11].

Usos y aplicaciones de Tango:

Project Tango tiene cada vez más aplicaciones que hacen uso de su tecnología. Existen usos educativos, comerciales, de entretenimiento, en el ámbito inmobiliario, para diseñadores y de carácter social, ayudando a personas con discapacidad visual a detectar obstáculos.

Desarrollo:

Todas las pruebas del mismo sólo se pueden hacer en un dispositivo compatible con Project Tango, lo cual supone la mayor limitación a la hora de empezar a desarrollar.

Hay espacios dedicados tanto para Java, C, C++ como para Unity. En principio nos vamos a centrar en Unity. La versión de Unity tiene que ser 5.2.1 en adelante, con lo que encaja con la versión que hemos utilizado (Unity 3D 5.2.1f1). Requiere de Android 4.2 o superior, equivalente a un nivel 17 del APK, lo que no es un problema debido a la antigüedad de ese sistema operativo (existen numerosas versiones posteriores; 4.3, 4.4, 5.0, 5.1, 6.0, 6.0.1, 6.1, 7.0 y la 7.1). [1], [8], [9].

Google facilita este tutorial para la realización de una app sencilla:
https://developers.google.com/tango/apis/unity/unity-howto-motion-tracking

Ventajas:

La ventaja principal de Project Tango es que es de Google, lo que nos abre esta tecnología al ecosistema Android, el sistema operativo que llevan más del 80% de los smartphones del mundo actualmente. Esto conlleva el traer la tecnología de la realidad aumentada a una gran parte de la población de una manera muy directa.

Otra ventaja es la facilidad para desarrollar que nos aporta la compañía del famoso buscador, adaptados a lenguajes y entornos muy utilizados entre los desarrolladores, sobre todo con Unity, nuestra principal herramienta de trabajo.

Inconvenientes:

En interiores, los datos que nos proporciona son fiables y precisos, pero en un escenario en el exterior, la capacidad de detectar la profundidad mengua, y empezamos a tener un error más notable.

Las condiciones lumínicas deben ser bastante favorables, o tendremos otro problema semejante. Es algo habitual para los sensores de las cámaras de los smartphones actuales que sufran en condiciones de baja luminosidad, ya que pierden información, pero en nuestro caso, ésta pérdida de información es aún más notoria ya que dejamos de percibir la profundidad de los elementos de una escena.

Por otro lado, la fiabilidad de entornos pre-programados sólo es fiable si no ha habido grandes cambios en la escena. En caso contrario, dicha escena podría no ser directamente detectada.

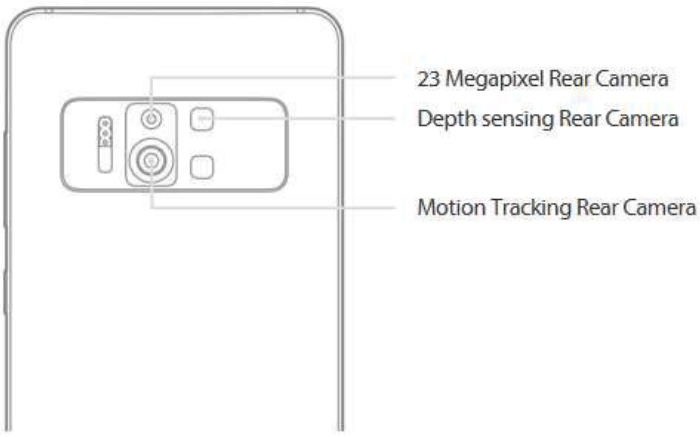
La necesidad de los elementos hardware descritos anteriormente, en especial los de la cámara, suponen una clara desventaja. Sin embargo, teniendo en cuenta que Lenovo ha creado para otros dispositivos móviles un sistema modular para añadir más batería, un zoom óptico real (no por software), los llamados "motomods", que se incorporan al Smartphone de manera sencilla con un mecanismo de imanes, no es una locura pensar en un motomod que añada estas características de manera rápida y fácil a cualquier Smartphone actual.

Por otro lado, y partiendo de lo que acabamos de ver, la otra gran desventaja es que, a fecha de 2017, sólo un único Smartphone es compatible con esta tecnología, siendo bastante complicado de adquirir. Además, tiene un coste muy elevado (en torno a 500€), es un dispositivo demasiado grande y pesado para el estándar actual, aunque bien su pantalla de gran resolución es clave para el uso de la realidad virtual. Su procesador, pese a estar optimizado para Project Tango, ya queda algo atrasado con los nuevos modelos de SnapDragon, el fabricante por excelencia de los procesadores para smartphones en el mundo Android.

Para finalizar, recalcar el calentamiento que todo smartphone sufre con usos intensivos, puede ser un aspecto importante a tener en cuenta aquí, pese a la modificación del procesador. El uso de la realidad aumentada supone un esfuerzo considerable para nuestro dispositivo, y sin lugar a dudas esto supone un calentamiento muy fuerte, ayudado del calor corporal propio del usuario.

* Nota: como novedad en el primer trimestre de 2017, se ha añadido al Lenovo Phab 2 Pro, el Asus ZenFone AR como únicos smartphones con Project Tango. Como un breve resumen del mismo, tenemos un dispositivo con una cámara de 23 Mpx que tiene incorporado un sensor de profundidad. Tiene una diagonal de pantalla de 5'7 pulgadas, ideal para la realidad aumentada, con tecnología WQHD (2560x1440) Super AMOLED, lo que hace de este panel lo mejor del mercado para disfurtar de AR. Su procesador también está optimizado para Tango, y, aunque es del mismo fabricante, Qualcomm, se trata de uno de última generación, el Snapdragon 821.

Con estas características, tenemos que el segundo Smartphone adaptado para Project Tango es un modelo engoblado en la gama alta. Su precio, a priori (ya que a fecha de edición es bastante novedoso y puede variar), es el mismo que el del Lenovo Phab 2 Pro, esto es, alrededor de los 500€. Es por tanto, que una hipotética elección entre ambos dispositivos, nos decantaríamos claramente a favor del el nuevo Asus Zenfone AR.



Fuente: https://www.asus.com/Phone/ZenFone-AR-ZS571KL/Tech-Specs/

2.2.-Kinect Fusion

Originalmente conocido como Project Natal, Kinect es un controlador de juego para PC y la consola XBOX (versiones 360 y ONE) desarrollado por Alex Kipman para la empresa Microsoft y sacada a la venta en 2011. Está diseñada para permitir al usuario controlar e interactuar con el juego con gestos, objetos y comandos de voz.

Dispone de un sensor con forma de barra horizontal de aproximadamente 23 cm, conectada a una pequeña base circular con un eje de articulación de rótula, diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla de vídeo.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un procesador personalizado que ejecuta el software patentado, el cuál proporciona una captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz. La cámara tiene una frecuencia de captura de fotogramas lo bastante alta (30 Hz) como para objetos en movimiento minimizando la distorsión.

El sensor de profundidad es un proyector de infrarrojos combinado con un sensor CMOS monocromo que permite ver la habitación en 3D en cualquier condición de luz ambiental. El rango de detección de la profundidad del sensor es ajustable gracias al software de Kinect capaz de calibrar automáticamente el sensor. La versión 2 tiene tres fuentes de luz infrarroja las cuales generan ondas con diferente amplitud. [12], [13], [14], [15].

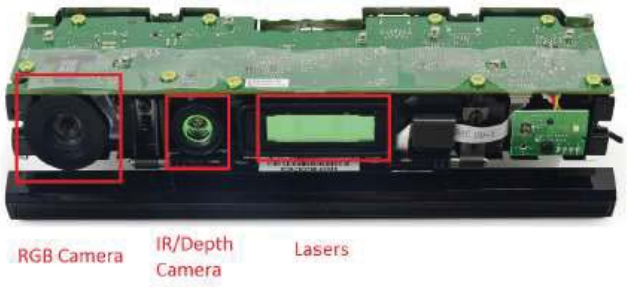


Figura 2.1 Ubicación de láseres de luz infrarroja en Kinect v2.

Desarrollo

Microsoft dispone de un SDK para diseñar aplicaciones para Kinect. [13].
Aquí se puede encontrar más información al respecto:
https://developer.microsoft.com/es-es/windows/kinect/develop

Ventajas:

Independencia de la cantidad de luz en la escena, gracias a su sensor de profundidad con el CMOS monocromo comentado anteriormente. En comparación con otros escáneres 3D, es muy asequible económicamente hablando, y la hace una pieza muy recomendable para el modelado 3D.

El principal problema de Kinect es el poco tiempo que tiene para realizar una precisa medición de parámetros como la profundidad debido a que está pensado para capturar el movimiento pero, en nuestro caso, eso no es un inconveniente dado que no necesitamos la interpretación de gestos en el proyecto.

Inconvenientes:

La principal desventaja que tiene este sistema es la incomodidad de usarlo en un ambiente exterior, dejando prácticamente como nula esta opción.
Investigadores del ámbito del modelado 3D aún no han obtenido resultados decentes usando la Kinect (v2) como herramienta principal.
El sensor contiene un mecanismo de inclinación motorizado y, en caso de usar un Xbox 360 del modelo original, tiene que ser conectado a una toma de corriente, ya que la corriente que puede proveerle el cable USB es insuficiente; para el caso del modelo de Xbox 360 S esto no es necesario ya que esta consola cuenta con una toma especialmente diseñada para conectar el Kinect y esto permite proporcionar la corriente necesaria que requiere el dispositivo para funcionar correctamente.

2.3.-Posicionamiento en Interiores:

La gran mayoría de smartphones de la actualidad llevan sistemas ubicuos que permiten una buena localización en exteriores pero cuya efectividad se ve reducida drásticamente cuando hablamos de escenas situadas en interiores. Una precisión con un margen de error de unos metros es suficientemente buena para exteriores, pero dentro de una oficina, por ejemplo, es un error demasiado abultado y que crece a medida que pasa el tiempo.

Figura 2.2 Indoor Positioning

Para paliar este problema, existen varios métodos para hallar el posicionamiento en interiores (*Indoor Positioning Methods*). De entre las tecnologías más efectivas para dicha tarea y que sean adecuadas para el uso en smartphones tenemos:

-Uso de señales de radiofrecuencia (RF), ya sea las ya existentes como las redes inalámbricas de área local (WLAN), o celulares, o las generadas por la nueva y dedicada infraestructura (RFID / NFC, Bluetooth).
-Sensores independientes: acelerómetro de tres ejes, giroscopio de tres ejes, magnetómetro de tres ejes, barómetro…
-Mapa interior (plano del edificio).
-Impresión magnética de archivos.

Ninguna de estas tecnologías es una solución completa para todas las necesidades de posicionamiento en interiores en dispositivos móviles actuales y sus ventajas y desventajas dependen de la naturaleza de cada tecnología. Por ejemplo, las soluciones basadas en UWB y RFID requieren el despliegue de nueva infraestructura y la adición de nuevos componentes de hardware a los teléfonos inteligentes. Actualmente, estos costes adicionales limitan el uso de soluciones basadas en etiquetas a Bluetooth Low Energy (BLE).

Otro ejemplo es el uso de la radio. La intensidad de la señal recibida (RSS) o el indicador de intensidad de la señal recibida (RSSI), es una métrica basada en la potencia que mide la intensidad de la señal transmitida en el lado del receptor. RSS es medido típicamente por todos los dispositivos sin hilos para el propósito de la comunicación de datos, por lo tanto el usarlo como una métrica de colocación no requiere ningún hardware adicional y así RSS se ha utilizado popular en muchos sistemas de colocación. Sin embargo, debido a que la señal de radio generalmente se vuelve más débil con mayor distancia, RSS puede ser utilizado para predecir la distancia entre los transceptores. Sin embargo, RSS sufre propagación de múltiples caminos, que hace cambiar la amplitud de la señal de una manera compleja. Como resultado, la asignación de valores RSS en distancias no es una tarea sencilla.

Actualmente, la mayoría de los teléfonos inteligentes utilizan la colocación en interiores basada en WLAN que por lo general satisface los requisitos de precisión de posición en áreas ricas en routers WLAN. Sin embargo, la precisión depende en gran medida de la densidad de los routers WLAN y de la calidad del mapa radioeléctrico que suele deteriorarse con el tiempo y debe actualizarse periódicamente

Los problemas usando sistemas inalámbricos de posicionamiento (como son WiFi, Bluetooth, WLAN, WPAN, etc) pueden ser clasificados en tres grandes grupos; por la variación espacial, por la variación temporal, por la variación del dispositivo.
La variación espacial y temporal son causadas por factores externos tales como el radio medio de propagación o la presencia de objetos alrededor de la trayectoria de propagación que cambian la amplitud y la fase de la señal, mientras que la variación del dispositivo es causada por las características intrínsecas del propio hardware del dispositivo.

Las soluciones, a su vez, se pueden separar en modelos paramétricos y modelos no paramétricos.

Los modelos paramétricos

En estos modelos, el posicionamiento es calculado como una ecuación cuyos componentes reflejan las condiciones ambientales de la escena. El modelo de propagación de radio más complejo conocido como modelo de *ray-tracing,* utiliza la información detallada de los diseños del sitio y la especificación de los transceptores. El modelo considera todas las rutas de propagación posibles que pueden tener lugar entre los transceptores para obtener una métrica de posicionamiento más precisa.

Aunque el modelo de *ray-tracing* representa mejor el comportamiento de propagación de radio, actualizar el modelo para adaptarse a los cambios en los diseños del sitio requiere una actualización de posicionamiento periódica que sería poco práctica si tratamos en un entorno altamente dinámico. En otras palabras, este modelo sólo es adecuado para manejar la variación espacial, pero no para acomodar la variación temporal.

Los modelos no paramétricos

En lugar de modelar el comportamiento de propagación de radio utilizando los supuestos de condiciones ambientales similares a los modelos paramétricos, los modelos no paramétricos aprenden de los datos recogidos del entorno real. Al aprender de los datos, el modelo no paramétrico puede ajustarse de acuerdo con el estado actual de la escena y resolver los problemas de variación espacial y temporal. Los modelos no paramétricos se implementan con técnicas deterministas o probabilísticas.

El modelo no paramétrico se realiza principalmente a través de la huella dactilar de ubicación, donde los datos de posicionamiento recogidos durante la fase offline se almacenan y se emparejan con los datos de posicionamiento en fase en línea para identificar la posición del dispositivo, rastreado durante la fase en línea.

Otro tipo de modelo no paramétrico se basa en un modelo de propagación de radio como en el modelo paramétrico, aunque en él los parámetros no están preconfigurados, sino aprendidos de los datos en su lugar. [17], [18], [19], [20], [21].

Ventajas

Al tratarse de métodos científicos puramente dichos, su uso no está ligado a la comercialización de un producto por una marca, por lo que son, en cierto grado, más asequibles económicamente hablando.
La variedad de métodos y conjuntos de métodos nos aporta una selección del que mejor se aplique en nuestro caso, evitando un uso al que no estaba dirigido un producto cuando es comercializado.

Inconvenientes

Como hemos explicado anteriormente, no se han hecho grandes desarrollos que sean compactos y decentes aún. Faltan por pulir los métodos, la eficiencia de los mismos está lejos de ser rentable desde el punto de vista de un desarrollador.
La complejidad de algunos métodos obliga a tener un amplio conocimiento en términos relacionados con ondas, señales, radio, estadística, probabilística, determinismo, etc.

La solución requerida podría estar a medio camino para ser resuelto por métodos paramétricos y/o métodos no paramétricos, debido a que los factores que nos hacen escoger pueden ser variables (iluminación, dinamismo de la escena, etc).

2.4.-Valoraciones y Conclusiones de la investigación

Tras el estudio y la profundización en las tecnologías descritas anteriormente, debemos destacar a Project Tango como la solución que mejor se posiciona para cubrir nuestras necesidades. Habrá que esperar un tiempo para que se extienda aún más su comercialización, así como el número de smartphones y dispositivos compatibles, pero con la gran importancia que está cogiendo la realidad virtual y la aumentada, parece que se pueden acortar los plazos.

Project Tango nos aporta facilidades para el desarrollo con nuestro entorno clave; Unity, nos genera un modelo geométrico y, aunque nos restringe al ámbito Android, nos lleva a más del 80% de los dispositivos móviles inteligentes del mercado.

La tecnología alternativa de Kinect Fusion nos da la ventaja de no depender tanto de las condiciones lumínicas de la escena a tratar, problema que se entiende será cada vez menos grave según avanza la tecnología de los sensores de las cámaras fotográficas de los smartphones. Como contrapartida, la posible necesidad de tener que usar varios dispositivos Kinect, incrementando costes, la necesidad de una toma de corriente para algunas versiones, y los pocos avances decentes que se han conseguido usando dicha tecnología, relega a Kinect de la opción más preferible en nuestro caso.

Para conseguir el menor error a la hora de la obtención del posicionamiento general del usuario, tanto en interiores como en exteriores, será necesario combinar la tecnología de Project Tango con alguna de las técnicas vistas de Posicionamiento en Interiores.

3.- DISEÑO DE UN MECANISMO PARA LA LOCALIZACIÓN DEL USUARIO

La localización del usuario dentro del mundo real es un apartado fundamental en nuestro trabajo. Como ya hemos visto en el capítulo anterior, en interiores nos encontramos con severas dificultades para posicionar al usuario, dado que tecnologías como el GPS no funcionan correctamente.

Mientras no se puedan llevar a la práctica los resultados extraídos de la investigación vista en el capítulo anterior, solventaremos este problema con una aplicación móvil que devuelva en tiempo real las coordenadas del geo-posicionamiento del usuario.

3.1.-Requisitos

En este apartado vamos a ver los requisitos y objetivos que se deben cumplir en lo que a localización y posicionamiento del usuario se refiere.

Es importante tener en cuenta que esta parte estará dentro de un sistema más complejo, que incluirá un modelo Cliente-Servidor con paso de mensajes entre otras funcionalidades. Para hacer más sencillo el visionado y comprensión de cómo funciona esta parte, se mostrará el prototipo desarrollado para hacer pruebas de manera individual relativas a la localización del usuario. Es por tanto que la implementación que se va a presentar en este apartado es la de dicho prototipo, ajena al resto del sistema. Su implementación final, junto al resto, será visible en el siguiente capítulo.

Objetivos y requisitos a cumplimentar en la sección de localización del usuario:

1.-La aplicación deberá dar en tiempo real la posición del usuario en el mundo real a través de la señal GPS.

2.-Los datos cuantitativos a transmitir al servidor serán dos; la longitud y la latitud.

3.-Los datos serán obtenidos de la herramienta Google Maps, debido a las facilidades que otorga cuando estamos desarrollando para Android y su inclusión en todos los dispositivos de este ecosistema.

4.-La aplicación deberá pedir permisos de lectura de la posición GPS del dispositivo.

5.-La aplicación deberá comprobar que la ubicación esté activada por el usuario antes de pedir y/o enviar ningún dato.

6.- [exclusivo del prototipo] la aplicación contará con una sencilla interfaz de usuario con la que mostrar por pantalla las coordenadas GPS, latitud y longitud, que está obteniendo en cada momento, con el fin de realizar pruebas de su correcto funcionamiento.

3.2.-Preparación y configuración para el desarrollo para Android en Unity 3D

Desarrollaremos la aplicación en lenguaje de programación C# en Unity 3D para dispositivos Android. Para ello debemos configurar nuestra herramienta Unity para el desarrollo de aplicaciones móviles en Android.

Primeramente debemos hacernos con el SDK de Android. El SDK es un conjunto de herramientas de desarrollo que contiene por ejemplo un depurador, un emulador y bibliotecas para poder empezar a trabajar en Android. Para obtener el SDK podemos descargar Android Studio, el entorno de trabajo desarrollado por Google para programar en Android, el cual ya contiene el SDK, o bien podemos descargarlo directamente desde otro enlace en la red. En nuestro caso, debido a que vamos a trabajar con Unity como entorno de programación, descargaremos el SDK mediante un enlace externo. [23], [25].

Muy importante, si al instalar se da la opción de elegir donde guardar el SDK, se debe hacer con un *path* que sea fácil recordar, ya que más adelante lo necesitaremos para indicar a Unity dónde debe buscarlo. Es posible que la carpeta donde se guarde esté oculta. En ese caso, se deberá hacerla visible, proceso sencillo pero que puede tardar unos minutos en completarse.

Podemos indicar a Unity dónde encontrar el SDK de Adroid en *Build Settings -> Android -> Build* y dar el correspodiente *path* hasta el SDK para finalizar. Si es necesario, podremos cambiar el *path* asignado en *File -> Preferences -> External Tools*.

Recomiendo también la instalación del NDK de Android. Es otro conjunto de herramientas que sirven para la combinación de lenguajes de programación, y puede ser de utilidad. Podemos sugerir su localización a Unity de la misma manera que como hemos explicado con el SDK

Para poder probar en un dispositivo Android nuestros proyectos, debemos primeramente ser desarrolladores. En mi caso, con la última versión actual del sistema operativo, Android 7.1.1, esto se hace en Ajustes -> Información del Teléfono -> presionar unas cuantas veces sobre el número de compilación. Una vez hecho esto, tendremos la opción de "Depuración USB" en Ajustes -> Opciones de Desarrollo.

Ahora, debemos obtener de la Play Store la app Unity Remote. Cabe destacar que el manual de Unity habla sobre la versión 4, pero ya es oficial la versión 5, y es con ella con la que trabajaremos.

En Unity, deberemos irnos a *Edit > Project Settings > Editor*, y marcar como accesible todo dispositivo con Android.

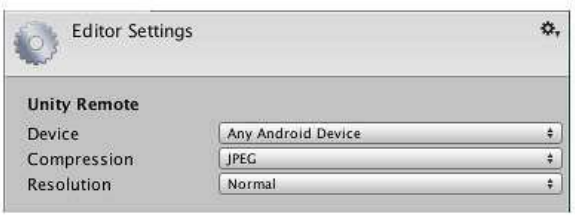


Figura 3.1 Editor Settings Unity 3D

Con el proyecto abierto, conectaremos el dispositivo Android al ordenador mediante un cable USB. Recomiendo la utilización de un USB type C si el dispositivo es compatible con dicho estándar, ya que mejora la experiencia de simulación. Ya sólo queda en Unity dar al botón de *Play* y podremos ver cómo quedaría la *app*.

Es posible que el ordenador no reconozca el terminal conectado. El problema radicará en los *drivers* instalados. Muy posiblemente, si esto es así, los instale automáticamente el propio ordenador. Si el error persiste, deberíamos buscar en la red los *drivers* más actualizados respecto a nuestro terminal.

En lo que al desarrollo en Unity se refiere, debemos seguir, de manera muy importante y obligatoria: *build settings -> Android -> switch plattform*; para poder programar para android y que sea compatible. [22]

Para desarrollar en Android desde Unity no es necesario un tipo de *scripting* distinto al usado normalmente. Al haber hecho el *switch plattform,* Unity ya realiza los cambios necesarios para que nuestro código funcione para Android.

Cuando hemos terminado la aplicación correspondiente, es necesario en *Build Settings -> Player Settings,* indicar los aspectos relacionados con la publicación y propietarios de la *app*.

Se debe mantener y respetar la forma <com.NombreEmpresa.NombreProducto>. Adicionalmente, para evitar que nos copien la aplicación, podemos añadir una clave de seguridad.

Si se han seguido todos los pasos de configuración y preparación para desarrollar en Android desde Unity vistos en la primera sección de este capítulo, podremos obtener el archivo APK para llevar a nuestro dispositivo Android e instalar.

3.3.-Implementación

Nos encontramos en un proyecto Unity en 3D. Sólo se requerirá una escena, en la cual tendremos un *canvass* sencillo, con un panel y varios textos en los que sobreimpresionar los datos a mostrar por la aplicación.

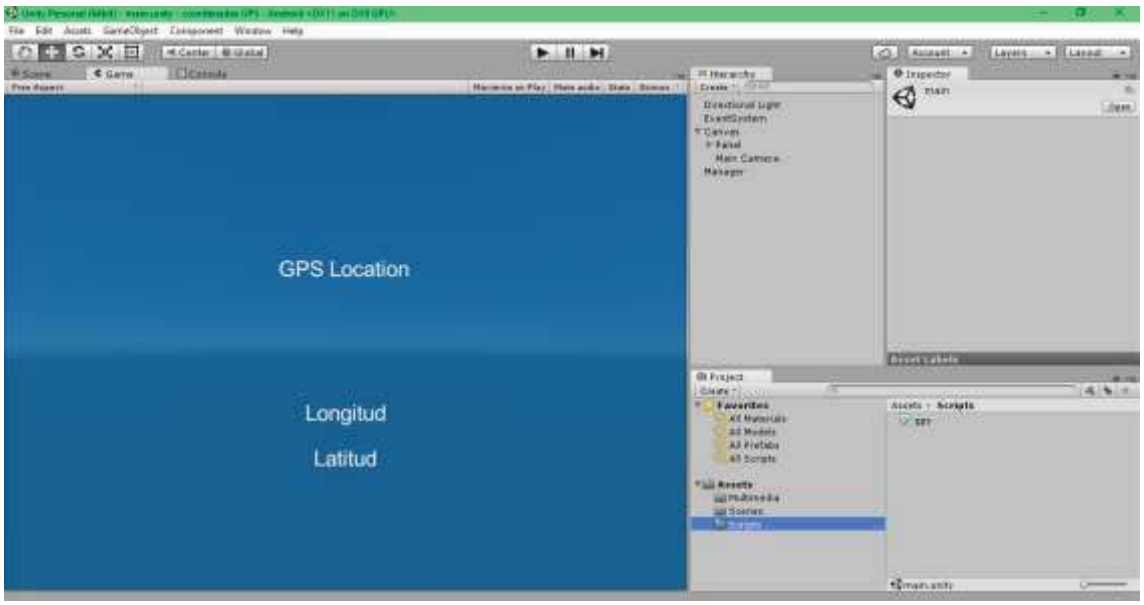


Figura 3.2 Vista general Prototipo Localización GPS

Figura 3.3 Vista de la jerarquía del prototipo

En lo que a *scripting* se refiere, disponemos de una clase en C# en la que, a las bibliotecas que vienen referenciadas por defecto, añadiremos UnityEngine.UI para controlar los elementos de la interfaz de usuario.

```
/* Autor:    Pablo Aceituno Ferro
 *           t110115
 * Proyecto: TFG
 *           Localizacion GPS
*/

using UnityEngine;
using System.Collections;
using UnityEngine.UI;
```

La clase tendrá un comportamiento *MonoBehaviour*, ya que en este prototipo no necesitamos de un comportamiento válido para trabajar en red, como en la versión final, que será *NetworkBehaviour*.

Tendremos dos variables globales en las que guardaremos los datos de la longitud y latitud para mostrar.

Dentro del método *Start*, el clásico método de Unity que es ejecutado nada más arrancar el programa o aplicación, debemos lanzar una co-rutina que, como veremos más adelante, inicie el servicio de localización.

```
public class gps : MonoBehaviour {

    public Text latitudeText;
    public Text longitudeText;

    void Start () {
        StartCoroutine ("getLocation");
    }
```

El servicio primeramente comprobará si la ubicación está activada. Si fuera el caso, lanzaría el servicio de localización de forma permanente, obteniendo los datos en tiempo real. En el sistema desarrollado, esta co-rutina es mejorada en lo que a eficiencia se refiere para que las actualizaciones de las coordenadas estén más controladas y sea más eficiente. [24]

```
    IEnumerator getLocation () {

        // Comprobamos si la localizacion esta acivada
        if (!Input.location.isEnabledByUser) {
            yield break;
        }

        else {
            while(true){
                Input.location.Start ();
                float latitude  = Input.location.lastData.latitude;
                yield return latitude;
                float longitude = Input.location.lastData.longitude;
                yield return longitude;
                latitudeText.text = "Latitud:" + latitude;
                longitudeText.text = "Longitud:" + longitude;
                yield return new WaitForSeconds(3);
            }
        }
    }
}
```

Como era requerido, al instalar dicha aplicación se pide que se acepten los permisos referidos a obtener la ubicación del usuario.

3.4.-Diseño de Pruebas y Ejecución

Se han realizado sencillas pruebas para determinar el correcto funcionamiento del prototipo de obtención de las coordenadas GPS. A continuación se muestran varias capturas de cómo se ve la aplicación en el móvil.

Las pruebas realizadas consistían en arrancar la aplicación una vez dentro del autobús 591 que lleva de Aluche a la Escuela, e ir comprobando cómo, dentro del vehículo en marcha, iban cambiando las coordenadas mostradas. Dichas pruebas se realizaron de esta forma para obtener cambios de latitud y longitud pronunciados debido a que en las pruebas en interiores cambia muy poco, y la precisión mostrada en pantalla (del orden de 10 elevado a -2 ó -3) era menor que los cambios realmente producidos (del orden de 10 elevado a -6 ó -7).

Figuras 3.4, 3.5 y 3.6 Pruebas app localización GPS

4.- ADAPTACIÓN DEL SISTEMA A UN MODELO CLIENTE-SERVIDOR

Disponemos de un sistema en realidad virtual en el que hay implementadas una gran variedad de utilidades, con el fin de calcular la saliencia de los objetos, para poder así dar indicaciones de ayuda en la localización de objetos. El fin de este capítulo es adaptar este sistema para la realidad aumentada, partiendo de la realidad virtual.

Una de las claves de la realidad aumentada es la localización del usuario. El servidor debe conocer dónde se encuentra ubicado el usuario en todo momento.

Para ello, dispondremos de un cliente en una aplicación móvil y un servidor ejecutando en un computador. La intención final es la de que el cliente envíe de manera periódica una actualización de su posición GPS (la cual hemos visto cómo obtenerla en el capítulo anterior) al servidor, el cual, ante una petición de ayuda por parte del usuario, le enviará una serie de indicaciones para que localice objetos del entorno real, teniendo en cuenta la posición de cliente. Así, podrá en el futuro generar indicaciones que ayuden a localizar cada objeto, dependiendo de si está enfrente, a la derecha o izquierda, detrás, etc.

4.1.-Requisitos

Los requisitos a implementar en el modelo Cliente-Servidor son:

1) El sistema debe registrar y conectar a la aplicación que se ejecuta en la máquina, que hará de Servidor, y a (al menos) una instancia móvil que ejecutará el papel de Cliente. Como veremos más adelante, la implementación realizada permite más de un cliente, lo cual puede ser de utilidad en el futuro si se necesita.

2) Una vez registrados los componentes de la comunicación, el cliente empezará a enviar su ubicación (coordenadas GPS: latitud y longitud obtenidas de Google Maps) en períodos de 2 segundos.

3) El cliente tendrá la opción de enviar un mensaje adicional, diferente en cuanto a contenido de los que actualizan la ubicación, en el que pedirá indicaciones para poder localizar un objeto dentro de la escena.

4) El servidor deberá saber diferenciar entre los mensajes periódicos de envío de la localización y los de petición de indicaciones.

5) Cuando el servidor recibe un mensaje con la ubicación GPS del usuario, actualizará la posición y orientación del punto de vista con el que se observa el modelo geométrico del mundo real, sin enviar ningún mensaje de vuelta al cliente.

6) Cuando el servidor recibe un mensaje de petición de indicaciones, contactará con la parte ya realizada en trabajos anteriores y solicitará una generación de indicaciones teniendo en cuenta la última posición obtenida del cliente. Enviará a dicho cliente un mensaje incluyendo la indicación o indicaciones generadas.

7) De manera complementaria, se creará un registro visual de los mensajes intercambiados por los módulos con el fin de conocer en tiempo real las interacciones entre el cliente y el servidor, distinguiendo quién es el emisor de cada mensaje.

4.2.-Diseño y Arquitectura

A continuación disponemos de un diagrama que explica la arquitectura y el paso de mensajes en el modelo Cliente-Servidor a implementar.

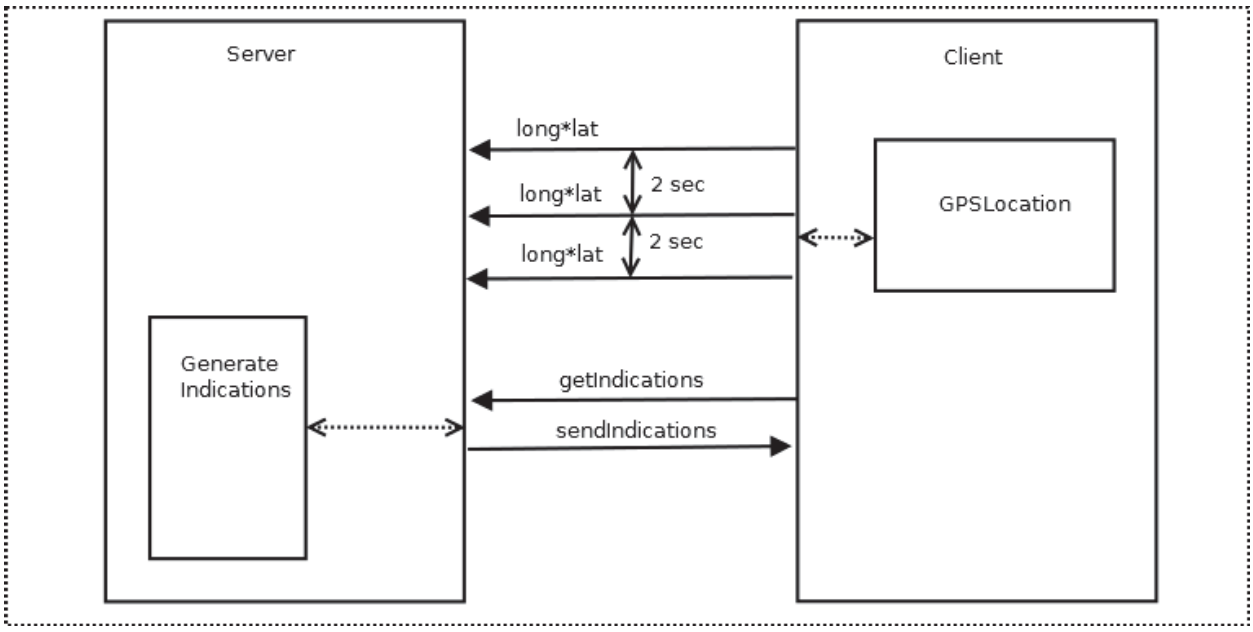


Figura 4.1 Vista modelo Cliente-Servidor

Para lograr un mayor entendimiento de la estructura e interconexión de módulos, a continuación disponemos de un diagrama adicional, que no pretende sustituir a un diagrama UML, que veremos más adelante en la subsección de implementación, pero sí arrojar más luz sobre el funcionamiento y distribución del planteamiento propuesto.

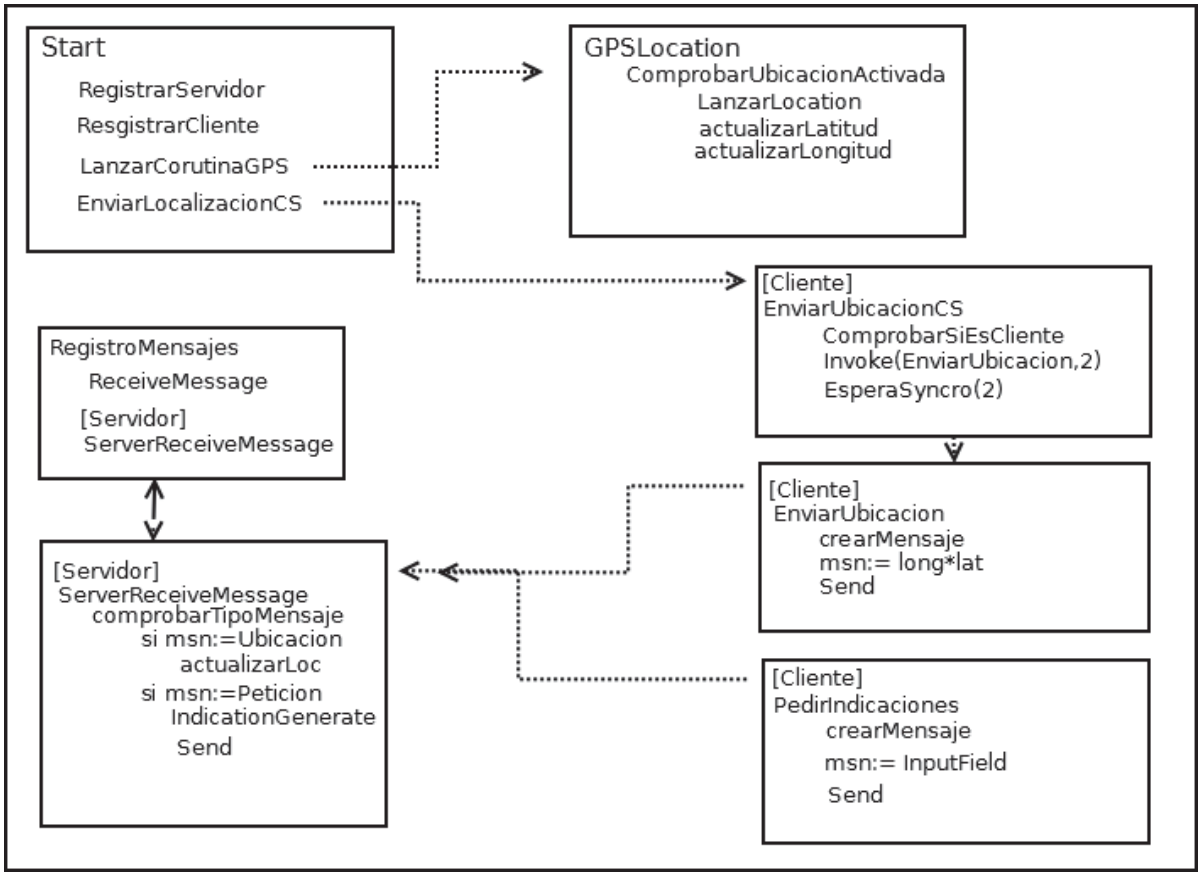


Figura 4.2 Comunicación entre módulos y servicios

Adicionalmente, a continuación disponemos de un diagrama UML que recoge el diseño de los comportamientos del proyecto.

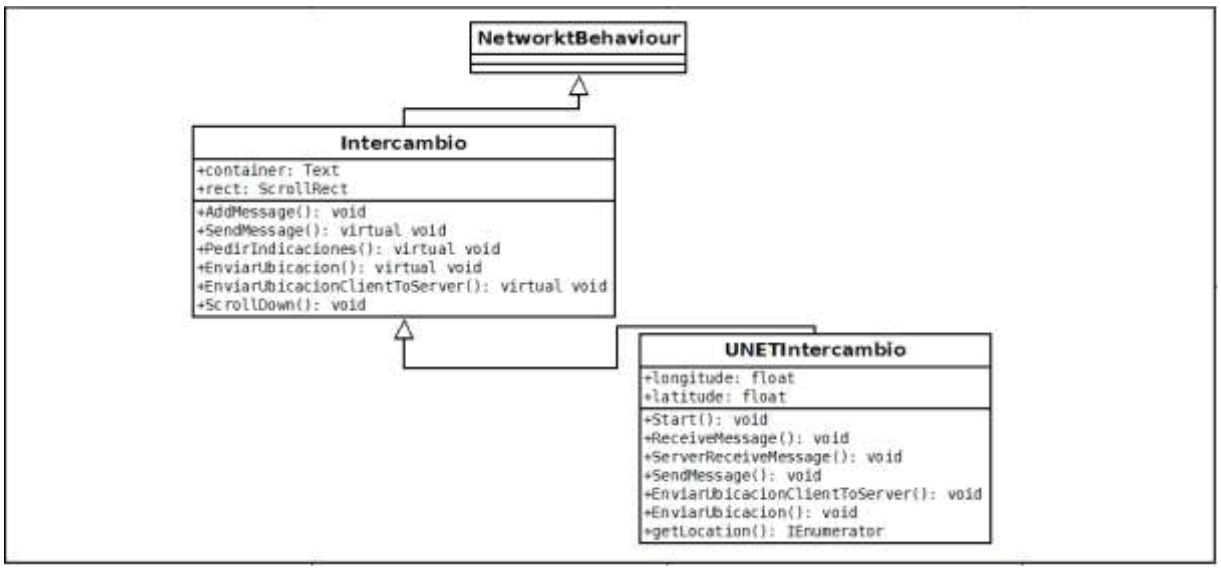


Figura 4.3 Diagrama UML

4.3.-Implementación

Para implementar un modelo Cliente-Servidor, primeramente realicé el tutorial avanzado que Unity ofrece en el siguiente enlace:
https://unity3d.com/es/learn/tutorials/topics/multiplayer-networking

Dicho tutorial me ayudó a comprender cómo funciona el comportamiento en redes de Unity, así como elementos fundamentales como el *Network Manager*. Sin embargo, como ya detallo en las valoraciones escritas al final de la memoria, al estar Unity enfocado en el desarrollo de videojuegos, este tutorial, y mucha documentación adicional que busqué, está orientado a este ámbito y no resulta del todo útil para el propósito que estábamos buscando.
Es por tanto, que tras investigar a fondo en la documentación que ofrece Unity, además de información adicional encontrada en la web [26], [27], [28], [29], [30], [31], he elaborado la siguiente documentación propia que valdría para proyectos en general.

Programación de un modelo Cliente-Servidor en Unity

Usaremos la *Transport Layer API* para el paso de mensajes. La capa de transporte es una capa delgada que funciona sobre la red basada en *sockets* del sistema operativo. De las dos APIs que Unity ofrece para el *Networking*, ésta es la más avanzada.

La capa de transporte admite servicios de base para la comunicación en red. Estos servicios básicos incluyen:

- Establecer Conexiones.
- Comunicarse con una variedad de "calidad de servicios".
- Control de flujo.
- Estadísticas de base.
- Servicios adicionales como comunicación a través de servidor de retransmisión o descubrimiento local.

La capa de transporte puede utilizar dos protocolos; UDP para comunicaciones genéricas y WebSockets para WebGL. Para utilizar la capa de transporte directamente, el flujo de trabajo típico a seguir sería el siguiente:

1. -Inicializar la capa de transporte de red
2. -Configurar la topología de red
3. -Crear anfitrión
4. -Iniciar la comunicación (manejo de conexiones y envío / recepción de mensajes)
5. -Apagar la conexión después del uso.

El motor de *networking* en Unity iOS/Android es completamente compatible con el *networking* para dispositivos móviles. Por lo tanto, ninguna modificación es necesaria para poder exportar en nuestro caso el proyecto como aplicación móvil (APK para el cliente) o como ejecutable para ordenadores.

1.-Inicializar la capa de transporte de red:

Al inicializar la capa de transporte de red, podemos elegir entre la inicialización predeterminada (sin argumentos), o podemos proporcionar parámetros que controlan el comportamiento general de la capa de red, como el tamaño máximo de paquete o el límite de tiempo de subproceso. La opción que tiene argumentos es más personalizada, útil para ciertos servicios específicos.

Ejemplo sin argumentos:

```
// Initializing the Transport Layer with no arguments (default settings)
   NetworkTransport.Init();
```

Ejemplo con argumentos:

```
// initializing the Transport Layer with custom settings
   GlobalConfig gConfig = new GlobalConfig();
   gConfig.MaxPacketSize = 500;
   NetworkTransport.Init(gConfig);
```

2.- Configuración:

```
   ConnectionConfig config = new ConnectionConfig();
   int myReiliableChannelId  = config.AddChannel(QosType.Reliable);
   int myUnreliableChannelId = config.AddChannel(QosType.Unreliable);
```

"QosType.Reliable" enviará mensajes y asegurará que el mensaje se entregue, mientras que "QosType.Unreliable" enviará mensajes sin ninguna garantía, pero lo hará de una manera más rápida.

También podemos especificar ajustes de configuración específicos para cada conexión, ajustando las propiedades en el objeto ConnectionConfig. Sin embargo, al realizar una conexión de un cliente a otro, la configuración debe ser la misma para ambos pares conectados o la conexión fallará con un error "CRCMismatch".

3.- Topología:

La topología de red define cuántas conexiones se permiten y qué configuración de conexión se utilizará.

```
HostTopology topology = new HostTopology(config, 10);
```

Aquí habríamos creado una topología que permite hasta 10 conexiones, cada una de ellas configuradas por los parámetros definidos en el paso anterior.

4.- Creación del host (anfitrión):

```
        int hostId = NetworkTransport.AddHost(topology, 8888);
```

Aquí añadimos un nuevo host en el puerto 8888 y cualquier dirección IP. Este host soportará hasta 10 conexiones, y cada conexión tendrá parámetros como los definidos en el objeto config.

5.- Iniciar la Comunicación:

Con el anfitrión creado, podemos iniciar nuestra comunicación. Para ello enviamos comandos diferentes para alojar y comprobar su estado. Hay 3 comandos principales que podemos enviar:

```
connectionId = NetworkTransport.Connect(hostId, "192.16.7.21", 8888, 0, out error);
NetworkTransport.Disconnect(hostId, connectionId, out error);
NetworkTransport.Send(hostId, connectionId, myReiliableChannelId, buffer, bufferLength,  out error);
```

El primer comando enviará la petición de conexión a un *host* con ip "192.16.7.21" y en el puerto 8888. Retornará el identificador asignado a esta conexión.
El segundo enviará la solicitud de desconexión.
El tercero enviará un mensaje, a la conexión con el identificador *connectionId*, usando el canal confiable con el id *myReiliableChannelId*, de forma que el mensaje debe ser almacenado en *buffer* y la longitud del *buffer* debe ser definida por *bufferLength.*
De manera complementaria, para comprobar el estado del host, puede utilizar dos funciones:

```
NetworkTransport.Receive(out recHostId, out connectionId, out channelId, recBuffer, bufferSize, out dataSize, out error);
NetworkTransport.ReceiveFromHost(recHostId, out connectionId, out channelId, recBuffer, bufferSize, out dataSize, out error);
```

La primera función devolverá los eventos recibidos de cualquier *host* (y devolverá el identificador del *host* vía *recHostId*). La segunda comprueba el host con id recHostId.
Estas funciones pueden usaser dentro del método *Update* de Unity, que se ejecuta en cada paso de *frame* de la ejecución.

Siguiendo esta documentación llegamos a la implementación final realizada.

Implementación realizada

El proyecto que he desarrollado en Unity 3D consta de dos escenas, una para el modo *offline*, en la que se elige el modo de entrada (cliente o servidor), y otra *online* en la que se muestra un registro de los mensajes intercambiados entre el cliente y el servidor. La escena *online* es donde se encuentra el *Network Manager* que administra la conexión. La arquitectura y jerarquía del proyecto quedaría así:

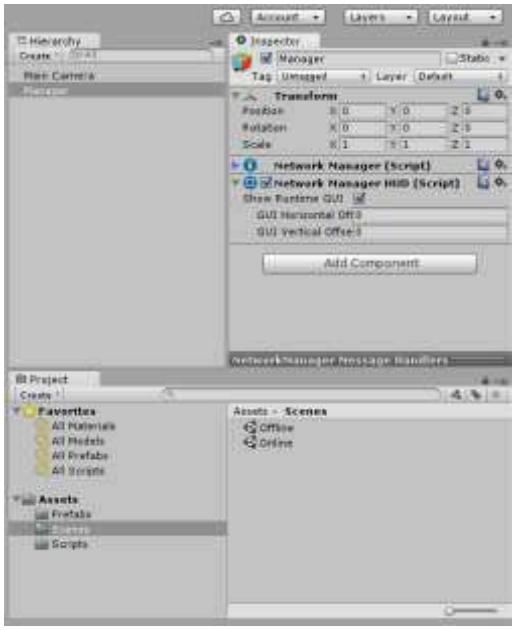


Figura 4.4 Jerarquía

Disponemos de dos clases, *Intercambio* y *UNETIntercambio*, clase padre e hija respectivamente. El comportamiento será *NetworkBehaviour* por razones obvias. La clase *Intercambio* sólo contiene un método implementado para añadir a un registro visible por pantalla el flujo de mensajes que se intercambien Cliente y Servidor.

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using UnityEngine.Networking;
using UnityEngine.Networking.NetworkSystem;
using UnityEngine.UI;

public class Intercambio : NetworkBehaviour {
    [SerializeField]
    private Text container;
    [SerializeField]
    private ScrollRect rect;

    // con esto veremos en el registro los mensajes enviados entre el cliente y servidor
    internal void AddMessage(string message)
    {
        container.text += "\n" + message;

        //just a hack to jump a frame and scrolldown the chat
        Invoke("ScrollDown", .1f);
    }

    public virtual void SendMessage(InputField input){}

    public virtual void PedirIndicaciones(InputField input){}

    public virtual void EnviarUbicacion(){}

    public virtual void EnviarUbicacionClientToServer(){}

    private void ScrollDown(){
        if (rect != null)
            rect.verticalScrollbar.value = 0;
    }
```

En la clase *UNETIntercambio* es donde se maneja todo el sistema de comunicación Cliente-Servidor. Primeramente, veamos todas las librerías necesarias para su desarrollo.

```
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Networking.NetworkSystem;
using System.Collections;
using UnityEngine.UI;
using System.Text;
```

Nada más arrancar el programa, registrará a quienes se conecten, con un único Servidor y la posibilidad de varios Clientes, aunque sólo necesitemos uno. Para un mayor entendimiento y reducción de la complejidad, los Clientes serán identificados por su ConnectionID. Además, se lanzará una co-rutina que arrancará el sistema de localización GPS, tal y como se hizo en el Capítulo III. También se lanzará una llamada a una función que enviará de forma periódica la localización GPS obtenida al servidor.

```
public class UNETIntercambio : Intercambio {

    private const short chatMessage = 131;

    public float latitude;
    public float longitude;


    private void Start(){
        // cuando arranca el servidor
        if (NetworkServer.active){
            // registramos el handler del servidor
            NetworkServer.RegisterHandler(chatMessage, ServerReceiveMessage);
        }

        //registramos el handler del cliente movil
        NetworkManager.singleton.client.RegisterHandler (chatMessage, ReceiveMessage);

        // lanzamos el servicio para obtener la localizacion
        StartCoroutine ("getLocation");

        // el cliente empieza a enviar la ubicacion
        EnviarUbicacionClientToServer ();
    }
```

Para obtener un *feedback* adecuado, llevamos un registro de los mensajes transmitidos. Para mostrarlo, disponemos del siguiente método que llama a la función antes mencionada de la clase padre.

```
 private void ReceiveMessage(NetworkMessage message){

     // lee el mensaje y lo añade al registro para ser visto
     string text = message.ReadMessage<StringMessage> ().value;
     AddMessage (text);
 }
```

Llegamos a la parte más importante. Cuando el servidor recibe un mensaje, aparte de hacerlo visible en el registro, debe clasificar ese mensaje dependiendo de si es una petición de indicaciones para localizar un objeto determinado de la escena en curso, o si es una mera actualización periódica de la localización o ubicación GPS del usuario.

Para hacer posible dicha identificación, se seguirá el siguiente protocolo:

- Si el cliente desea recibir una indicación, deberá escribir manualmente un mensaje de un carácter, y hacer *click* o pulsar en el botón SEND diseñado para ello.

- El resto de mensajes serán enviados de forma automática por el cliente, de forma transparente al usuario, actualizando las coordenadas GPS. Este mensaje llevará el siguiente formato: <longitudxlatitud>. Por ejemplo: 40.123x3.3499.

```
private void ServerReceiveMessage(NetworkMessage message){

    StringMessage myMessage = new StringMessage ();

    // connectionId sera el identificativo del cliente
    myMessage.value = message.conn.connectionId + ": " + message.ReadMessage<StringMessage> ().value;

    // el mensaje aparecera en el registro, visible para todos los clientes y el servidor
    NetworkServer.SendToAll (chatMessage, myMessage);

    // tratamiento del mensaje
    if (myMessage.value.Length == 1) {
        print (" el mensaje recibido es una peticion de indicaciones");

        // aqui debe comunicarse con el modulo de generacion de indicaciones existente
    }

 // si es un envio de ubicacion
 if(myMessage.value.Length > 1) {
    int c = 0;
    int posX = 0;
    StringBuilder sbLong = new StringBuilder ();                    // para el primer dato (longitud)
    StringBuilder sbLat = new StringBuilder ();                     // para el segundo dato (latitud)
    // primero averiguar en que posicion esta la x
    foreach( char ch in myMessage.value ){
        if(char.Equals(ch,'x')){
            posX = c;
        }
        else {
            c++;
        }
    }
    // ahora que tenemos la posicion de la x en posX, debemos obtener dos float del string |
    // obtenemos longitud
    for(int i = 0; i <= posX-1; i++){
        sbLong.Append(myMessage.value[i]);
    }
    // obtenemos latitud
    for(int j =posX+1; j < myMessage.value.Length; j++){
        sbLat.Append(myMessage.value[j]);
    }

    // ahora dejamos ambos datos como float
    float longRecibida = float.Parse(sbLong.ToString());
    float latRecibida  = float.Parse(sbLat.ToString());
    // estos dos datos deben ser transmitidos al sistema para reubicar al usuario

 } // if de si es envio de ubicacion
```

El siguiente método es el que se utiliza cuando el cliente necesita obtener las indicaciones necesarias para localizar un objeto. Hace uso del Input, que está relacionado con el botón SEND obtenido.

```
// para el cliente, pedir indicaciones : mensajes de un solo caracter
public override void SendMessage (UnityEngine.UI.InputField input){

    StringMessage myMessage = new StringMessage ();

    // aqui podremos enviar manualmente un mensaje
    myMessage.value = input.text;

    //sending to server
    NetworkManager.singleton.client.Send (chatMessage, myMessage);
}
```

El siguiente par de funciones se encargan de enviar cada dos segundos un mensaje desde el Cliente al Servidor con los últimos datos obtenidos de la ubicación; longitud y latitud, respetando el formato de envío mencionado anteriormente.

```
public override void EnviarUbicacionClientToServer (){
    if(Network.isClient){
        while(true){
              Invoke("EnviarUbicacion",2);
        }
    }
}


public override void EnviarUbicacion (){
    StringMessage mnsUbicacion = new StringMessage ();
    mnsUbicacion.value=""+longitude+"x"+latitude+"";
    NetworkManager.singleton.client.Send (chatMessage, mnsUbicacion);
}
```

Para finalizar de revisar cómo ha sido la implementación, tenemos de nuevo el desarrollo de la co-rutina lanzada al principio de la ejecución del programa. Con ella, como ya vimos en el Capítulo III, obtenemos la localización GPS del usuario en tiempo real.

```
IEnumerator getLocation () {

    // comprobamos si la ubicacion esta acivada
    if (!Input.location.isEnabledByUser) {
        yield break;
    }

    // si el usuario tiene la ubicacion activada
    else {
        while (true) {
            Input.location.Start ();
            latitude = Input.location.lastData.latitude;
            longitude = Input.location.lastData.longitude;
        }
    }
}
```

4.4.-Integración con el Sistema anterior

Cuando el Servidor hace el *parsing* correcto del mensaje recibido, en caso de que sea una petición de indicaciones, deberá comprobar sobre qué objeto se están pidiendo. En el sistema vigente existe la posibilidad de ver un desplegable con todos los objetos disponibles para localizar.

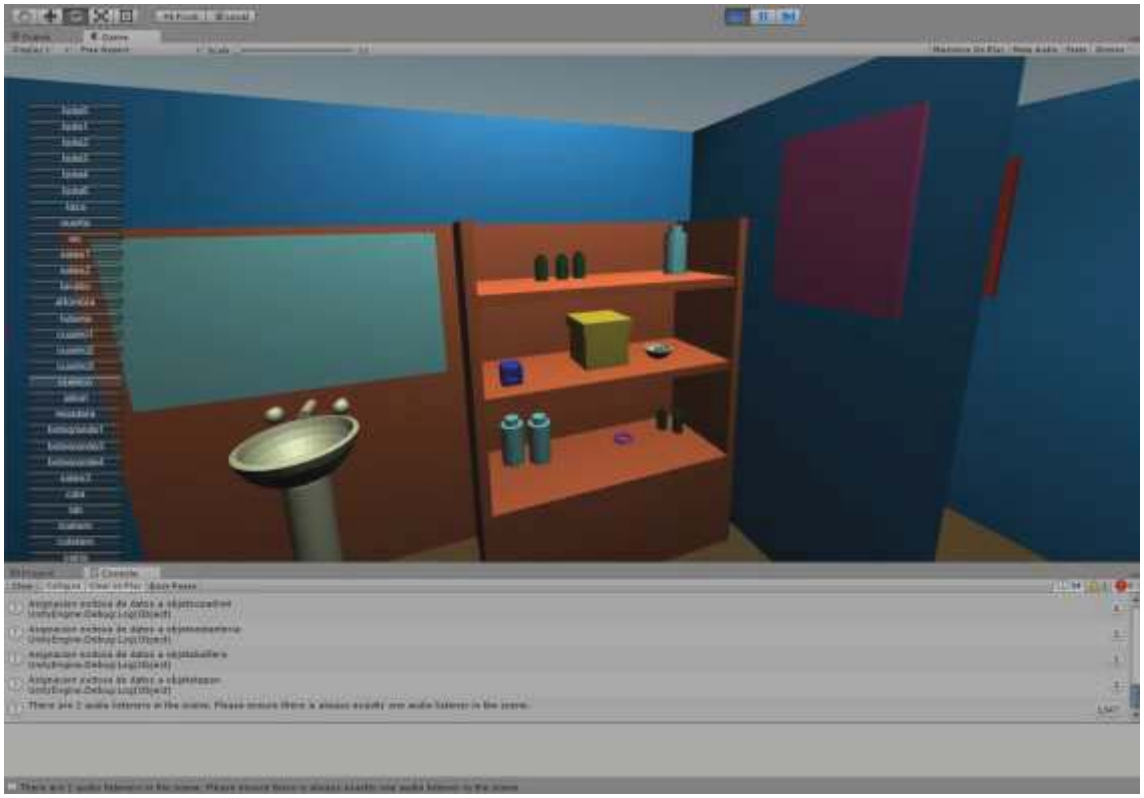


Figura 4.5 Visualización de la colección de objetos a localizar

En nuestro caso, cuando el Cliente lance una petición de generación de indicaciones para un objeto concreto, es necesario que se comunique al módulo ya implementado de generación de indicaciones para obtener dichas instrucciones en relación con la posición actual del usuario.

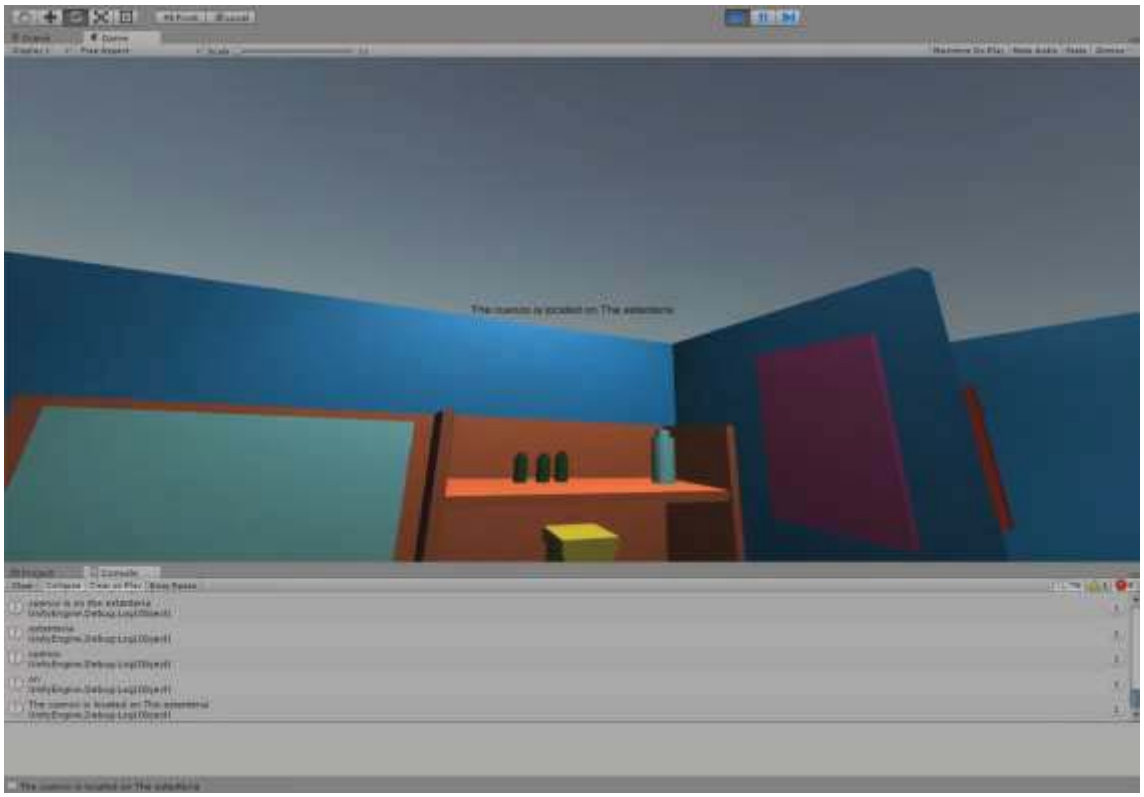


Figura 4.6 Generación de Indicaciones del Sistema

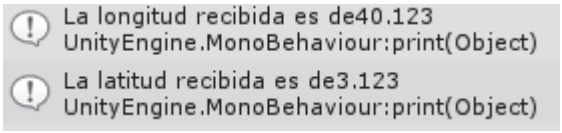
Por otro lado, se le deberán aportar los datos de actualización de la ubicación del usuario enviados al Servidor desde el Cliente, una vez que el Servidor haya separado la información recibida.

4.5.-Pruebas Realizadas

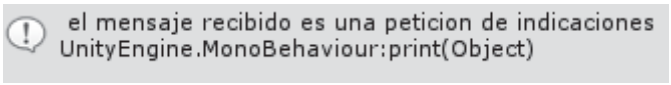
Las pruebas realizadas se han divido en diferentes módulos para su correcta comprobación. La parte de obtención de los datos de posicionamiento del usuario fue lógicamente llevada a cabo durante la realización del Capítulo III.

Otro tipo de pruebas realizadas fue para comprobar el correcto *parsing* a la hora de enviar y recibir mensajes con *StringMessage*, respetando el formato acordado para el envío de los mismos. Para ello, en un proyecto distinto en Unity, se creó una escena con un objeto vacío al que le añadimos el *script* para ejecutar las pruebas (esto es estrictamente necesario en Unity, y recomiendo hacer las pruebas sobre un proyecto Unity antes que sobre otro entorno de programación para C# ya que algunos métodos y estándares no son exactamente iguales). Se realizaron pruebas para los dos tipos de mensajes que puede recibir el Servidor.

Para un mensaje recibido con datos de ubicación, se comprueba que se separa correctamente ambos “float”, imprimiendo por pantalla el resultado.

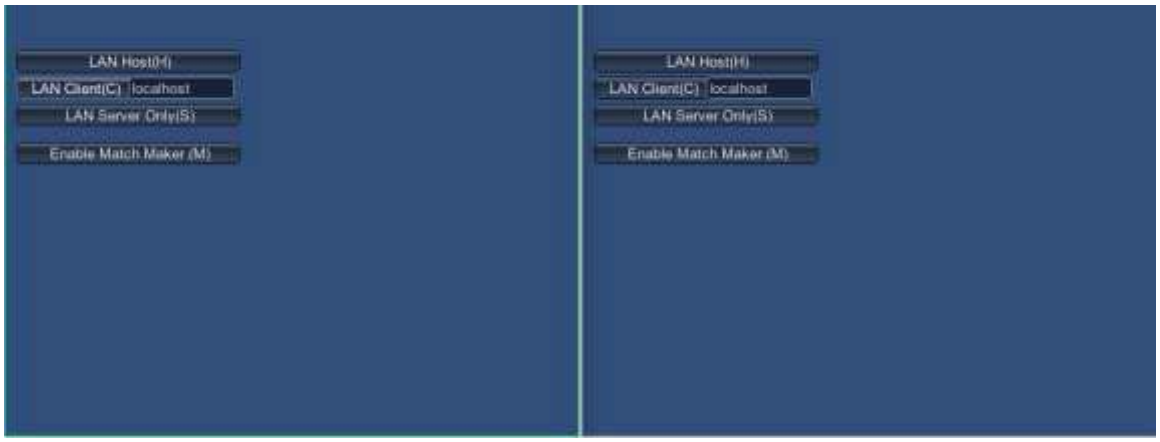


En caso de un mensaje recibido de sólo un carácter, que se refiere a un mensaje de petición de indicaciones, se muestra en pantalla un aviso que refleje dicha situación.

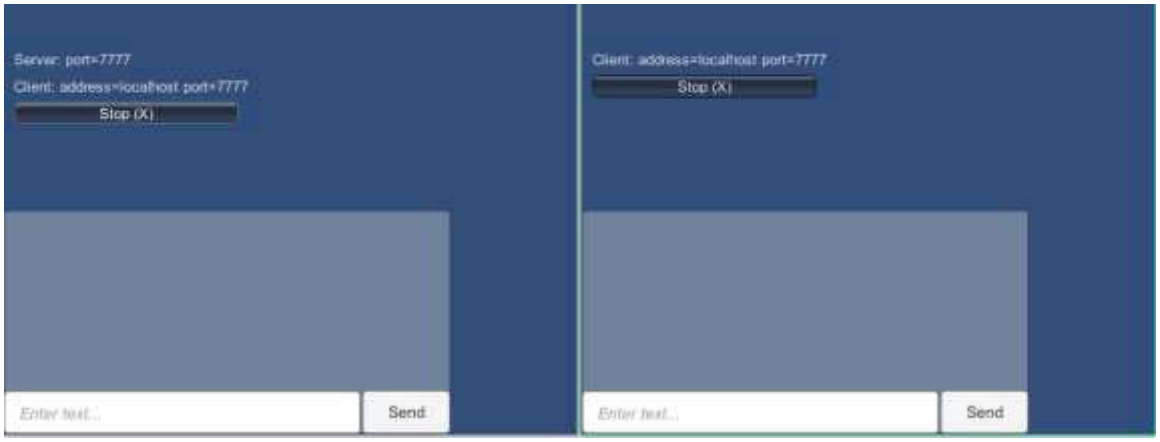


Por otro lado, en lo que se refiere al paso de mensajes, se hizo unas comprobaciones de conexión e intercambio de mensajes sin incluir toda la parte relacionada con la ubicación GPS ni los diferentes tipos de mensajes, para comprobar el correcto envío y retransmisión de mensajes.

A continuación vemos dos instancias de Unity ejecutando dichas pruebas, en la fase de elección de rol: Cliente o Servidor. Esta interfaz se provee en la escena *Offline* que vimos anteriormente.



A continuación, la instancia de la izquierda está trabajando como Servidor y la de la derecha como Cliente.



Para finalizar, vemos el paso de mensajes de dos actualizaciones de la ubicación junto a una petición de generación de indicaciones. El número que precede a cada mensaje es el *ConnectionId* con el que especificamos que íbamos a identificar al cliente (o clientes si en un fututo se necesitara).

NOTA: Las pruebas vistas están hechas en *localhost* y con dos instancias de Unity para poder obtener unas capturas con mejor resolución para esta memoria que las que se obtendrían de un móvil con una pantalla mucho menor. Para las pruebas en remoto, en vez de realizarse en *localhost*, se me facilitó de forma temporal una dirección IP para el servidor (junto a la máscara, DNS, etc. correspondientes).

5.- CONCLUSIONES, VALORACIONES Y LÍNEAS FUTURAS

Para finalizar, las conclusiones del proyecto y un resumen de cómo se ha llevado a cabo el desarrollo y monitorización del trabajo, comparando con lo planificado al inicio del mismo, evaluando los conocimientos adquiridos y enumerando las dificultades encontradas así como las posibles líneas futuras.

5.1.-Organización del trabajo y tiempo dedicado

Estoy realmente agradecido a mi tutora el haber mantenido el estilo de trabajo que ya llevamos en su momento durante la asignatura de Prácticum. La idea es hacer reuniones semanales cortas, de entre veinte y cuarenta y cinco minutos cada una, en las que se exponen los avances conseguidos respecto a la reunión anterior, se plantean nuevos objetivos y, si es necesario, se discuten las mejores maneras de cumplir los objetivos. De esta manera, se consigue un seguimiento muy práctico del proyecto.

En lo que al tiempo empleado se refiere, he llevado una bitácora al día en la que resumía el trabajo realizado y las horas dedicadas, con lo que poder hacer una comparación entre el tiempo planificado al principio del proyecto con el tiempo real necesitado, así como la obtención del cómputo global.

En el plan de proyecto se estipularon las siguientes tareas en concordancia con los objetivos del proyecto:

1.-Explorar diferentes métodos para la realización del modelo geométrico del mundo real, buscando la mayor automatización posible, analizando las ventajas y desventajas de cada uno.

2.-Diseñar un esquema para el uso del modelo de representación geométrica del mundo real previamente analizado, adaptando el ya existente sistema editor-exportador para ello.

3.-Implementar un mecanismo que consiga localizar la posición del usuario y su punto de vista en una interfaz cliente para móvil.

4.-Diseñar la interfaz de usuario que permita recibir las peticiones y dar las indicaciones necesarias desde el propio cliente en el móvil. Se realizará en dos maneras, la primera meramente textual y otra por voz.

5.-Adaptar el sistema actualmente existente para realidad virtual para que actúe como el servidor del cliente móvil.

6.-Diseñar escenarios y una batería de pruebas para la comprobación del buen funcionamiento del sistema.

7.-Ejecución de las pruebas.

8.-Documentación y preparación de la defensa.

En dicho plan de proyecto se programaba el trabajo a través de un diagrama de Gantt que resultaba de la siguiente forma:

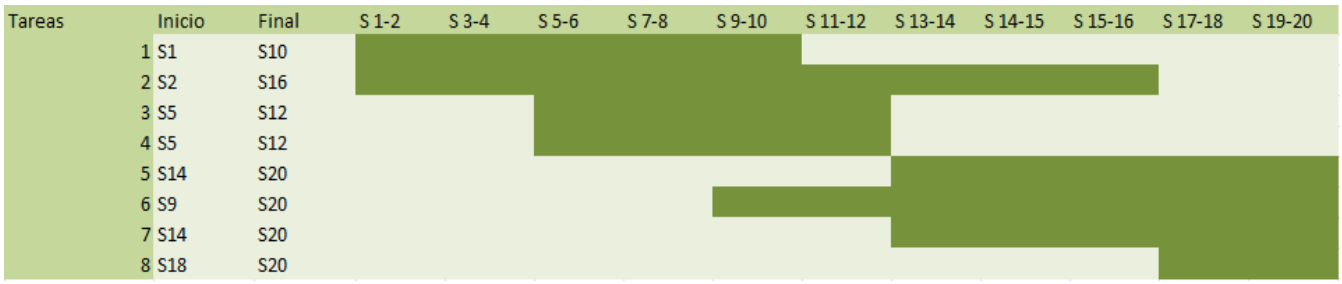


Figura 5.1 Diagrama de Gantt Planificado

Una vez después de finalizar el proyecto, y tras analizar los tiempos dedicados plasmados en la bitácora de trabajo, obtenemos:

-Para el Capítulo II, investigación de un modelo geométrico para el mundo real:
Semanas: Semana 1 (9 Febrero) – Semana 10 (4 Abril)
Tiempo: 43 horas de trabajo.

-Para el Capítulo III, Diseño de un mecanismo para la localización del usuario:
Semanas: Semana 7 (25 Marzo) – Semana 12 (27 Abril)
Tiempo: 38 horas de trabajo

-Para el Capítulo IV, Adaptación del Sistema a un modelo Cliente-Servidor:
Semanas: Semana 3 (24 Febrero) – Semana 18(7 Junio)
Tiempo: 183 horas de trabajo.

-Para la preparación de la Documentación:
Semanas: Semana 16 (13 Mayo) – Semana 18 (7 Junio)
Horas: 36 horas de trabajo.

-Para ensayo y preparación de la Defensa:
Semanas: Semana 18 (9 Junio) – Semana 20 (21 Junio).
Tiempo: 27 horas.

*Nota: la estipulación de las horas necesitadas para la preparación y ensayo de la defensa son sólo una estimación de las mismas, ya que la preparación es llevada a cabo después de la entrega del presente documento.

Por lo tanto, podemos concretar el número de horas totales empleadas en el desarrollo del trabajo de fin de grado en 327 horas, incluyendo reuniones y tutorías, y obtenemos el siguiente diagrama de Gantt finalizado y actualizado con las tareas y labores que se han llevado a cabo:

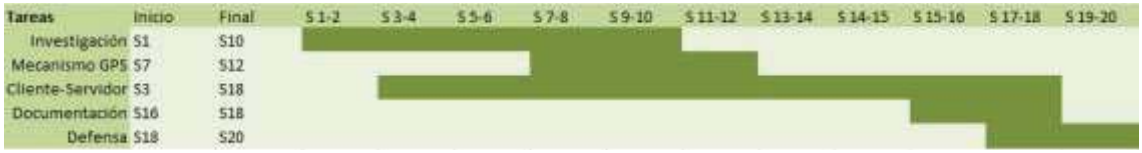


Figura 5.2 Diagrama de Gantt Final

Comparando con el diagrama de Gantt planificado de la Figura 5.1, podemos concluir que la implementación de un modelo Cliente-Servidor es la tarea que más tiempo ha requerido, si bien es verdad que se ha desarrollado en paralelo junto a las demás tareas.

En comparación con el tiempo establecido, todas las tareas han guardado una relación de similitud en cuanto a horas empleadas, a excepción del modelo Cliente-Servidor, como ya hemos mencionado.

5.2.-Dificultades Encontradas

En lo que a dificultades se refiere, dividiremos esta sección entre los tres capítulos desarrollados.

En el Capítulo I, para llevar a cabo la investigación, no he tenido problemas en encontrar detalles acerca de cómo funcionan y qué ventajas tendrían cada una de las dos tecnologías estudiadas, a saber; Project Tango de Google y Kinect Fusion de Microsoft. Además, la tutora me proporcionó diversos *papers* para consultar y ampliar información, cuyos detalles se pueden encontrar en la Bibliografía.

Para la parte de investigar técnicas de posicionamiento en interiores, resaltaría que en ocasiones, resulta necesario informarte de manera adicional del significado y uso de algunos términos y tecnologías, ya que es un ámbito muy técnico, y se requieren conocimientos sobre ondas, etc.

Por otro lado, a la hora de desarrollar una aplicación móvil en Android que permitiera recoger los datos de posicionamiento del usuario encontré dificultades al principio por querer desarrollar dicha aplicación con el entorno de programación de Android Studio, lo cual resultó ser mucho menos eficiente que implementarla en Unity 3D, el cual facilita en gran medida la exportación como apk de cualquier proyecto que lleves a cabo, a pesar de la configuración previa poco intuitiva que hay que realizar para ello.

En el Capítulo III, relacionado con adaptar el sistema a un modelo Cliente-Servidor, fue donde he encontrado un mayor número de dificultades. Por un lado, Unity es una herramienta pensada principalmente para el desarrollo de videojuegos. Es cierto que muchos investigadores usan esta potente herramienta de desarrollo para sus trabajos, ya que tiene una gran selección de librerías que ayudan para implementar algoritmos relacionados con temas variados como física, movimientos, *tracking*, realidad virtual o aumentada, etc. Sin embargo, Unity está orientado a la creación de videojuegos y gran parte de la documentación existente, consejos, así como tutoriales realizados, apenas aportaban valor a las soluciones que andábamos buscando. Más concretamente, a la hora de realizar un modelo Cliente-Servidor, toda documentación o propuesta encontrada solía proponer una arquitectura típica de videojuegos, con un servidor que acepta clientes (jugadores), manteniendo objetos sincronizados entre todos ellos. Sacar de ahí toda la información necesaria y sólo la necesaria, moldearla y adaptarla a nuestros objetivos e intereses, ha sido uno de los grandes retos del proyecto.

También, refiriéndonos a todo el proyecto en general, el tener que usar una versión ligeramente anticuada de Unity conlleva una gran problemática a la hora de consultar la documentación ofrecida por Unity. Dicha documentación es amplia y, en mi opinión, bastante buena, pero acceder a versiones más antiguas puede ser a veces más complicado, y en numerosas ocasiones suele estar a medio traducir, entiendo que por la salida de una nueva versión de la documentación en su momento. Recomiendo seguir siempre que sea posible la documentación inglesa original con el fin de evitar fallos de implementación debido a traducciones incorrectas o incompletas.

5.3.-Valoraciones

Para finalizar, valoro en gran medida los conocimientos adquiridos durante el desarrollo de este trabajo, incluyendo el entendimiento de las tecnologías de Project Tango y Kinect, así como de las diferentes técnicas para el posicionamiento en interiores vistas en la investigación del primer capítulo, el *scripting* en C# de los siguientes, desarrollando por primera vez una aplicación funcional en Android, viendo cómo funcionan los permisos y la gestión del GPS en dicho ecosistema, y trabajando en un modelo Cliente-Servidor con paso de mensajes.

En mi opinión, creo que se han dado unos importantes primeros pasos para llevar este proyecto, (iniciado por muchos compañeros en Prácticum y Trabajos de Fin de Grado anteriormente), al ámbito de la Realidad Aumentada, que podrá mejorar los resultados obtenidos previamente gracias a la profunda inmersión que genera la Realidad Aumentada.

5.4.-Líneas Futuras

Los siguientes pasos a dar en el proyecto serían:

- Obtener la tecnología de Project Tango en cuanto su precio y disponibilidad lo permitan.

- Estudiar su uso y poder generar así una reconstrucción geométrica del mundo (escena a tratar).

- Acoplar la parte del Servidor ya realizada con el generador de indicaciones por voz del sistema existente cuando se recibe un mensaje por parte del cliente solicitando las mismas, tal y como se describe en la sección 4.4 de este documento.

- Modificar la interfaz de usuario del cliente (actualmente usa la misma que el servidor), para que tenga acceso a poder enviar peticiones de indicaciones (ya implementado) con una visualización escrita de las mismas para corroborar la información recibida de manera oral. El cliente necesitará además poder visualizar el desplegable de posibles objetos a localizar del que ya dispone el sistema.

6.-REFERENCIAS CONSULTADAS

[1] [Google Tango, 2017] https://get.google.com/tango/ consultado en Febrero de 2017.

[2] [Xataka Project Tango, 2017] https://www.youtube.com/watch?v=yPnf-0GT3Uk&index=1&list=WL consultado en Febrero de 2017.

[3] [Xataka Lenovo Phab 2 Pro, 2017] https://www.xataka.com/moviles/lenovo-phab-2-pro-analisis-la-realidad-aumentada-tambien-necesita-una-killer-app consultado en Febrero de 2017.

[4] [Xataka review Lenovo Phab 2Pro,2017] https://www.youtube.com/watch?v=S22Pd4tv3dU consultado en Febrero de 2017.

[5] [Andro4all Project Tango, 2017] https://andro4all.com/2016/06/project-tango-google-caracteristicas-detalles consultado en Febrero de 2017.

[6] [Topes de Gama Lenovo Phab 2 Pro, 2017] https://www.youtube.com/watch?v=42pSKdWn9_Y consultado en Febrero de 2017.

[7] [Wikipedia Project Tango, 2017] https://es.wikipedia.org/wiki/Project_Tango consultado en Febrero de 2017.

[8] [Google Project Tango Developers, 2017] https://get.google.com/tango/developers/ consultado en Febrero de 2017.

[9] [Google Project Tango Unity Developers, 2017] https://developers.google.com/tango/apis/unity/ consultado en Febrero de 2017.

[10] [Google Project Tango Motion Tracking, 2017] https://developers.google.com/tango/apis/unity/unity-howto-motion-tracking consultado en Marzo de 2017.

[11] [Nico Kuhn, 2017] Nico Kuhn, “Project Tango: core technologies,development tools”

[12] [Xbox One Kinect, 2017] http://www.xbox.com/es-ES/xboxone/accessories/kinect consultado en Marzo de 2017.

[13] [Microsoft Developers, 2017] https://developer.microsoft.com/es-es/windows/kinect consultado en Marzo de 2017.

[14] [Wikipedia Kinect, 2017] https://es.wikipedia.org/wiki/Kinect consultado en Marzo de 2017.

[15] [Izadi, 2011] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, Andrew Fitzgibbon, "KinectFusion: Real-time 3D Reconstruction and Interaction
Using a Moving Depth Camera".

[16] [Microsoft Library, 2017] https://msdn.microsoft.com/en-us/library/dn188670.aspx consultado en Marzo de 2017.

[17] [Lembit Valgma, 2016] Lembit Valgma, "Lembit Valgma 3D reconstruction using Kinect v2 camera"

[18] [Dwiyasa, 2016] Felis Dwiyasa, Meng-Hiot Lim, "A Survey of Problems and Approaches in Wireless-based Indoor Positioning"

[19] [Davidson, 2016] Pavel Davidson, Robert Piché, "A Survey of Selected Indoor Positioning Methods for Smartphones"

[20] [Indoor Positioning system, 2017] https://senion.com/indoor-positioning-system/ consultado en n Marzo de 2017.

[21] [UGR, 2017] http://www.ugr.es/~bioestad/_private/cpfund4.pdf consultado en Marzo de 2017.

[22] [Unity Manual, 2017] https://docs.unity3d.com/es/current/Manual/ consultado en Abril de 2017.

[23] [Unity SDK, 2017] https://docs.unity3d.com/es/current/Manual/android-sdksetup.html consultado en Abril de 2017.

[24] [Unity Location, 2017]
https://docs.unity3d.com/ScriptReference/LocationInfo.html consultado en Abril de 2017.

[25] [Android Studio, 2017] https://developer.android.com/studio/install.html consultado en Abril de 2017.

[26] [Torres, 2016] David Torres Madrigal, "Exploración y evaluación de mecanismos para la captura de la posición y postura de personas en entornos reales".

[27] [Unity NetUsingTransport, 2017]
https://docs.unity3d.com/es/current/Manual/UNetUsingTransport.html consultado en Mayo de 2017.

[28] [Unity NetworkClient, 2017]
https://docs.unity3d.com/es/current/Manual/class-NetworkClient.html consultado en Mayo de 2017.

[29] [Unity NetworkServer, 2017]
https://docs.unity3d.com/es/current/Manual/class-NetworkServer.html consultado en Mayo de 2017.

[30] [Unity Network, 2017]
https://docs.unity3d.com/ScriptReference/Network.html consultado en Mayo de 2017.

[31] [Frostybay UNET, 2017]
http://frostybay.com/articles/a-simple-tutorial-with-unity-unet consultado en Mayo de 2017.

[32] [Unity 3D Parsing, 2017]
http://answers.unity3d.com/questions/993908/how-to-parse-a-string-between-certain-characters.html consultado en Mayo de 2017.

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Wed Jun 07 16:10:36 CEST 2017
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)